Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

Vývoj a ověření aplikace na podporu výuky HACMP clusteru pod OS IBM AIX

Diplomová práce

Bc. Dagmar Bendová

Školitel: RNDr. Libor Dostálek

České Budějovice 2015

Věnováno manželovi Pavlovi a dceři Rebece.

Bendová D., 2015: *Vývoj a ověření aplikace na podporu výuky HACMP clusteru pod OS IBM AIX*. [Development and verification application for support of HACMP cluster education in IBM AIX OS. Mgr. Thesis, in Czech.] – 171 p. (počet stran), Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic.

Anotace:

This thesis describes development and verification of application for support of HACMP clustering education in IBM AIX OS. In fact, development of this application can help students to understand basic functions of this type of cluster. Users, eventually, can verify full function of input configuration cluster in real environment. Ultimately, it can simulate basic cluster function and create basic configuration file, which can be direct apply to operational installation of HACMP software of version 5.3.

Prohlašuji, že svoji diplomovou práci jsem vypracovala samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

……

Dagmar Bendová

5. 4. 2015

Poděkování

Na tomto místě bych ráda poděkovala vedoucímu práce RNDr. Liborovi Dostálkovi za jeho cenné odborné a pedagogické rady a připomínky při zpracování tohoto námětu. Dále bych ráda poděkovala kolegům Vlastimilu Havlovi, Ing. Jaroslavovi Schurrerovi a Martinu Svárovskému z firmy Telefónica O2, a.s., za oporu při studiích a za jejich odborné náměty a připomínky k tomuto programu. Rovněž bych chtěla poděkovat své úžasné rodině a přátelům Lence Vybíralíkové, Jiřině Vaškové a Karlovi Melenovi za neutuchající podporu v celém průběhu studia.

OBSAH

1 Úvod

Tento projekt vyplynul z dlouholetých zkušeností autorky v oboru administrace clusterů. Na základě těchto zkušeností byla zmapována potřeba vytvoření softwaru, který by pomohl nejen nově příchozím administrátorům orientovat se v této problematice, ale i stávajícím administrátorům při plánování, analýze, přípravě a provozu clusteru. Tento sofware byl vyvíjen i s přihlédnutím na zkvalitnění kooperace mezi jednotlivými částmi personální infrastruktury v datových centrech. Z důvodu separace jednotlivých částí infrastruktury na pododdělení dochází k nekonzistenci a k nedorozumění mezi jednotlivými požadavky na přípravu vhodné infrastruktury clusteru. Tento software tedy napomůže i ke zlepšení komunikace mezi jednotlivými odděleními datového centra.

Samotná aplikace byla vyvíjena s přihlédnutím na hlavní učební obor autorky. Jelikož by nadále chtěla působit jako učitelka informatiky na střední škole, byla aplikace koncipována s ohledem na vytvoření a demonstrace didaktické pomůcky, na které je možné demonstrovat učební látku, v tomto případě funkčnost clusteru.

Jako model pro demonstraci clusterové problematiky byl vybrán cluster software HACMP [1,2] pod OS AIX[3], kterému se autorka momentálně nejvíce věnuje.

1.1 Uspořádání textu

Text je uspořádán do deseti kapitol. V první kapitole je představen samotný projekt a jednotlivé realizační cíle. Druhá kapitola se zabývá teorií clusterové problematiky. Třetí a čtvrtá kapitola je věnována analýze vývojového prostředí. Pátá a šestá kapitola je

[1] High Availability Cluster Multi-Processing provozovaný pod operačním systémem AIX od firmy IBM. Pro vývoj aplikace byl použit software HACMP verze 5.3 (dále jen HACMP)

[2] HACMP. *TheFreeDictionary.com* [online]. Nedatováno. [cit. 2015-03-28]. Dostupné z: http://acronyms.thefreedictionary.com/HACMP

[3] Operační systém AIX firmy IBM verze 5.3 (dále jen AIX)

zasvěcena samotnému vývoji softwaru. Sedmá kapitola je uživatelská příručka použití aplikace. Poslední kapitoly obsahují zhodnocení realizace projektu, návrhy na možné budoucí rozšíření a závěrečné zhodnocení projektu.

1.2 Cíle

Primární cíl aplikace je umožnit studentovi vyzkoušet si vytvořit cluster v testovacím prostředí předtím, než bude schopen aplikovat své zkušenosti na provozní prostředí. Hlavní cíle vývoje:

1) Vytvořit funkční clusterovou konfiguraci v architektuře *standby* (active/passive)[1]

2) Průběžně předkládat studentovi grafický výstup z vytváření clusterové konfigurace. Díky tomu má student zpětnou odezvu na jednotlivé dílčí kroky při sestavování clusterové konfigurace a to mu umožní získat ucelený přehled o clusterové problematice.

3) Realizovat funkci *TAKEOVER* [2]a to provedením grafické simulace přepnutí resource groupy z primárního nodu na sekundární nod.

4) Realizovat funkci *FAILOVER* [3]a to provedením grafické simulace výpadku primárního nodu a přepnutí resource groupy[4] na sekundární nod.

---

[1] Konfigurace clusteru v režimu stand-by, kdy primární nod je aktivní (resource groupa je online) a sekundární nod je pasivní (resource groupa je offline) (viz 2.5.1).

[2] Aktivní funkce TAKEOVER v konfiguraci stand-by (active/passive) režimu clusteru je volána v případě manuálního přesunutí resource groupy z primárního nodu clusteru na sekundární nod (viz 2.7.5)

[3] Aktivní funkce FAILOVER v konfigurace stand-by (active/passive) režimu clusteru je volána v případě havárie serveru a implementuje automatické přesunutí resource groupy z primárního nodu clusteru na sekundární nod (viz 2.7.1)

[4] Resource groups (dále RG). Logická jednotka, jejíž součástí je samotná aplikace a nad kterou cluster provádí jednotlivé aktivity jako TAKEOVER, FAILOVER apod.

5) Vytvořit soubor spustitelný v AIX z důvodu jeho další implementace do provozní instalace softwaru HACMP clusteru. Po spuštění souboru bude na základě definovaných údajů vytvořen cluster.

Sekundární cíl vývoje (cíl, který vyplynul z vývoje aplikace):

Vytvoření funkčního API[1] pro další možné programové moduly např. podpůrný program pro výuku virtualizace apod.

1.3 Struktura přiloženého CD

Samotný obsah CD je rozčleněn do čtyř částí základní adresářů.

1) JAVA - runtime java JRE 7 Update 67
2) APL - binární kód aplikace
3) SRC - jednotlivé implementační třídy aplikace v podobě zdrojového kódu
4) DOC - diplomová práce, manuál k ovládání aplikace

V kořenovém adresáři CD je uložen soubor README.txt, který obsahuje podrobnou informaci ohledně instalace, provozu a odinstalace aplikace. Dále obsahuje soubor setup.exe, pomocí kterého je spuštěna samotná instalace aplikace.

1.4 Použité technologie

Design aplikace byl vytvořen ve webové aplikaci Mockup firmy Balsamiq (viz 3.2).

Pro samotnou realizaci vývoje aplikace byl použit nástroj NetBeans 7.4 Patch 3 s technologií JDK 1.7. Pro realizaci dílčích částí byly využity tyto knihovny:

- Swing Application Framework 1.1
- Swing Layout Extension 1.0.4

[1] Application Programming Interface

- Hibernate 1.25.3.1
- Java Persistence 1.25.1
- Derby 1.24.1
- Chunk 2.6.3
- FileUtils 1.3.1_09-85
- Ibatis 2.3.4.726
- Java Tree Api 1.7.0_17-b02

1.5 Metodika vývoje aplikace

Vývoj aplikace probíhal v několika krocích, které se vzájemně prolínaly. Jednotlivé dílčí kroky uvedených cílů byly realizovány v těchto oblastech:

- Analýza projektu v oblasti teoretické části clusteru
- Analýza a návrh vhodného prostředí projektu
- Analýza projektu z pohledu použitých knihoven
- Vývoj globálního API pro realizace clusterové logiky programu
- Implementace clusterové logiky do globálního API
- Metodika používání aplikace včetně impementace realizace jednotlivých funkčních („clusterových“) částí aplikace

Vývoj aplikace lze i jednodušeji rozdělit do dvou na sobě nezávislých celků. První část se týkala výzkumu na úrovni HACMP clusteru a aplikace požadovaných funkcionalit do výsledného programu. Druhá část výzkumu se věnovala programování v Javě a včlenění výstupu z výzkumu HACMP clusteru do samotné implementace programové části.

2 TEORIE

Cílem této kapitoly je obeznámit čtenáře se základními teoretickými pojmy v oblasti konceptů HACMP technologie a vybavit čtenáře teoretickými znalostmi, které bude potřebovat v dalších kapitolách.

2.1 Definice clusteru

Serverový cluster je termín pro nezávislou skupinu počítačových systémů označovaných jako nody nebo uzly. Tyto uzly fungují společně jako jeden systém s cílem zajistit klientům co největší dostupnost. Uzly spolu přes interní infrastrukturu neustále komunikují a v případě selhání jednoho uzlu, veškerý provoz převezme další uzel v clusteru dle nastavené specifikace. Počet uzlů v clusteru závisí na podpoře provozovaného systému.

2.2 Typy clusterů

Podle způsobu jejich využití rozlišujeme tyto typy clusterů:

2.2.1 High-performance computing cluster

Tento typ clusteru se používá pro řešení výpočetně náročných úloh (fyzikální a matematické výpočty). Pro zvýšení výkonu se propojí výkon jednotlivých uzlů. Je to výhodnější než sestrojit jeden počítačový systém s požadovaným výkonem. Výpočet se provádí na základě tzv. paralelizace, která umožňuje výpočet rozdělit do několika nezávislých výpočtů. Aplikace musí být vyvíjena s podporou paralelizace[1]. Podle Rouse[2] se většinou tento druh clusteru realizuje v případě požadovaného výpočetní

[1] Výpočetní cluster. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-04-06]. Dostupné z:http://cs.wikipedia.org/wiki/V%C3%BDpo%C4%8Detn%C3%AD_cluster

[2] *Rouse, M.* (Nedatováno). What is high-performance computing (HPC)?. WhatIs.com. [online]. [cit. 2015-03-21]. Dostupné z: http://searchenterpriselinux.techtarget.com/definition/high-performance-computing

výkonu na systém větší než teraFLOP[1]. Tento druh clusteru si většinou pořizují vědecké instituce. Jejich využití bývá i pro vojenské účely, předpověď počasí, finanční analýzy aj. Poskytovatelé tohoto typu clusteru jsou např. firma DELL, HP aj.

2.2.2 High-availability cluster

Tento typ clusteru preferuje a je hlavně založený na splnění měřítka vysoké dostupnosti. Tento typ cluster je koncipován s minimálním *down-time* dobou. (tj. dobou, v rámci které je server popř. aplikace nedostupná.) Aby bylo možné dosáhnout takového požadavku, jednotlivé prvky clusteru jsou koncipovány redundantně. V případě nedostupnosti hardwarového prvku uzlu, převezme provoz záložní hardwarový prvek daného uzlu popř. jiný uzel v clusteru dle nastavené specifikace. Přepnutí je vyvoláno přes funkci FAILOVER (viz 2.7.1), která ihned reaguje na nedostupnost a na základě toho vyvolá správnou reakci/přepnutí. Tento typ clusteru se často používá pro kritické databáze, sdílení souborů na síti business aplikace a zákaznické aplikace jako komerční web aplikace. Redundance je realizována na síťové úrovni, na úrovni nodů, SAN[2] popř. NAS[3] infrastruktuře. Pro interní komunikaci se využívá tzv. systém *heartbeat*ů. *Heartbeat* je privátní, neveřejná síť dostupná jen pro nody clusteru, která je primárně určená ke komunikaci mezi nody a pro detekování nedostupnosti jednoho z nodu clusteru[4]. V případě nedostupnosti všech interních linek (*heartbeat*ů) se musí zabránit startu aplikací na všech uzlech tím, že se definují privátní, neveřejné komunikační linky mezi nody clusteru. Mezi tento typ řadíme např. cluster HACMP od firmy IBM, Microsoft Windows clustery, HP OpenVMS cluster aj.

---

[1] **FL**oating-point **OP**erations per **S**econd, měření výkonu počítačů. *FLOP*. [online]. Nedatováno. [cit. 2015-03-18]. Dostupné z WhatIs.com: http://whatis.techtarget.com/definition/teraflop

[2] Storage Area Network

[3] Network Attached Storage

[4] *Heartbeat*. [online]. Nedatováno. [cit. 2015-03-08]. Dostupné z IBM: http://www-01.ibm.com/support/knowledgecenter/SSPHQG_7.1.0/com.ibm.powerha.admngd/ha_admin_config_hacmp_heartbeat.htm

2.2.3 Load ballancing cluster

Tento typ clusteru je založen na rozložení zátěže mezi jednotlivými uzly. Služba je poskytována paralelně. Obsah se replikuje mezi všechny uzly clusteru. Všechny uzly mají jednotnou virtuální IP a ARP[1] adresu. Klient, který komunikuje s touto virtuální adresou, může komunikovat s jakýmkoliv uzlem v clusteru. Takto je zajištěna neustálá dostupnost služby, a pokud některý z nodů neběží, klient komunikuje s ostatními nody. Neopomenutelnou výhodou je i možnost rozložení zátěže na všechny nody clusteru dle specifikovaného poměru. Typické využití tohoto typu clusteru je v oblasti web serverů, DNS[2], VPN serverů a terminálových farmách. Mezi nejznámější projekty tohoto typu je Windows NLB cluster, OpenSCE aj.

2.2.4 Storage cluster

Tento druh clusteru se zaměřuje na správu diskové kapacity. Pro dosažení vyššího toku dat, paralelizace dat a zajištění vyšší spolehlivosti je disková kapacita rozložena mezi více nodů. Pro tento typ clusteru se využívají speciální souborové systémy, které jsou schopny zajistit rozložení zátěže, redundanci dat, pokrytí výpadku jednotlivých uzlů a distribuci mechanismů zamykání souborů[3]. Mezi známé projekty, které poskytují clustery tohoto typu, patří Open-e[4], Ceph[5] aj.

2.2.5 Grid cluster

Tento druh clusteru je tvořen nody, které poskytují clusteru výpočetní výkon, ale jejich primární funkce je jiná. Jednotlivé nody provozují nehomogenní OS a vzájemná

[1] Address Resulation Protocol

[2] Domain Name System

[3] Počítačový cluster. *Wikipedie* [online]. Nedatováno. [cit. 2015-04-05]. Dostupné z: https://cs.wikipedia.org/wiki/Po%C4%8D%C3%ADta%C4%8Dov%C3%BD_cluster

[4] HA Storage Cluster | Open-E. [online]. Nedatováno. [cit. 2015-03-11]. Dostupné z: http://www.open-e.com/solutions/ha-storage-cluster/

[5] *Ceph Storage Cluster*. [online]. Nedatováno. [cit. 2015-03-25]. Dostupné z: http://ceph.com/docs/master/rados/

součinnost je řízena na úrovní aplikačního softwaru. Pro komunikaci se nevyužívá privátní interní síť, ale nody mohou být umístěny kdekoliv. Z toho vyplývá, že pro komunikaci mezi nody lze využít LAN, WAN nebo celý Internet [19]. Tento typ clusteru se často využívá v oblastech vývoje a testu. Mezi produkty patří např. bwGRID cluster,[1] Oracle GRID aj.

2.3 High availability koncept

Koncept vyšší dostupnosti je nasazován všude tam, kde je potřeba eliminovat plánovanou nebo neplánovanou nedostupnost aplikace. Tohoto lze dosáhnout zmapováním možných nedostupností aplikace pomocí koncepce SPOF[2] a na základě této koncepce předvídat možné dopady v případě nedostupnosti aplikace a vybrat vhodný hardware popř. kvalitní konfiguraci softwaru. Vyšší dostupnost tedy znamená chod aplikace bez přerušení jejího běhu.

2.4 SPOF (Redukce „single points of failure“)

Každý prvek v zaběhnuté infrastruktuře může způsobit nedostupnost aplikace. Proto se snažíme eliminovat tyto možné přičiny na více úrovních. Snaha každého clusterového řešení je eliminovat všechny možné nedostupnosti na níže uvedených úrovních:

***NOD*y clusteru**

Základní myšlenka high availability konceptu je navržení takového clusterového řešení, kdy v případě výpadku jednoho nodu clusteru převezme aplikaci jiný uzel clusteru, tak aby se nepřerušila činnost clusteru. Při navrhování strategie umístění serverů do clusteru, je potřeba počítat i s možností vytopení serverovny popř. s výpadkem napájení v celém datovém centru. V tomto případě se přistupuje u velmi důležitých aplikací k umisťění nodů clusterů do různých místností či lokalit.

[1] *BwGRiD Cluster*. [online]. Nedatováno. [cit. 2015-03-20]. Dostupné z: http://www.urz.uni-heidelberg.de/server/grid/index.en.html

[2] Single points of failure – jednotlivé hardwarové komponenty, které mohou způsobit jeho nedostupnost.

Napájení

V konceptu napájení nodu clusteru se většinou využívají tzv. několika okruhové rozvody. V případě výpadku jednoho okruhy převezme kompletní činnost napájení záložní okruh.

Síťové adaptéry

Při konfiguraci clusteru je samozřejmostí využití redundantních síťových adaptérů. Běžnou chybou je například neredundantní využití dalších síťových prvků jako switch či router. U každého nodu clusteru je potřeba realizovat duplicitní zapojení nodu clusteru i do switche popř. routeru v jiné místnosti než je umístěn samotný server. V případě kompletního výpadku jedné síťové cesty (NIC, switch, router) je tím zajištěn plynulý přenos dat.

Virtualizace

Redundanci je potřeba zajistit i na úrovni virtuálních serverů, které zajišťují správu IO zdrojů.

V případě virtualizace na IBM platformě je potřeba počítat i s duplikací dat na úrovni Virtual Input/Output serverů (dále jen VIO)[1] . Jedná se o technologii, která virtualizuje IO[2] adaptéry pro jednotlivé virtuální servery. Funkcionalita VIO serveru spočívá v přiřazení a správě fyzických IO adaptérů např. NIC. Tyto fyzické adaptéry na úrovni VIO serverů dále virtualizujeme jednotlivým koncovým virtuálním serverům. Tento způsob se realizuje v případě, že chceme sdílet fyzické adaptéry pro více virtuálních serverů. Implementace koncepce vyšší dostupnosti v tomto případě je, že v případě výpadku jednoho VIO serveru plynule převezme komunikaci záložní VIO server.

[1] Virtual I/O server. *Virtual I/O server* [online]. Nedatováno. [cit. 2015-04-05]. Dostupné z: http://www-01.ibm.com/support/knowledgecenter/POWER7/p7hb1/iphb1kickoff.htm?cp=POWER7%2F1-8-3-5

[2] Input/Output adapter

Ostatní prvky High Availability konceptu

Pro splnění požadavku konceptu vyšší dostupnosti je potřeba u každého nodu clusteru nastavit mirrorování disků na úrovni RAID[1] technologie, využívat aplikační monitory, nastavit logické ošetření stavů v případě nedostupnosti aplikace, používat redundantní hardwarové cesty apod.

2.5 HACMP koncept

Operační systém *AIX* implementuje clusterový manager *HACMP*, který přispívá ke stabilitě celého systému. Tato služba běží v konceptu *SRC*[2] a je implementována *DAEMON*y[3], kteří zajištují obnovu clusteru v případě havárie hw komponenty. Tato služba také monitoruje a koordinuje aktivity mezi jednotlivými nody. Dále koncept zahrnuje i rysy jako Hot Swap Adaptéry[4,5], ECC[6,7], FFDC[8,9] aj.

---

[1] Redundant Array of Independent Disks, způsob zabezpečení diskové kapacity v případě selhání disku

[2] System Resource Controller. *System Resource Controller* [online]. Nedatováno. [cit. 2015-04-05]. Dostupné z: http://www-01.ibm.com/support/knowledgecenter/ssw_aix_71/com.ibm.aix.osdevice/sysrescon.htm

[3] Disk And Execution MONitor; In Unix, what is a daemon? [online]. Nedatováno. [cit. 2015-03-28]. Dostupné z: https://kb.iu.edu/d/aiau

[4] Nahrazení hardwarové komponenty bez nutnosti realizovat nedostupnost počítačového systému

[5] *Hotswap*. [online]. Nedatováno. [cit. 2015-03-19]. Dostupné z WhatIs.com: http://whatis.techtarget.com/definition/hot-swap

[6] Error checking/correction memory (Kontrola a oprava chyb v paměti), funkcionalita integrovaná u pamětí DDR2,3 apod. V případě chybné interpretace bitu tato funkcionalita, pomocí několika bitů navíc, opraví chybný kod

[7] *Rouse, M.* (Nedatováno). *ECC*. [online]. [cit. 2015-03-02]. Dostupné z: http://searchnetworking.techtarget.com/definition/ECC

[8] First Failure Data Capture, ochrana kritických diagnostických dat před neočekávaným pádem systému

[9] *FFDC*. [online]. Nedatováno. [cit. 2015-03-17]. Dostupné z FFDC: http://www-01.ibm.com/support/knowledgecenter/SSPHQG_6.1.0/com.ibm.hacmp.admngd/ha_admin_first_failure_data_capture.htm

2.5.1 HACMP charakteristika

Celý koncept je založen na službě RSCT[1,2], která umožňuje vybudovat dvě možné varianty prostředí clusteru:

1. Sériové prostředí – v HACMP terminologii je tento typ clusteru označován jako standby (active/passive). Aktivace aplikace je realizována jen na primárním nodu clusteru. Ostatní nody slouží jako záložní (pasivní) a jsou aktivovány jen v případě nedostupnosti primárního nodu

2. Paralelní prostředí – v HACMP terminologii je tento typ clusteru označován jako mutual (active/active). Aplikace je aktivní na všech nodech clusteru. Žádný nod není pasivní.

2.5.2 HACMP základní topologie

Topologie clusteru se člení z pohledu celé infrastruktury clusteru na tyto jednotlivé subkategorie:

- Toplogy
- Resources
- Resources groupy (RG)
- Customizace

[1] IBM Reliable Scalable Cluster Technology.

[2] *IBM Knowledge Center*. [online]. Nedatováno. [cit. 2015-02-08]. Dostupné z IBM Knowledge Center: http://www-01.ibm.com/support/knowledgecenter/SGVKBA_3.1.4/com.ibm.rsct314.aixcmds/idrsct_aixcmds_kickoff.htm

2.5.3 HACMP subkategorie topology

V této subkategorie se definují základní komponenty clusteru. Mezi ně patří nody clusteru, networking a sdílená storage.

NODy

Jednotlivé nody clusteru mohou být jak fyzické servery, tak i virtuální servery např. LPARy[1,2].

Networking

Síťové služby v clusterové topologii se definují v několika úrovních:

1) Boot IP - každý nod clusteru má definovanou tzv. bootovací IP adresu, definovanou na úrovni OS. Jedná se o síťovou adresu, která je dostupná v případě, že clusterový software HACMP neběží. Nepoužívá se pro administraci HACMP clusteru. Pro tento způsob komunikace se běžně využívají fyzické, virtuální adaptéry popř. etherchannel[3,4].

2) Service IP nodu (Persistent IP) – persistentní IP adresa nodu, není možné ji přesouvat mezi nody, je definovaná na úrovni OS a slouží jen k administraci HACMP clusteru

[1] Logical PARtision – virtuální servery vytvořené a provozované na fyzickém serveru. Toto označení je používáno pod platformou AIX

[2] *LPAR Configuration and Management Working with IBM eServer iSeries Logical Partitions*. [online]. Nedatováno. [cit. 2015-03-26]. Dostupné z An IBM Redbooks publication: http://www.redbooks.ibm.com/abstracts/sg246251.html

[3] Technologie na podporu agregace síťových component

[4] *EtherChannel and IEEE 802.3ad Link Aggregation*. [online]. Nedatováno. [cit. 2015-03-16]. Dostupné z EtherChannel and IEEE 802.3ad Link Aggregation: https://www-01.ibm.com/support/knowledgecenter/ssw_aix_61/com.ibm.aix.networkcomm/etherchannel_intro.htm

3) Heartbeat – určený pro přímou komunikaci mezi dvěma nody a je definován na úrovni OS. Definice komunikace je realizována dvěma způsoby. Jednotlivé nody spolu komunikují pomocí heartbeatů, které se realizují buď pomocí tzv. IP zařízení (Crossover[1] ethernet) nebo non-IP zařízení (disková kapacita ze SAN/NAS, RS-232[2] nebo target-mode SCSI[3,4])

Sdílené storage

Disková kapacita se do clusterové topologie implementuje dvěma způsoby. A to buď přímým přiřazením fyzického adaptéru nodu clusteru anebo se realizuje přes VIO server, který spravuje fyzické adaptéry a serverům přiřazuje jen virtuální diskovou kapacitu, kterou mapuje na kapacitu fyzickou.

2.5.4 HACMP subkategorie resources

Jedná se o cílové entity, u kterých chceme koncept vysoké dostupnosti zajistit. Mezi tyto entity patří např. aplikace, VG[5,6], FS[7] apod.

Service IP (Servisní IP aplikace)

Uživatelé se připojují do aplikace na základě síťové architektury client/server. Z tohoto důvodu musí mít aplikace definovanou svoji IP adresu a hostname. Tyto definice musí

[1] Křížené zapojení síťových ethernetových kabelů mezi dvěma počítačovými systémy, z důvodu zajištění přímé komunikace mezi nimi, bez nutnosti vkládat další síťové prvky

[2] Sériová linka, sériová komunikace mezi počítačovými systémy

[3] SCSI mod pro komunikace mezi dvěma počítačovými systémy

[4] *SCSI Target Mode*. [online]. Nedatováno. [cit. 2015-03-27]. Dostupné z SCSI Target Mode: http://www-01.ibm.com/support/knowledgecenter/ssw_aix_61/com.ibm.aix.kernextc/scsi_target.htm

[5] Volume group(s) – logická vrstva LVM (logical volume manager) pro virtuální správu diskové capacity

[6] *Volume group (VG)*. [online]. Nedatováno. [cit. 2015-03-29]. Dostupné z: volume group (VG): http://www.tldp.org/HOWTO/LVM-HOWTO/vg.html

[7] Filesystem

být součástí resource groupy, protože musí být asociovány s uzlem, na kterém aplikace běží. Jedna resource groupa může mít definovánu více než jednu servisní IP adresu. Servisní IP adresa je v tomto případě definována na úrovni HACMP softwaru. Clusterový software sám zajišťuje migraci a následnou rekonfiguraci mezi nody.

VG

Pokud aplikace vyžaduje diskovou storage, musí být sdílena pro všechny nody a aktivní na primárním nodu clusteru. Z tohoto vyplývá, že i případné filesystémy jsou mountovány na primárním nodu clusteru.

Aplikační server

I aplikační server musí být součástí resource groupy a obsahuje startovací a ukončovací skripty.

2.5.5 HACMP subkategorie resource groups

Jedná se o kompilaci subkategorií resources a topology, která je vnímána jako jednotka, kterou dále monitorujeme a provádíme na ní potřebné aktivity. Mezi nejběžnější aktivity nad resource groups patří TAKEOVER, FAILOVER, FALLBACK, SHUTDOWN a STARTUP, které budou popsány v dalších podkapitolách.

2.5.6 HACMP subkategorie customizace

Součástí konceptu vysoké dostupnosti je i možnost upravit na míru sledovací procesy pro jednotlivé RG. Tato nadstavba dovoluje monitorovat jednotlivé aplikace. Obsah monitorovacích činností závisí na vlastnostech konkrétní aplikace.

2.6 HACMP services

Komunikaci mezi jednotlivými nody zajišťuje služba CLSTRMGR[1,2]. Na každém z nodu musí být tato služba aktivována. Tato služba kontroluje dostupnost popř. nedostupnost všech nodu clusteru a v případě detekce nedostupnosti jednoho z nich převezme aktivitu nad RG další nod v clusteru dle nastavených politik.

2.6.1 Dílčí komponenty HACMP services

Služby cluster manageru monitorují jednotlivé nody clusteru. Jednotlivé části cluster manageru se skládají z níže uvedených komponent:

1. topology manager – řídí a spravuje toplogii clusteru
2. resource manager - řídí a spravuje RG.
3. event managera – na základě event. skriptů reaguje na jednotlivé výpadky HW komponent.

Cluster manager obsahuje i SNMP[3] SMUX Peer funciton pro cluster manager MIB[4], která povoluje SNMP[5] monitoring. Součástí cluster managera je i API, které poskytuje

[1] Cluster Manager Daemon – subsystém HACMP, který zajišťuje clustrové služby jako např. monitoring a iniciace přepnutí RG na funkční nod _manager.htm

[2] *Cluster Manager and Clinfo*. [online]. Nedatováno. [cit. 2015-03-21]. Dostupné z Cluster Manager and Clinfo: http://www-01.ibm.com/support/knowledgecenter/SSPHQG_6.1.0/com.ibm.hacmp.progc/ha_clients_cluster

[3] *Case, J., Fedor, M., Schoffstall, M., & Davin, J*. (05 1990). *SNMP*. (MIT Laboratory for Computer Science). [online]. [cit. 2015-03-01]. Dostupné z SNMP: https://www.ietf.org/rfc/rfc1157.txt

[4] Management Information Base

[5] Simple Network Management Protokol

komunikaci mezi cluster managerem a aplikací. A nakonec další utlity, které poskytují status a informaci o stavu clusteru.

2.7 HACMP monitoring

Nejdůležitější funkce monitoringu je kontrolovat stav nodů, sítě a síťových adaptérů. Také kontroluje běh RG a může sledovat i stav aplikací a reagovat na možné chyby dle customizace v nastavení (viz 2.5.6).

Koncept vyšší dostupnosti v sw HACMP reaguje na tři možné nedostupnosti:

- Nedostupnost na úrovni komunikačního adaptéru či zařízení
- Nedostupnost na úrovni nódu (dochází k nedostupnosti všech adaptérů/zařízení na úrovni nodu
- Nedostupnost na úrovni síťového adaptéru (dochází k nedostupnosti všech adaptérů/zařízení na úrovni síťového adaptéru)

HACMP monitor reaguje na případné nedostupnosti tím, že vyhledá funkční komponentu, která by mohla přebrat funkci komponenty, která je nedostupná. To vše na základě nastavených politik. Například pokud je nod nedostupný, HACMP monitor inicializuje akci FAILOVER, která zahrnuje přesunutí RG na funkční nod (viz 2.7.1). V dalších podkapitolách jsou uvedeny další možné politiky, které HACMP monitor, na základě vyhodnocené nedostupnosti, iniciuje.

2.7.1 HACMP status FAILOVER

V případě že HACMP monitor detekuje nedostupnost některé z komponent clusteru, vyhledá náhradní funkční komponentu, která převezme funkci vadné komponenty. Pokud nedostupnost nastane na úrovni nodu, HACMP monitor realizuje akci FAILOVER. V rámci této akce se provede přesun RG ze z havarovaného nodu na

funkční nod. Pokud zhavaruje NIC[1] kompletně, HACMP Services provedou přesun *Servisní IP adresy aplikace* na jiný dostupný NIC v rámci stejného nodu. V případě nedostupnosti všech NIC v rámci stejného nodu se cluster manager pokusí přesunout IP adresu na další nod dle nastavených politik RG. Pokud je nedostupností ovlivněna jen jedna RG, přesouvá se jen dotčená resource groupa, ostatní jsou ponechány v původním stavu.

2.7.2 HACMP status FALLBACK (po opravě závady)

Pokud byla předchozí komponenta opravena, musí být zaintegrována zpět do clusteru. Reintegrace je proces, který učiní komponentu opět dostupnou. Některé komponenty jsou automaticky reintegrovány např. NIC. Ostatní komponenty jako například nody nemůžou být explicitně (automaticky) reintegrovány, dokud administrátor clusteru manuálně nevygeneruje žádost o reintegraci, a to nastartováním HACMP services (viz 2.6) na obnoveném nodu. Další možné manuální reintegrace jsou přesun resource groupy popř. nastavení resource groupy online.

2.7.3 HACMP status FALLBACK ve standby režimu (active/passive)

Ve standby režimu clusteru se aktivně provozuje RG jen na jednom z nodů. Druhý nod je pasivní a je využit v případě nedostupnosti aktivního nodu. V konfiguraci každé resource groupy je nastaveno, který nod bude pro danou aplikaci primární a který nod bude sekundární. V případě nastavení statusu FALLBACK v nastavení RG jsou, po opravě vadné komponenty a opětovné dostupnosti nodu, iniciovány aktivity k navrácení RG do původního stavu běhu na primární nod. Mezi významné nevýhody tohoto kroku je tzv. *down-time*[2,3] aplikace. Pokud totiž HACMP monitor komponentu detekuje jako opravenou a status FALLBACK je nastaven, přesun RG se provede ihned bez ohledu na utilizaci serveru. Takto může být FALLBACK vyvolán v čase největší utilizace a tím

[1] Network Interface Card

[2] Doba po kterou aplikace popř. RG neběží

[3] *www.thesaurus.com*. [online]. Nedatováno. [cit. 2015-03-22]. Dostupné z www.thesaurus.com: http://www.thesaurus.com/browse/downtime

dojde k porušení konceptu nejvyšší dostupnosti. K možnému vyvarování se tohoto nedostaku je nastavení non-FALLBACK policy statusu na RG.

2.7.4 HACMP status non-FALLBACK ve standby režimu (active/passive)

Tento stav HACMP monitor realizuje, pokud není nastavena FALLBACK policy u resource groupy. Použije se v případě, pokud chceme zkrátit dobu, kdy je aplikace nedostupná. Tento stav tedy znamená, že RG se po opravě a zprovoznění komponenty nepřepne zpět na primární nod a zůstane běžet na sekundárním nodu. Tímto se vyhneme druhotnému výpadku, který je potřeba pro přepnutí nodu zpět. Přepnutí pak realizujeme manuálně a to v době nižší utilizace.

2.7.5 HACMP status TAKEOVER v standby režimu (active/passive)

V případě servisního zásahu na jednom z nodu clusteru můžeme předejít k nucenému odstavení serveru tím, že vyvoláme manuální přerušení běhu resource groupy. Tento zásah realizujeme manuálním vyvoláním přepnutí resource groupy. Po servisní události na odstaveném nodu můžeme stejným způsobem a to manuálním zásahem vyvolat přesunutí RG na původní nod.

2.7.6 HACMP status TAKEOVER v režimu mutual (active/active)

V případě paralelního běhu aplikace na více nodech je po havárii spuštěna procedura převzetí běhu aplikace funkčním nodem. V tomto případě je potřeba počítat s větším naddimenzováním jednotlivých uzlů. V případě výpadku musí zbývající nody být schopny aplikaci provozovat. Aplikace musí být naprogramována tak, aby podporovala concurrent mode, raw devices popř. service labels.

Concurrent mode

V případě tohoto režimu všechny nody spouští kopii stejné aplikace a sdílí stejný diskový prostor. VG musí být definovány v tzv režimu concurrent, kdy ve stejnou dobu mohou na stejné místo přistupovat více aplikací.

Service labels

Pokud je aplikace aktivní na více nodech, každý nod má vlastní servisní IP adresu. Klientský systém musí být nakonfigurován tak, aby výběr přístupu na aplikaci byl náhodný, popř. musí být připraven přepnout na další service IP, v případě nedosupnosti nodu. Je žádoucí definovat IP multiplexer mezi clienty a clusterem. Aplikace se pak nemusí starat, na který nod bude přistupovat nebo který nod je nedostupný. Multiplexer na základě nastavených politik sám rozdistribuovává jednotlivé požadavky od klientů mezi jednotlivé uzly clusteru. Multiplexer musí být realizován v duplicitním režimu, aby splňoval koncept vyšší dostupnosti a SPOF.

Raw logical volumes

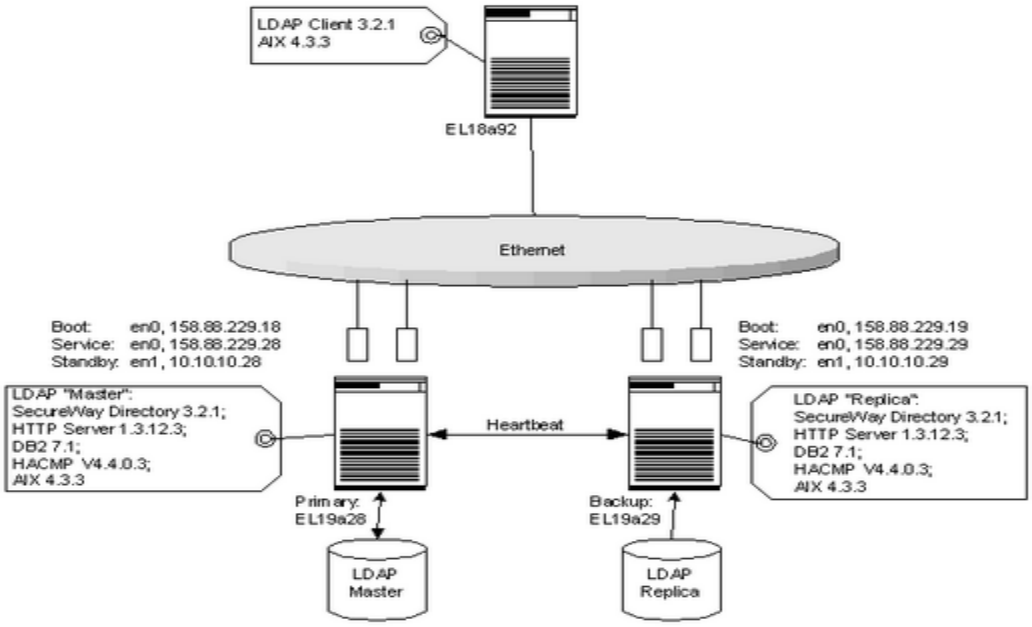
V případě implementace concurrent režimu je vyžadována práce na úrovni raw devices. Není možné pracovat na úrovni filesystémů.

2.8 Příprava clusteru

Při přípravě clusteru se nesmí opomenout nezbytné činnosti, které zahrnují pečlivý návrh clusterové konfigurace, ošetření jednotlivých základních funkcionalit clusteru a jakým způsobem se budou realizovat. Nakonec je potřeba se vyvarovat funkcionalitám, která clusterový software HACMP nepodporuje.

2.8.1 Plánování

Aby bylo možné úspěšně cluster provozovat, je potřeba věnovat více času na plánování, design, konfiguraci a testování clusteru. Pečlivý metodický přístup ke všem fázím životního cyklu clusteru je důležitý faktor k úspěšnosti provozování clusteru. Čas věnovaný všem oblastem přípravy clusteru, snižuje čas, který vyžádá oprava chyb v přípradě špatné konfigurace.



Obrázek 2.1: Ilustrace návrhu clusterové konfigurace[1]

2.8.2 Metodický přístup

Při plánování clusterové infrastruktury je potřeba zvážit všechny požadavky pro přípravu všech relevantních komponent. Při samotné přípravě je potřeba rozpracovat následující body:

- RG musí být provozována nejméně dvěma nody
- Nastavení politik u RG (FAILOVER, FALLBACK apod.)
- RG může být migrovatelná manuálně nebo automaticky pro potřebu rebalancování zátěže

[1] *getdomainvids.com*. [online]. Nedatováno. [cit. 2015-03-16]. Dostupné z HACMP: http://www.getdomainvids.com/keyword/hacmp/

- Nody musí mít nejméně jednu IP network adresu a jednu non-IP network adresu
- Cluster může být složen z libovolné kombinace fyzických či virtuálních nodů
- Cluster může být rozdělen do několika lokalit
- Aplikace může být provozováná skrze monitoring a řízená pomocí uživatelsky definovaných startovacích skriptů.

2.8.3 Nepodporované funkcionality v HACMP clusteru

Zero down-time

HACMP sotware nepodporuje dostupnost 7x24x365[1]. Příležitostně potřebuje být nedostupný pro potřeby údržby. Údržbu nelze provádět online. Tento typ clusteru není tedy vhodný pro životně kritická prostředí.

Security

Cluster management podporuje jen nekryptovaný přístup pro správu clusteru. Sotware využívá systém rhosts files. Nepodporuje TCSEC[2,3] standardy C2 a B1.

2.9 Aplikace

Jednotlivé aplikace jsou provozovány v rámci resource groupy. Při plánování clusterové infrastruktury je potřeba zohlednit možnost automatizace ovládání RG a vyvarovat se vnitřní kolize, které mohou při špatném plánování nastat. Při definici samotné RG je

[1] Zajištění dostupnosti aplikace v režimu 7 dní v týdnu, 24 hodin denně, 365 dní v roce.

[2] Trusted Computer System Evalution Criteria, kritéria pro hodnocení spolehlivosti počítačových systémů

[3] *Handbook of Information Security Management:Policy, Standards, and Organization.* [online]. Nedatováno. [cit. 2015-03-04]. Dostupné z Handbook of Information Security Management:Policy, Standards, and Organization.: https://www.cccure.org/Documents/HISM/390-393.html

potřeba zahrnout všechna potřebná nastavení, aby po aktivaci RG byla aplikace plně funkční.

2.9.1 Požadavky

Automatizace

Jedním z klíčových požadavků na úspěšný provoz aplikace je, že aplikace musí být schopná startovat a zastavovat svůj běh bez manuální intervence. HACMP Services tuto funkcionalitu ovládají, proto není důvod k interakci. V případě nastavené politiky FAILOVER na RG, recovery proces sám zavolá start skript, který zajistí naběhnutí aplikace na pasivním nodu. Tímto je automatická obnova plně zajištěna.

Dependencies

Při navrhování obsahu start/stop skriptů je důležité vyhnout se definicím, které na jiných nodech neexistují. Mezi tyto definice např. patří:

- Odkazy na lokální devices
- Používání hostname, které není stejné na ostatních nodech
- Pevné cesty (/dev/tty0)
- Software licensing[1]

Interference (vnitřní kolize)

V případě nabíhání obou nodů najednou, může dojít ke konfliktu spuštění aplikace na obou nodech. Po čase je tento konflikt vyhodnocen jako stav FAIL[2] a je potřeba ruční zásah administrátora k vyřešení tohoto konfliktu.

[1] Zohlednění softwaru, který je licencován k určitému CPU ID; v případě FAILOVER nemusí správně aplikace nastartovat.

[2] Cluster je v nekonzistentním módu a Cluster manager označí cluster jako nefunkční

2.9.2 Vytvoření resource groupy

Každá resource groupa obsahuje několik základní zdrojů. Před samotným vytvořením nejprve nadefinujeme základní kameny RG. Mezi ně patří definice aplikačního serveru (start a stop skripty aplikace) a servisní IP adresa aplikace pro síťovou komunikaci s klienty. Poté už je možné přistoupit k samotné definici resource groupy, která se skládá z kompilace výše uvedených prvků a dalších údajů jako informace o VG, FS, popř. NFS[1] apod.

2.9.3 Policy resource group

Při provozu RG jsou použity tři stupně politik:

1. STARTUP – používá se při startu služeb clusteru, kdy se zjišťuje, zda už RG neběží na jiném nodu. Pokud RG již neběží na některém nodu clusteru, je přednostně nastartována na primárním nodu clusteru. Teprve pokud se nezdaří RG na primárním nodu nastartovat, pokusí se HACMP monitor vyhledat další nod v seznamu nodu pro RG a pokusí se nastartovat RG na dalším nodu.

2. FAILOVER – pokud nod s aktivní RG zhavaruje, cluster manager vyhledá další funkční nod v seznamu politik resource groupy a RG na něm nastartuje.

3. FALLBACK – v případě odstranění závady, cluster manager určí na základě politik resource groupy, zda se má RG vrátit na původní místo spuštění, a pokud ano, tak provede přepnutí RG.

[1] Network FileSystem

3 Analýza a návrh prostředí

Tato kapitola se zaměří na analýzu a návrh samotného didaktického softwaru. Uvede počáteční kroky, které byly realizovány pro přípravu prostředí aplikace. Také popíše úskalí při výběru programovacího jazyka a programovacího nástroje.

3.1 Výběr programovacího jazyka

Pro výběr programovacího jazyka byly stanoveny tyto kritéria:

1. Podpora OOP[1]
2. Sofistikovaný jazyk s možností využití frameworků
3. Ideálně ve formě API, do kterého by bylo možné realizovat už jen clusterovou část sotwaru.

Po analýze dostupných nástrojů jsem došla k závěru, že nebude možné využít nějaké již stávající API, kde bych mohla svoji clusterovou část programu vybudovat. Z tohoto důvodu jsem musela přistoupit na vytvoření celého API. Bod 1 a 2 splňovalo více programovacích jazyků, ale jako nejvíce vhodný z pohledu další využitelnosti byl nakonec zvolen jazyk Java.

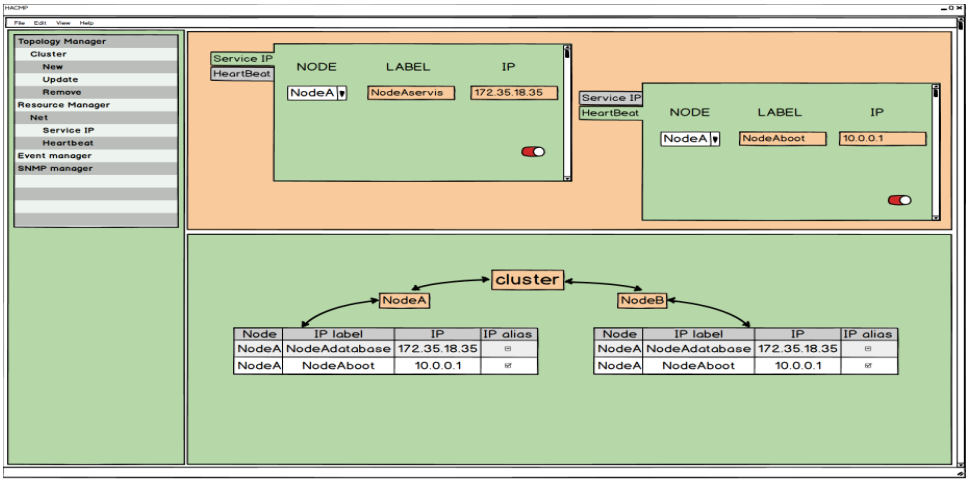
3.2 Návrh prostředí - design

Pro návrh grafické podoby konečného prostředí byl použit program Mockup[2,3]. Tento program umožnil vhodně navrhnout jednotlivé vstupní i výstupní komponenty. Dále umožnil i vytvořit celý design aplikace po estetické stránce.

[1] Objektově Orientované Programování

[2] Sotware Mockup od firmy Balsamiq pomocí kterého byl připraven wireframe(skica webu) aplikace

[3] *Mockup*. [online]. Nedatováno. [cit. 2014-10-02]. Dostupné z Balsamiq: https://balsamiq.com/products/mockups/



Obrázek 3.1: Návrh designu pro vstupní komponentu ServiceIP a Heartbeat v programu Mockup

3.3 Návrh prostředí – technické hledisko

Stanovení kritérií pro výběr správného programovacího nástroje, na začátku vývoje, bylo velmi obtížné. Realizace jednotlivých komponent programu vznikaly postupně a na začátku bylo těžké definovat přesné nároky na programovací nástroj. Toto i bohužel přineslo velké potíže při vlastním vývoji. Pokud by se podařilo stanovit kritéria hned na začátku, nemuselo by dojít k dalším komplikacím. Jako první adepty do konkurzu na programovací nástroj byly zvoleny programy BlueJ a Greenfoot. Jedná se o velmi zajímavá programovací prostředí, ale bohužel vhodné jen pro menší projekty. Poté se vývoj přesunul do programovacího nástroje *Eclipse*[1], kde byla vyvinuta základní struktura programu, ale velmi těžko se mi dařilo realizovat vícenásobnou dědičnost ve frameworku SWING[2,3] pro realizaci GUI. Snaha byla, co nejvíce času ušetřit na vývoji API a spíše se podílet na vývoji samotné logiky (clusterové části) programu. Úskalí

[1] *Eclipse*. [online]. Nedatováno. [cit. 2015-02-21]. Dostupné z Eclipse: https://eclipse.org/

[2] Grafické API, které poskytuje GUI pro Java program

[3] *Using SWING Components*. [online]. Nedatováno. [cit. 2015-03-26]. Dostupné z SWING: http://docs.oracle.com/javase/tutorial/uiswing/components/

frameworku SWING pod programovacím nástrojem Eclipse je nedostatečná designová složka tohoto nástroje. Každý layout panel (viz 4.1) se musel nastavovat a ladit manuálně a to zabíralo velké množství času. Z tohoto důvodu bylo přistoupeno k využití nástroje *Netbeans*[1]. Nástroj *Netbeans* obsahuje grafickou podporu při vytváření GUI[2]. Výstupní panel GUI se navrhne v grafické části nástroje SWING a na základě tohoto návrhu se automaticky vygeneruje kód programu. Tento nástroj byl vyhodnocen jako optimální. Bohužel se ale další životní cyklus vývoje sotwaru nesl ve znamení migrace na nový druh programovacího nástroje a často byly zaznamenány velmi těžce řešitelné chyby s tím spojené. Programování v nástroji *Netbeans* se výrazně odlišuje při použití frameworku SWING. Těžce se hledá zpětná kompatibilita mezi těmito dvěma nástroji.

[1] *Netbeans*. [online]. Nedatováno. [cit. 2015-03-02]. Dostupné z Netbeans: https://netbeans.org/

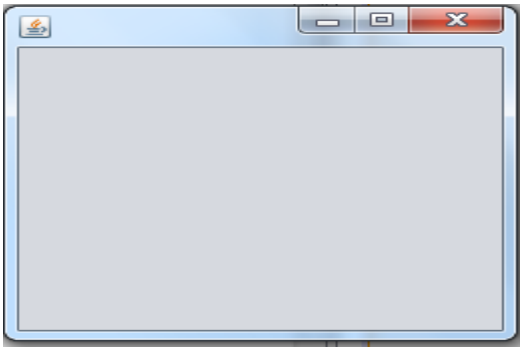
[2] Graphical User Interface (grafické uživatelské rozhraní)

4 Analýza přídavných knihoven

Tato kapitola představí knihovny, které prošly výběrem pro usnadnění samotné realizace programu.

4.1 SWING

Knihovna SWING poskytla při vývoji softwaru potřebné GUI komponenty. Knihovna je založena na architektuře *top-level containeru*[1,2], kdy samotné zobrazení je realizováno na základě více úrovňové hiearchie vrstev. Root této hiearchie byl pro potřeby programu zvolen SWING prvek *JFrame*.

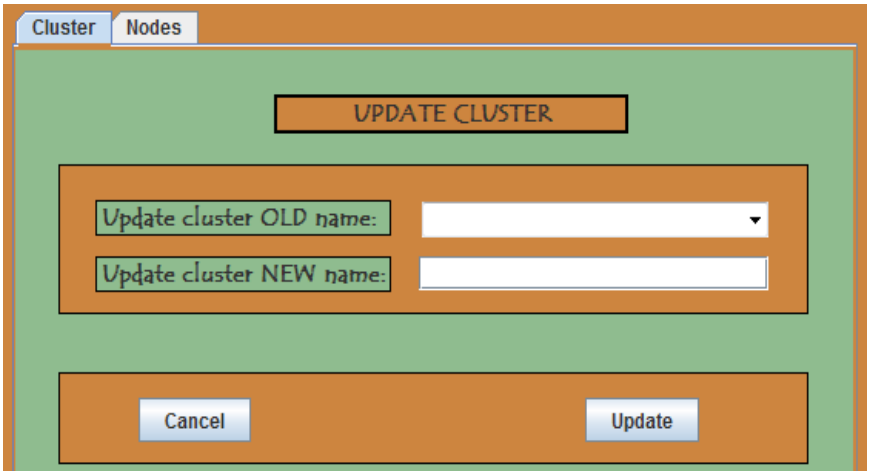


Obrázek 4.1: SWING prvek JFrame

Komponenta *JFrame* obsahuje další vrstvy logických prvků jako MenuBar jehož součástí jsou např. volby New (viz Obrázek 7.1) nebo aktivní volba FAILOVER (viz Obrázek 7.22) apod. Mezi další komponentu v hiearchii SWING konceptu je velmi rozšířený prvek *Content Panel*, který poskytuje GUI pro samotné zobrazení vstupních či výstupních dat. *Content Panel* je v programu realizován třídou *JPanel*.

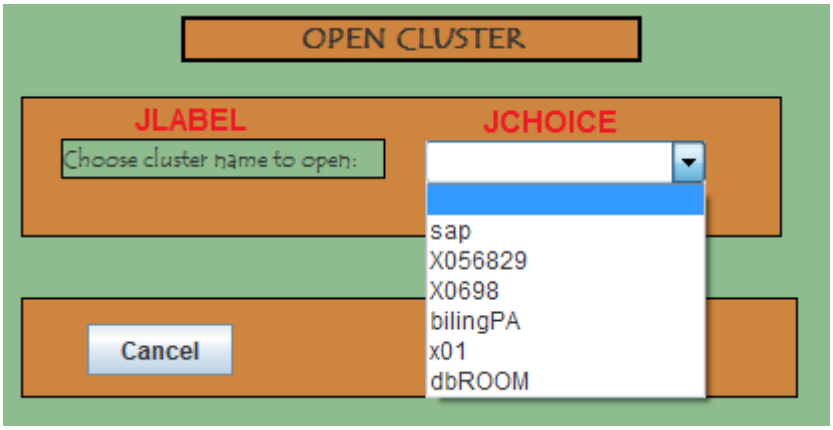
[1] Způsob realizace GUI přes víceúrovňovou hiearchii komponent, kdy jednotlivé prvky jsou vkládány do GUI ve vrstvách

[2] *Top-Level container*. [online]. Nedatováno. [cit. 2015-03-26]. Dostupné z web-feats.com: http://www.web-feats.com/classes/javaprog/lessons/swing_gui_intro/containment.htm



Obrázek 4.2: Content Panel implementován třídou JPanel. V tomto konkrétním případě Content Panel zobrazuje vstupní formulář pro změnu názvu clusteru

V programu byly velmi často využity i prvky hiearchie jako *Layered Panel*, *Glass Panel* aj. Jelikož je logika použití hiearchie komponent knihovny SWING velmi rozsáhla a není námětem této práce, proto odkazuji na další pochopení SWING problematiky na oficiální webové stránky firmy Oracle http://docs.oracle.com/javase/tutorial/uiswing/components/toplevel.html. Pro účely této práce je důležité pochopení pojmu *Content Panel*, který realizuje konkrétní grafické vstupy a výstupy a je realizován třídou *JPanel*, která poskytuje potřebné grafické prvky pro samotnou realizaci zobrazení. Mezi tyto prvky patří např. třídy *JChoice* nebo *JLabe*l aj. (viz Obrázek 4.3)



Obrázek 4.3: SWING prvky JLabel a JChoice

4.2 Derby

Na začátku hledání vhodného způsobu pro správu a archivaci dat byl jednoznačně zamítnut způsob ukládání dat do souboru. Tento způsob nenabízí sofistikované řešení pro vyhledávání a další správu dat. Proto byla jednoznačně zvolena správa dat pomocí databáze.

Po otestování databází Derby a MySQL byla nakonec vybrána databáze Derby, která na rozdíl od databáze MySQL má minimální velikost a je jednoduší na řízení databáze. Databáze Derby je na druhé straně velmi obtížná, přes nástroj Netbeans, na samotnou administraci konkrétně na vytváření struktury databáze, vkládání dat apod.

Další výběr se týkal vhodného typu databázového driveru. Na začátku realizace projektu byl zvolen nešťastný způsob přístupu do databáze přes architekturu client/server. Tento způsob byl postupně vyhodnocen jako nežádoucí. Komplikace nastaly ve fázi spuštění aplikace, která si vyžádala přístup do běžící databáze. Nedalo se předpokládat, že by uživatel provozoval databázi Derby. V tomto bodu bylo tedy jasné, že by databáze musela být na straně uživatele již před spuštěním programu zprovozněna a pro funkčnost by museli být dostupné síťové služby. Proto se jevilo vhodnější využít

Embedded Derby JDBC driver[1,2], který podporuje *stand-alone* databáze bez nutnosti síťového připojení.

4.3 Hibernate

Pro vlastní přístup a správu dat v dabázi byla využita knihovna Hibernate[3,4]. Tato knihovna podporuje takzvané objektově relační zobrazení. Tento princip zajišťuje automatickou konverzi dat mezi relační databází a objektově orientovaným programovacím jazykem. Prakticky celý program provází jedna databáze Cluster, která obsahuje systém tabulek Cluster, Nodes, Heartbeat, Fo a Apl. Na začátku programu dojde k nainstalování databáze Derby a k vytvoření databáze Cluster společně s uvedeným systémem tabulek. V systému pro potřeby programu již byly vytvořeny mapovací třídy pro jednotlivé databázové tabulky. Program sám tedy pracuje na základě těchto tříd a logika Hibernate již obstará požadované přístupy do databáze.

4.4 Chunk

Pro vytvoření konfiguračního souboru, který má za cíl po jeho spuštění na serveru vygenerovat clusterovou infrastrukturu, byla použita knihovna *Chunk*. Princip spočívá ve vytvoření šablony podle určitých klíčů a pravidel a do této šablony se poté dosazují jednotlivá klíčová slova z databáze (viz Obrázek 6.12).

---

[1] Pokud je použit pro přístup do databáze Embedded Derby JDBC driver, tak derby engine není startován jako separátní proces, ale běží uvnitř JVM (Java Virtual Machine). Databáze derby se stane součástí programu

[2] *Apache Derby*. [online]. Nedatováno. [cit. 2015-03-15]. Dostupné z Apache: http://db.apache.org/derby/papers/DerbyTut/embedded_intro.html

[3] Framework umožňující objektově relační mapování (dále jen ORM). Dokáže zachovat stav objektů mezi dvěma aplikacemi a tím udržuje data persistentní. Mapuje Javovské objekty na entity v relační databázi

[4] *Hibernate ORM*. [online]. Nedatováno. [cit. 2015-03-02]. Dostupné z Hibernate: http://hibernate.org/orm/

5 Implementace API

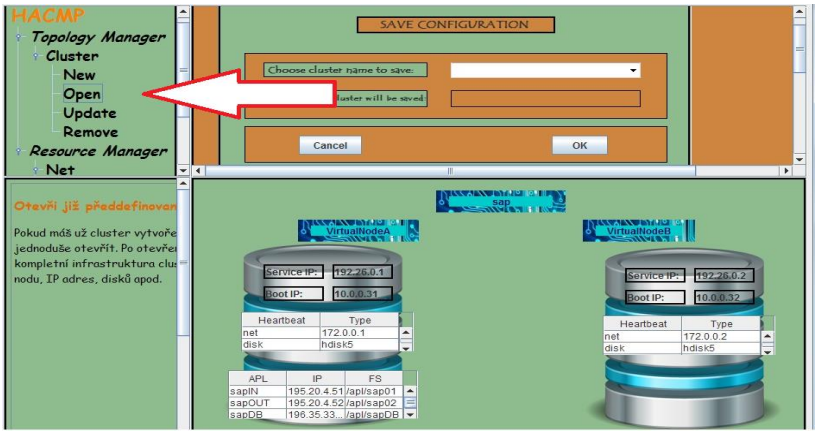
Tato kapitola má za cíl popsat implementaci aplikačního rozhraní, do kterého se vloží samotná logika programu, a to realizace samotné clusterové části programu.

5.1.1 Grafická realizace API

V první fázi bylo potřeba vytvořit základní vrstvu (API) programu, do kterého by bylo možné vkládat vlastní logiku programu. GUI tohoto API bylo vytvořeno na základě top-level hiearchie (viz 4.1). *Content Panel* této hiearchie zobrazuje čtyři implementace třídy JPanel a tím rozděluje grafickou plochu na čtyři na sobě nezávislé statické části, které realizují samotné API (zobrazovací nástroj, který bude následně použit pro implementaci clusterové části programu).

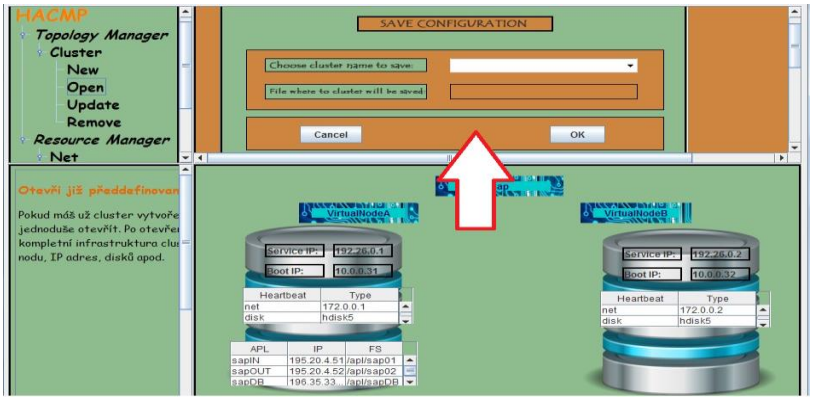
Pro přehlednost byly jednotlivé části pojmenovány takto:

1. *Levý horní vstupní content panel*, který zobrazuje rozšíření třídy JMenu, kde se definuje interakce na jednotlivé volby. Dále obsahuje definice, kde se bude zobrazovat dokumentace.



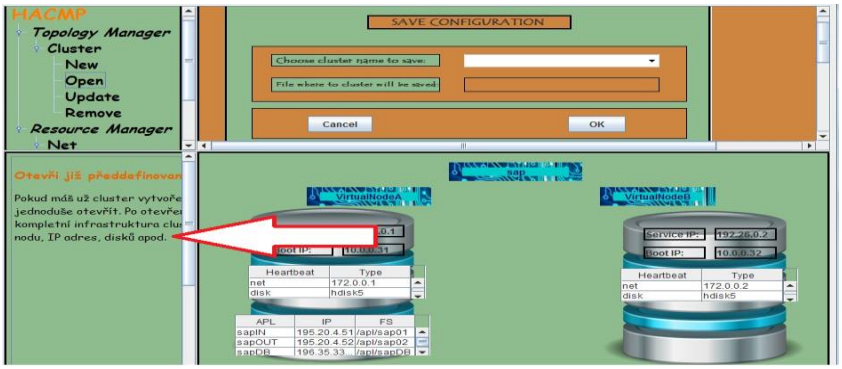
Obrázek 5.1: Levý horní vstupní content panel pro rozbrazení SWING prvku JMenu

2. *Pravý horní vstupní content panel*, který zobrazuje další grafická okna pro vstup základních dat od uživatelů.



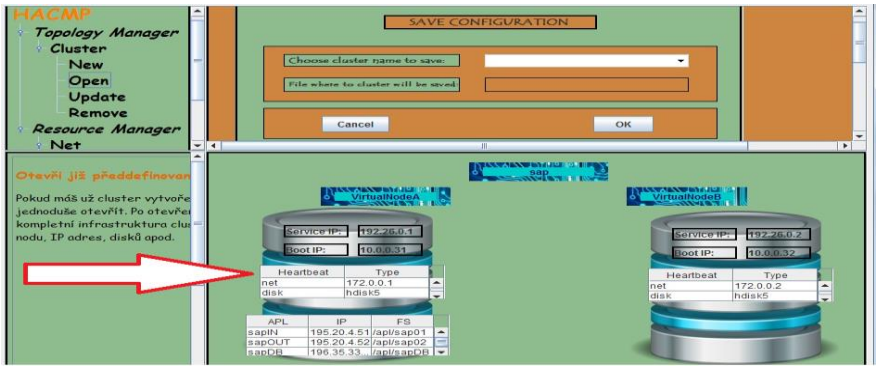
Obrázek 5.2: Pravý horní vstupní content panel

3. *Levý dolní výstupní content panel*, který zobrazuje html průvodce použití softwaru.



Obrázek 5.3: Levý dolní výstupní content panel pro zobrazení html průvodce

4. *Pravý dolní výstupní content panel*, který zobrazuje zpracovávaná výstupní data. Úskalí tohoto panelu byla správná reprezentace grafického objektu přes vícenásobnou dědičnost.



Obrázek 5.4: Pravý dolní výstupní content panel

5.2 Programová realizace API

Základní programové vybavení (API) bylo navrhnuto v programovacím nástroji Eclipse. V tomto nástroji bohužel nefungoval správně design plugin knihovny SWING, proto byla grafická část zrealizována ručně. Pro realizaci tohoto API byly využity třídy Mainy, Desktop, Dispacher

5.2.1 Třída Mainy

Třída Mainy je hlavní třída programu, která spouští samotnou aplikaci. Tato třída je určena i pro vytvoření hlavního prvku JFrame a k nadefinování jednotlivých komponent rámce (frame) JMenu, ikony pro uzavření, minimalizaci popř. maximalizaci rámce.
V případě že uživatel zvolí vytvoření nového souboru, spustí se kompletně nová čtyřvrstvá architektura (viz 5.1.1), která je realizována ve třídě Desktop.

```
if (e.getActionCommand().equals("New")) {
        contentPane.removeAll();
        contentPane.add(desktop);
        contentPane.repaint();
        contentPane.revalidate();
        setContentPane=true;
}
```

Obrázek 5.5: Spuštění nové instance třídy Desktop a přidání do content panelu prvku JFrame

5.2.2 Třída Desktop4

Třída Desktop4 realizuje rozvrstvení zobrazovací části do čtyř statických částí, které jsou realizovány pomocí třídy JPanel. Pro definici jednotlivých tříd JPanel nebyla využita designová část knihovny SWING. Všechny čtyři části se skládají z prvku JScrollPanel [1,2] a do něj včleněného JPanelu. Celá část je usazena do celkového zobrazení pomocí layoutu GridBagLayout[3,4]. V jaké části zobrazení bude panel usazen, rozhoduje poměr parametrů GridX a GridY. Z Obrázek 5.6 je patrné že se bude jednat o *levý horní vstupní content panel*, jelikož má poměr 0:0. Poměr 0:1 usazuje *pravý horní vstupní content panel* apod.

```
JScrollPane scrollPaneRightMenu = new JScrollPane();
GridBagConstraints gbc_scrollPaneRightMenu = new GridBagConstraints();
gbc_scrollPaneRightMenu.fill = GridBagConstraints.BOTH;
gbc_scrollPaneRightMenu.insets = new Insets(0, 0, 0, 0);
gbc_scrollPaneRightMenu.gridx = 0;
gbc_scrollPaneRightMenu.gridy = 0;
add(scrollPaneRightMenu, gbc_scrollPaneRightMenu);
```

Obrázek 5.6: Usazení content panelu do celkového rozvrstvení zobrazovací části

V této třídě se realizuje i samotné vytvoření popř. otevření databáze. Dále vytvoření šablony existující clusterové konfigurace, která se nahraje do databáze. A nakonec i realizaci funkcionalit FALLBACK, TAKEOVER a SAVE.

[1] Prvek JScrollPanel knihovny Swing je použit v případě dynamického obsahu, který přesahuje rámec zobrazovaného okna

[2] *How to use Scroll Panes*. [online]. Nedatováno. [cit. 2015-02-02]. Dostupné z Using Swing components: https://docs.oracle.com/javase/tutorial/uiswing/components/scrollpane.html

[3] Prvek GridBagLayout knihovny Swing jedná se o tzv. vrstvu definovanou uvnitř content panelu a slouží k usazení „viditelných" prvků (např. JButton, JLabel apod.) v rámci zobrazovaného okna

[4] *How to use GridBagLayout*. [online]. Nedatováno. [cit. 2015-03-16]. Dostupné z Laying Out Components Within a Container: https://docs.oracle.com/javase/tutorial/uiswing/layout/gridbag.html

5.2.3 Třída Dispacher

Třída se stará o obecnou funkční část programu a propojuje interakce na jednotlivé podněty. Nejprve inicializuje třídu *Tree*[1], která vytvoří JMenu strukturu s jednotlivými funkčními volbami. Na třídě *Tree* je nastaven listener, který hlídá interakci od uživatele. V případě, že uživatel stiskne tlačítko myši v levém horním vstupním content panelu, listener zavolá funkci *mousePressed*. Součástí této funkce je zjištění pozice myši, z které je možné následně zjistit, kterou volbu si uživatel přál vykonat.

```
tree.addMouseListener(new MouseListener() {

        @Override
        public void mousePressed(final MouseEvent e) {

                DefaultMutableTreeNode node = (DefaultMutableTreeNode)
                tree.getLastSelectedPathComponent();


               if(tree.getSelectionRows().length!=0){
                        final Rectangle r = tree.getRowBounds(tree.getSelectionRows()[0]);
                        if (node == null || !r.contains(e.getX(), e.getY())) {
                                  return;
                        }
               }
```

Obrázek 5.7: Funkce mousePressed zjistí polohu myši po výběru konkrétní volby uživatelem.

Dále spustí funkci *triggerBookmark*, která na základě požadované volby vygeneruje vhodnou html stránku průvodce programem a zobrazí ji v levém dolním výstupním content panelu. Na základě vygenerovaného řetězce se spustí požadovaná aktivita např. vytvoření nové clusterové konfigurace apod. Na závěr se implementuje třída Tree do levého horního vstupního content panelu a tím se zrealizuje vlastní zobrazení prvku JMenu.

5.3 Databázová struktura

Pro uchování clusterové konfigurace byla vytvořena databáze Cluster, která obsahuje jednotlivé tabulky v níže uvedené konfiguraci.

[1] Uživatelsky definovaná třída pro naplnění hodnot SWING JMenu

5.3.1 Table Cluster

Tabulka cluster obsahuje dvě položky a byla vytvořena s níže uvedeným nastavením. Položka Name obsahuje jméno clusteru.

```
CREATE TABLE "Cluster"
(
"IDCluster" INT not null primary key
        GENERATED ALWAYS AS IDENTITY
        (START WITH 1, INCREMENT BY 1),
"Name" VARCHAR(50)
);
```

Obrázek 5.8: SQL příkaz pro vytvoření tabulky Cluster

5.3.2 Table Nodes

Tabulka Nodes obsahuje šest položek a byla vytvořena s níže uvedeným nastavením. Položka NAMECLUSTER slouží k uložení názvu clusteru pro daný nod. NAMENODES obsahuje jméno nodu. Uložení se vykoná pro oba nody. Položka TYPNODE má hodnotu char a tím je určeno, zda se jedná o primární nebo o sekundární nod. V položce SERVICEIP je uložena servisní IP adresa nodu. BOOTIP obsahuje bootovací IP adresu nodu.

```
CREATE TABLE "Nodes"
(
        "IDNODE" INT not null primary key
                GENERATED ALWAYS AS IDENTITY
                (START WITH 1, INCREMENT BY 1),
        "NAMECLUSTER" VARCHAR(50) not null,
        "NAMENODES" VARCHAR(50) not null,
        "TYPNODE" CHAR(1),
        "SERVICEIP" VARCHAR(15),
        "BOOTIP" VARCHAR(15)
);
```

Obrázek 5.9: SQL příkaz pro vytvoření tabulky Nodes

5.3.3 Table Heartbeat

Tabulka Heartbeat obsahuje pět položek a byla vytvořena s níže uvedeným nastavením. Položka NAMECLUSTER slouží k uložení názvu clusteru pro daný nod. NAMENODES obsahuje jméno nodu, pro který se heartbeat bude definovat. Položka TypeHB umožní uložit jen dvě hodnoty. Tyto hodnoty jsou nabídnuty automaticky ve

vstupním content panelu a uživatel vybere vhodnou hodnotu. Hodnotu může nabývat buď non-IP, v tomto případě je použit disk ze SAN anebo hodnotu IP, kdy je vybrána síťová volba heartbeatu, která je konfigurací podporována, ale není preferována.

```
create table "Heartbeat"
(
    "IDHB" int not null primary key
        generated always as identity
        (start with 1, increment by 1),
    "Namecluster" varchar(50),
    "Namenode" varchar(50),
    "TypeHB" varchar(10),
    "ValueHB" varchar(20)
);
```

Obrázek 5.10: SQL příkaz pro vytvoření tabulky Heartbeat

5.3.4 Table Apl

Tabulka Apl obsahuje osm položek a byla vytvořena s níže uvedeným nastavením. Je určená pro kompletní uložení konfigurací provozovaných resource Gross. Položka NAMECLUSTER slouží k uložení názvu clusteru pro daný nod. NamePrimNode obsahuje jméno primární nodu, na kterém resource groupa poběží. V případě havárie primární nodu položka NameSecNode slouží jako informace k přepnutí konfigurace na daný nod. Položka IP obsahuje servisní IP adresu, přes kterou bude uživatelská aplikace komunikovat. Položka LV obsahuje logickou volume, která se použije pro uchování dat pro uživatelskou aplikaci. Položka FS je konkrétní filesystém, kde budou uloženy data aplikace.

```
CREATE TABLE "Apl"
(
    "IDAPL" INT not null primary key
        GENERATED ALWAYS AS IDENTITY
        (START WITH 1, INCREMENT BY 1),
    "NAMEAPL" VARCHAR(50) not null,
    "NAMECLUSTER" VARCHAR(50) not null,
    "NamePrimNode" VARCHAR(50) not null,
    "NameSecNode" VARCHAR(50) not null,
    "IP" VARCHAR(15) not null,
    "LV" VARCHAR(30) not null,
    "FS" VARCHAR(30) not null
);
```

Obrázek 5.11: SQL příkaz pro vytvoření tabulky Apl

5.3.5 Tabulka FO

Tabulka FO je pomocná tabulka, která obsahuje čtyři položky a byla vytvořena s níže uvedeným nastavením. Položka NAMECLUFO slouží k uložení názvu clusteru pro migrovaný nod. NAMENODFO obsahuje jméno nodu, na který byla zmigrovaná aplikace. Tato informace slouží k uchování stavu v případě další migrace, aby bylo možné dohledat aktuální nastavení, na kterém nodu je resource groupa online. Dále je možné z této tabulky zjistit, která clustrová konfigurace je aktuální. Stejná tabulka se použije, jak v případě havárie serveru, tak i v případě přesunu aplikace např. z důvodu údržby nodu.

```
Create table FO
(
    "IDFO" int not null primary key
        GENERATED ALWAYS AS IDENTITY
        (START WITH 1, INCREMENT BY 1),
    "NAMECLUFO" varchar(50),
    "NAMENODFO" varchar(50),
    "TURN" boolean
);
```

Obrázek 5.12: SQL příkaz pro vytvoření tabulky FO

5.3.6 Java persistence API

Pro samotnou správu a řízení chodu databáze je využit programovací interface JPA[1,2] , který na základě nakonfigurovaného modulu persistence ve formátu xml, dokáže propojit všechny již dříve nadefinované frameworky pro práci s databází. Tento interface na základě nadefinovaného JDBC driveru, v našem případě embedded derby

[1] Java Persistence API, Framework umožňující ORM (Objektové Orientované Mapování)

[2] *Java Persistence API*. [online]. Nedatováno. [cit. 2015-02-09]. Dostupné z Java Persitence API: http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html

dirver, nastartuje vhodnou databázi podle tzv. *„table generation strategy“*[1,2] vytvoří popř. smaže celou databázovou strukturu. A na základě zvolené persistentní knihovny, zde je nastavena knihovna hibernate, vytvoří zázemí pro ORM.

V programu jsou použity dva persistentní soubory pro správu a řízení databáze. Soubor dipl2PU2 je použit v případě, že se spouští program poprvé. Na základě schématu drop-and-create se vytvoří databázová struktura, komplex tabulek a uloží se základní šablona pro práci s databází. Níže jsou uvedeny základní charakteristiky z konfigurace tohoto souboru.

```
<persistence-unit name="dipl2PU2" transaction-type="RESOURCE_LOCAL">
<property name="javax.persistence.jdbc.url" value="jdbc:derby:C:\clderby\cluster"/>
<property name="javax.persistence.jdbc.driver…org.apache.derby.jdbc.EmbeddedDriver"/>
<property name="javax.persistence.jdbc.user" value="nb"/>
<property name="javax.persistence.schema-generation.database.action" value=" drop-and-
create"/>
```

Obrázek 5.13: Soubor dipl2PU2.xml – schéma databáze drop and create

Z konfigurace je poznat že byl použit embedded driver, se schématem drop-and-create. Toto schéma znamená, že se po každém spuštění programu se stávající databáze smaže a znovu vytvoří.

Soubor dipl2PU je použit při každém dalším spuštění programu. Tento soubor má nastavené schéma create a používá se pro přístup do jednotlivých tabulek databáze. Níže jsou uvedeny základní charakteristiky z konfigurace tohoto souboru.

```
<persistence-unit name="dipl2PU" transaction-type="RESOURCE_LOCAL">
<property name="javax.persistence.jdbc.url" value="jdbc:derby:C:\clderby\cluster"/>
<property name="javax.persistence.jdbc.driver…org.apache.derby.jdbc.EmbeddedDriver"/>
<property name="javax.persistence.jdbc.user" value="nb"/>
<property name="javax.persistence.schema-generation.database.action" value=" create"/>
```

Obrázek 5.14: Soubor dipl2P2. Xml – schéma databáze create

[1] Strategie JPA, definuje způsob vytvoření databázové struktury

[2] *The Table Strategy*. [online]. Nedatováno. [cit. 2015-03-29]. Dostupné z ObjectDB: http://www.objectdb.com/java/jpa/entity/generated#The_Table_Strategy_

Z konfigurace je poznat, že byl použit embedded driver se schématem create. Toto schéma znamená, že se databáze a tabulky vytvoří jen při prvním spuštění programu.

Pro přístup do databáze se použijí třídy EntityManagerFactory a Entity Manager[1]:

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory("dipl2PU");
EntityManager em = emf.createEntityManager();
```

Obrázek 5.15: Přístup do databáze

Začátek a ukončení transakce se spustí příkazem:

```
em.getTransaction().begin();
em.getTransaction().end();
```

Obrázek 5.16: Začátek a konec transakce

Uzavření databáze jsou realizovány touto sekvencí:

```
em.close();
emf.close();
```

Obrázek 5.17: Uzavření databáze

Vložení nových dat do tabulky jsou realizováno níže uvedenou sekvencí. Nejprve se nastaví jednotlivé položky, poté se potvrdí vklad a provede se commit (potvrzení) transakce.

```
hb.setNamenode(node.getNamenodes());
hb.setNamecluster(node.getNamecluster());
em.persist(hb);
em.getTransaction().commit();
```

Obrázek 5.18: Vložení nových dat do tabulky

Vyhledání položky dle definovaného kritéria je realizováno např.

[1] Třídy EntityManagerFactory a Entity Manager používaná pro práci s JPA

```
Query query = em.createNativeQuery(
                    "Select * FROM Nodes WHERE NameCluster=?",Nodes.class);
query.setParameter(1, name);//.executeUpdate();
List<Nodes> results = query.getResultList();
```

Obrázek 5.19: Vyhledání položek dle zadaného kritéria

Nejprve se do datové struktury query vloží SQL příkaz Select, kterým specifikujeme požadovaný dotaz. Dále příkazem setParameter blíže určíme hodnotu, kterou chceme zrealizovat daný výběr. Výsledek se zapíše do všeobecné struktury List<Object>. Tuto strukturu je možné následně přetypovat a dále s ní pracovat a vybírat konkrétní položky.

Smazání clusterové struktury je realizováno na základě datové struktury QUERY. Do této struktury je uložen SQL příkaz Delete a následně zrealizován pomocí příkazu executeUpdate.

```
Query query = em.createNativeQuery(
      "DELETE FROM CLUSTER");
query.executeUpdate();
```

Obrázek 5.20: Smazání clusterové struktury

Operace UPDATE je realizována pomocí příkazu setParameter, ve kterém se nastaví konkrétní hodnota, kterou požadujeme změnit a příkazem executeUpdate se potvrdí změna v dané tabulce.

```
Query query = em.createNativeQuery(
      "UPDATE Fo SET Namenodfo=?");
query.setParameter(1, node.getNamenodes()).executeUpdate();
```

Obrázek 5.21: Operace UPDATE

6 Implementace clusterové části programu

Tato kapitola obsahuje popis způsobu clusterové části programu v programovacím jazyce Java a způsob její implementace do hlavní programové části aplikace. Samotná realizace je popsána pomocí Java tříd.

6.1 Třída PanelNew

Pro realizaci vytvoření nového clusteru byla zvolena třída PanelNew (viz Příloha A.1), kerá se skládá ze dvou částí. První je design část, která se automaticky vygeneruje programem Netbeans a pomocí pluginu SWING a na základě návrhu provedeného v grafické části pluginu SWING (viz 4.1). Druhá část je vlastní realizační část programu, která v konstruktoru třídy nejprve iniciuje spuštění GUI. Následně se po stisknutí tlačítka Save provede připojení do db, zadané údaje se uloží do databáze Cluster pomocí persistentní funkcionality insert (viz Obrázek 5.18) a následně se uzavře připojení do db. Dále se provede překreslení nově zadaných údajů do pravého dolního výstupního content panelu (viz Obrázek 5.4) pomocí nové instance třídy RepaintingN (viz 6.15).

```
repaintPanel.removeAll();
repaintPanel.add(new RepaintingN(node));
repaintPanel.repaint();
repaintPanel.revalidate();
```

Obrázek 6.1: Překreslení stávajícího zobrazení

Obrázek 6.1 znázorňuje samotné překreslení stávající konfigurace pravého dolního výstupního content panelu, kde pomocí příkazu removeAll bez parametrů zrealizujeme smazání stávajícího zobrazení a poté vytvoříme instanci třídy RepaintingN, která realizuje samotné překreslení a nakonec pomocí příkazu repaint provedeme vykreslení nové konfigurace a na závěr provedeme revalidaci. Tato funkce je uvedena pro případ, že by došlo k porušení zobrazení vlivem neočekávaného zásahu.

Z tohoto důvodu se předá pokyn *Layout managerovi*[1,2], aby vykonal přepočet vrstvy. Přepočtem se spustí požadavek na překreslení.

6.2 Implementace třídy PanelOpen

Implementace třídy PanelOpen (viz Příloha A.2) se skládá ze dvou částí. První je design část, která se automaticky vygeneruje programem Netbeans a pomocí pluginu SWING a na základě návrhu provedeného v grafické části pluginu SWING (viz 4.1). Druhá část je vlastní realizační část programu, která v konstruktoru třídy iniciuje spuštění grafické části programu a poté pomocí funkce initCustom() naplní prvek JChoice (viz Obrázek 4.3) jmény všech clusterových konfigurací, které program vyhledá v tabulce CLUSTER (viz Obrázek 6.2).

```
Query query = em.createNativeQuery(
                    "Select * FROM Cluster",Cluster.class);
List<Cluster> results = query.getResultList();

Iterator itr = results.iterator();

while(itr.hasNext()) {
         Cluster element = (Cluster)itr.next();
         choice1.addItem(element.getName());
}
```

Obrázek 6.2: Výběr všech prvků z tabulky CLUSTER a použití názvů clusteru pro zobrazení všech clusterových konfigurací.

Po stisknutí tlačítka OK se spustí funkce jButton2ActionPerformed, která provede aktivity smazání stávajícího zobrazení aktivní clusterové konfigurace z pravého dolního výstupního content panelu a překreslí ho novou konfigurací (viz Obrázek 6.1). Dále smaže obsah databázové tabulky FO pro uložení nové aktivní clusterové konfigurace a uloží do ní novou aktivní clusterovou konfiguraci, jak byla vybrána v poli

[1] Layout manager implementuje interface LayoutManager. Tento interface určuje velikost a pozici komponent v kontejneru

[2] *Using Layout Manager*. [online]. Nedatováno. [cit. 2015-03-26]. Dostupné z Oracle: https://docs.oracle.com/javase/tutorial/uiswing/layout/using.html

JChoice (viz Obrázek 6.3). A na závěr se zavře vstupní okno PanelOpen v pravém horním vstupním content panelu (viz Obrázek 5.2).

```
db.removeFO();
db.setFO(db.getANode(node.getNamecluster()));
```

Obrázek 6.3: Smazání a nastavení nové aktivní clusterové konfigurace do tabulky FO pro potřeby realizace FAILOVER a TAKEOVER.

6.3 Implementace třídy Panel Update (JTabbledPanel Cluster)

Implementace třídy PanelUpdate (viz Příloha A.4) se skládá ze dvou částí. První je design část, která se automaticky vygeneruje programem Netbeans a pomocí pluginu SWING a na základě návrhu provedeného v grafické části pluginu SWING (viz 4.1). Druhá část je vlastní realizační část programu, která v konstruktoru třídy iniciuje spuštění grafické části programu a poté pomocí funkce initCustom() naplní prvek JChoice (viz Obrázek 6.3) jmény všech clusterových konfigurací, které program vyhledá v tabulce CLUSTER (viz Obrázek 6.2).

Po stisknutí tlačítka UPDATE se spustí funkce jButton2ActionPerformed, která načte žádost o změnu názvu clusteru z položky JTextField a provede SQL dotaz UPDATE v databázi ve všech tabulkách (viz Obrázek 5.21). Dále provede aktivity smazání stávajícího zobrazení aktivní clusterové konfigurace z pravého dolního výstupního content panelu a překreslí ho novou konfigurací. V další části smaže obsah databázové tabulky FO pro uložení nové aktivní clusterové konfigurace a uloží do ní novou aktivní clusterovou konfiguraci, jak byla vybrána v poli JChoice (viz Obrázek 6.3). Na závěr se zavře vstupní okno PanelUpdate v pravém horním vstupním content panelu (viz Obrázek 5.2).

6.4 Implementace třídy Panel Update (JTabbledPanel Nodes)

Implementace třídy PanelUpdate (viz Příloha A.4) se skládá ze dvou částí. První je design část, která se automaticky vygeneruje programem Netbeans a pomocí pluginu

SWING a na základě návrhu provedeného v grafické části pluginu SWING (viz 4.1). Druhá část je vlastní realizační část programu, která v konstruktoru třídy iniciuje spuštění grafické části programu a poté pomocí funkce initCustom() naplní prvek JChoice2 (viz Obrázek 6.3) jmény všech clusterových konfigurací, které program vyhledá v tabulce CLUSTER (viz Obrázek 6.2). Pro prvek JChoice2[1] je nadefinovaný i ActionListener (viz Obrázek 6.4).

```
choice2.addItemListener(new ItemListener() {

  @Override
  public void itemStateChanged(ItemEvent ie) {
```

Obrázek 6.4: ActionListener naslouchá na prvku JChoice2

V případě výběru clusterové konfigurace v poli JChoice2 se automaticky vyplní prvek JChoice3[2] adekvátními nody, které s vybranou clusterovou konfigurací souvisí.

Po stisknutí tlačítka UPDATE se spustí funkce jButton1ActionPerformed, která načte žádost o změnu názvu nodu z položky JTextField a na základě staré a nové hodnoty jména nodu zrealizuje SQL dotaz UPDATE v databázi ve všech tabulkách (viz Obrázek 5.21). Dále provede aktivity smazání stávajícího zobrazení aktivní clusterové konfigurace z pravého dolního vstupního content panelu a překreslí ho novou konfigurací. Dále smaže obsah databázové tabulky FO pro uložení nové aktivní clusterové konfigurace a uloží do ní novou aktivní clusterovou konfiguraci, jak byla vybrána v poli JChoice (viz Obrázek 6.3). Na závěr se provede zavření vstupního okna PanelUpdate v pravém horním vstupním content panelu (vi. Obrázek 5.2).

[1]JChoice2 – prvek knihovny SWING, který je naplněn názvy všech existujících clusterů z databáze CLUSTER

[2] JChoice3 – prvek knihovny SWING, který je naplněn názvy všech nodů, které byly nadefinovány k vybranému clusteru z databáze NODES

6.5 Implementace třídy Panel Remove

Implementace třídy PanelRemove (viz Príloha A.3) se skládá ze dvou částí. První je design část, která se automaticky vygeneruje programem Netbeans a pomocí pluginu SWING a na základě návrhu provedeného v grafické části pluginu SWING (viz 4.1). Druhá část je vlastní realizační část programu, která v konstruktoru třídy iniciuje spuštění grafické části programu a poté pomocí funkce initCustom() naplní prvek JChoice2 jmény všech clusterových konfigurací, které program najde v tabulce CLUSTER (viz Obrázek 6.2).

Po stisknutí tlačítka UPDATE se spustí funkce jButton2ActionPerformed, která z položky JChoice načte žádost o smazání clusteru a na základě této žádosti vygeneruje SQL dotaz DELETE ve všech tabulkách databáze CLUSTER (viz Obrázek 5.20). Dále provede aktivity smazání stávajícího zobrazení aktivní clusterové konfigurace z pravého dolního vstupního content panelu (viz Obrázek 6.1). Na závěr se zavře vstupní okno PanelRemove v pravém horním vstupním content panelu (viz Obrázek 5.2).

6.6 Implementace třídy Panel SIPNew

Implementace třídy PanelSIPNew (viz Příloha A.5) se skládá ze dvou částí. První je design část, která se automaticky vygeneruje programem Netbeans a pomocí pluginu SWING a na základě návrhu provedeného v grafické části pluginu SWING (viz 4.1). Druhá část je vlastní realizační část programu, která v konstruktoru třídy iniciuje spuštění grafické části programu a poté pomocí funkce initCustom() naplní prvek JChoice2 jmény všech clusterových konfigurací, které program vyhledá v tabulce CLUSTER (viz Obrázek 6.2). Pro prvek JChoice2[1] je nadefinovaný i ActionListener.

[1] JChoice2 – prvek knihovny SWING, který je naplněn názvy všech existujících clusterů z tabulky CLUSTER

V případě výběru clusterové konfigurace v poli JChoice2 se automaticky vyplní prvek JChoice1[1] s adekvátními nody, které s vybranou clusterovou konfigurací souvisí.

Po stisknutí tlačítka SAVE se spustí funkce jButton2ActionPerformed, nejprve zrealizuje připojení do db kod, uložení zadaných údajů do tabulky NODES pomocí SQL kodu UPDATE, neboť záznam již v tabulce existuje a následně uzavření připojení do databáze (viz Obrázek 5.21). Dále se provede překreslení nově zadaných údajů do pravého dolního content panelu pomocí nové instance třídy RepaintingN (viz 6.15).

6.7 Implementace třídy Panel BIP

Implementace třídy PanelBIP (viz Příloha A.6) se skládá ze dvou částí. První je design část, která se automaticky vygeneruje programem Netbeans a pomocí pluginu SWING a na základě návrhu provedeného v grafické části pluginu SWING (viz 4.1). Druhá část je vlastní realizační část programu, která v konstruktoru třídy iniciuje spuštění grafické části programu a poté pomocí funkce initCustom() naplní prvek JChoice2 jmény všech clusterových konfigurací, které program vyhledá v tabulce CLUSTER (viz Obrázek 6.2). Pro prvek JChoice2 je nadefinovaný i ActionListener (viz Obrázek 6.4). V případě výběru clusterové konfigurace v poli JChoice2 se automaticky vyplní prvek JChoice1 s adekvátními nody, které s vybranou clusterovou konfigurací souvisí.

Po stisknutí tlačítka SAVE se spustí funkce jButton2ActionPerformed, která nejprve zrealizuje připojení do databáze (viz Obrázek 5.15), uložení zadaných údajů do tabulky NODES pomocí SQL kodu UPDATE, neboť záznam již v tabulce existuje a následně uzavření připojení do databáze (viz 5.3.6). Dále se zrealizuje překreslení nově zadaných údajů do pravého dolního content panelu pomocí nové instance třídy RepaintingN (viz 6.15).

[1] JChoice1 – prvek knihovny SWING, který je naplněn názvy všech nodů, které byly nadefinovány k vybranému clusteru z tabulky NODES

6.8 Implementace třídy PanelHBNew

Implementace třídy PanelHBNew (viz A.7) se skládá ze dvou částí. První je design část, která se automaticky vygeneruje programem Netbeans a pomocí pluginu SWING a na základě návrhu provedeného v grafické části pluginu SWING (viz 4.1). Druhá část je vlastní realizační část programu, která v konstruktoru třídy iniciuje spuštění grafické části programu a poté pomocí funkce initCustom() naplní prvek JChoice1 jmény všech clusterových konfigurací, které program vyhledá v tabulce CLUSTER (viz Obrázek 6.2). Pro prvek JChoice1 je nadefinovaný i ActionListener. V případě výběru clusterové konfigurace v poli JChoice1 se automaticky vyplní prvek JChoice2 adekvátními nody, které s vybranou clusterovou konfigurací souvisí. Obsah prvku JChoice3 je statický a naplní se konstatními hodnotami disk nebo net dle použitého typu heartbeatu (viz 2.5.3).

Po stisknutí tlačítka OK se spustí funkce jButton2ActionPerformed, nejprve zrealizuje připojení do databáze, uložení zadaných údajů do tabulky Heartbeat pomocí SQL kodu UPDATE, neboť záznam již v tabulce existuje a následně uzavře připojení do databáze (viz 5.3.6). Dále se provede překreslení nově zadaných údajů do pravého dolního content panelu pomocí nové instance třídy RepaintingN (viz 6.15).

6.9 Implementace třídy PanelHBTable

Třída PanelHBTable (viz A.8) obsahuje konstruktor třídy, který při vytvoření instance této třídy vykreslí tabulku, jejíž design část se automaticky vygeneruje programem Netbeans a pomocí pluginu SWING a na základě návrhu provedeného v grafické části pluginu SWING (viz 4.1). Po inicializace této třídy se zavolá procedura getQueryHB, která vyhledá všechny položky tabulky HEARTBEAT a vloží je do tabulky (viz Obrázek 6.5). Nakonec se kompletně překreslí celý výsledný obrázek do pravého dolního content panelu pomocí nové instance třídy RepaintingN (viz 6.15).

```
Heartbeat element = (Heartbeat)itr.next();
Object[] row = new Object[2];
row[0] = element.getTypehb();
row[1] = element.getValuehb();

((DefaultTableModel) jTable1.getModel()).insertRow(i,row);
i++;
```

Obrázek 6.5: Vložení jednotlivých položek tabulky Heartbeat do jednotlivých řádků tabulky JTable[1]

V případě, že nebyla ještě volána instance třídy PanelHBNew, a to je v případě, že nebyl zadán a uložen požadavek na nový Heartbeat, tak nebude prvek JTable zobrazen (viz Obrázek 6.6).

```
jScrollPane1.setVisible(false);
jScrollPane1.repaint();
jScrollPane1.revalidate();
```

Obrázek 6.6: Prvek JTable není zobrazen

6.10 Implementace třídy PanelAplNew

Implementace třídy PanelAplNew se skládá ze dvou částí. První je design část, která se automaticky vygeneruje programem Netbeans a pomocí pluginu SWING a na základě návrhu provedeného v grafické části pluginu SWING (viz 4.1). Druhá část je vlastní realizační část programu, která v konstruktoru třídy iniciuje spuštění grafické části programu a poté pomocí funkce initCustom() naplní SWING prvek JChoice1 jmény všech clusterových konfigurací, které program vyhledá v tabulce CLUSTER (viz Obrázek 6.2). Pro prvek JChoice1 je nadefinovaný ActionListener. V případě výběru clusterové konfigurace v poli JChoice1 se automaticky vyplní prvek JChoice2 adekvátními nody, které s vybranou clusterovou konfigurací souvisí. Na prvek JChoice2 je taktéž definovaný Action Listener (viz Obrázek 6.4). Po výběru nodu v prvku JChoice2 se prvek JChoice3 vyplní hodnotou zbývajícího uzlu. Pokud je tedy v prvku JChoice2 vybrán primární nod, prvek JChoice3 bude automaticky vyplněn sekundárním nodem a naopak (viz Obrázek 6.7).

[1] JTable – grafický prvek knihovny SWING

```
while(itr.hasNext()) {
Nodes element = (Nodes)itr.next();
        if (!element.getNamenodes().equals(ie.getItem())) {
                choice3.addItem(element.getNamenodes());
        }
}
```

Obrázek 6.7: Výběr zbývajícího nodu

Po stisknutí tlačítka OK se spustí funkce jButton2ActionPerformed, která nejprve zrealizuje připojení do databáze a uloží zadané údaje do tabulky APL pomocí proměnné třídy Apl namapované do této tabulky. V dalším kroku se provede příkaz persistence, který uloží hodnotu proměnné třídy Apl do tabulky APL (vizObrázek 5.18). Dále se provede překreslení nově zadaných údajů do pravého dolního výstupního content panelu pomocí nové instance třídy RepaintingN (viz 6.15).

6.11 Implementace třídy PanelAplTable

Třída PanelAplTable (viz A.10) obsahuje konstruktor třídy, který při vytvoření instance této třídy vykreslí tabulku, jejíž design část se automaticky vygeneruje programem Netbeans a pomocí pluginu SWING a na základě návrhu provedeného v grafické části pluginu SWING (viz 4.1). Po inicializace této třídy se zavolá procedura getApl, která vyhledá všechny položky z tabulky Apl, dle určeného nodu a vloží je do tabulky JTable (viz Obrázek 6.8) a kompletně překreslí pravý dolní výstupní content panel pomocí nové instance třídy RepaintingN (viz 6.15).

```
while(itr.hasNext()) {
        Apl element = (Apl)itr.next();
        Object[] row = new Object[3];
        row[0] = element.getNameapl();
        row[1] = element.getIp();
        row[2] = element.getFs();

        ((DefaultTableModel) jTable1.getModel()).insertRow(i,row);
        i++;
}
```

Obrázek 6.8: Vložení jednotlivých položek resource groupy do jednotl. řádků tabulky.

Pokud nebyla vytvořena instance třídy PanelAplNew a to je v případě, že nebyl zadán a uložen požadavek na novou resource groupu, tak není prvek JTable zobrazen vůbec (viz Obrázek 6.9).

```
if (apl.isEmpty()) {
       jScrollPane1.setVisible(false);
       failover=true;
}
```

Obrázek 6.9: Prvek JTable není zobrazen

6.12 Realizace stavu TAKEOVER

Pro realizaci této funkcionality byla použita databázová tabulka FO (viz 5.3.5), jejíž hlavní primární aktivita spočívá v uchování informace o momentálním aktuálním stavu resource groupy a v realizaci samotného přepnutí resource groupy. Na základě informace, na kterém nodu je resource groupa aktivní, je možné zrealizovat samotné přepnutí tzn. překreslení resource groupy na další nod.

Změny se do tabulky FO ukládají na základě níže uvedených aktivit.

1) Při práci s programem[1] se iniciuje informace o aktivní clusterové konfiguraci a o aktivním nodu[2] a uloží se do tabulky FO (viz 5.3.5)

2) Při přepnutí resource groupy na jiný nod clusteru se změní záznam o aktivním nodu v tabulce FO

Při následné simulaci přepínání se vyvolá informace o aktuálním aktivním nodu z databázové tabulky FO a na základě toho se přepočte informace o nodu, na který bude přepnutí zrealizováno. Po přepnutí se zobrazí aktuální stav pravého dolního výstupního content panelu, kde se přepíše obsah zobrazovacích dat zohledněných o přepnutí (viz 6.15).

6.13 Realizace stavu FAILOVER

Pro realizaci této funkcionality byla použita databázová tabulka FO (viz 5.3.5), jejíž hlavní primární aktivita spočívá v uchování informace o momentálním aktuálním stavu

[1] vytváření, změna, otevření clusterové konfigurace

[2] nod, na kterém je resource groupa online

resource groupy a v realizaci samotného přepnutí resource groupy v důsledku havárie nodu. Na základě informace, na kterém nodu je resource groupa aktivní, je možné zrealizovat samotné přepnutí tzn překreslení resource groupy na další nod.

Změny se do tabulky FO ukládají na základě níže uvedených aktivit.

3) Při práci s programem se iniciuje informace o aktivní clusterové konfiguraci a o aktivním nodu (rozuměj nodu, na kterém je resource groupa online) a uloží se do tabulky FO (viz 5.3.5)

4) Pro případ havárie nodu a následném vynuceném přepnutí resource groupy se v tabulce FO uchová informace o aktuálním aktivním nodu.

Při následné simulaci havárie se vyvolá informace o aktuálním aktivním nodu z databázové tabulky FO a na základě toho se přepočte informace o nodu, na který se provede přepnutí. Po přepnutí se zobrazí aktuální stav pravého dolního výstupního content panelu (viz Obrázek 5.4), kde se přepíše obsah zobrazovacích dat zohledněných o přepnutí.

6.14 Implementace třídy Panel Save

Implementace třídy PanelSave (viz A.11), která realizuje vlastní logiku vytvoření a uložení vybrané konfigurace do souboru, se skládá ze dvou částí. První je design část, která se automaticky vygeneruje programem Netbeans a pomocí pluginu SWING a na základě návrhu provedeného v grafické části pluginu SWING (viz 4.1). Druhá část je vlastní realizační část programu, která nejprve v konstruktoru třídy iniciuje spuštění grafické části programu a poté pomocí funkce initCustom() naplní prvek JChoice1 jmény všech clusterových konfigurací, které program najde v tabulce CLUSTER (viz Obrázek 6.2). Po vybrání názvu clusteru, se automaticky vygeneruje cesta, kde uživatel konfigurační soubor po uložení najde.

```
choice1.addItemListener(new ItemListener() {

        @Override
        public void itemStateChanged(ItemEvent ie) {
            String item = choice1.getSelectedItem();
            item="C:/clderby/"+item+".odm";
            jLabel2.setText(item);
            jPanel2.repaint();
            jPanel2.revalidate();
        }
    }
```

Obrázek 6.10: Vytvoření cesty k souboru.

Po stisknutí tlačítka OK se spustí funkce jButton2ActionPerformed, která nejprve vytvoří vstupní soubor pro uložení konfigurace (viz Obrázek 6.11).

```
String name="C:/clderby/"+node.getNamecluster()+".odm";
File f = new File(name);
try{
        f.createNewFile();
} catch (IOException e){
        System.out.println("Nelze vytvorit soubor.");
}
```

Obrázek 6.11: Vytvoření souboru

Poté nahraje šablonu ve formátu (viz Obrázek 6.12) a uloží do ní dle daného klíče celou clusterovou konfiguraci.

```
Theme theme = new Theme();
Chunk chunk = theme.makeChunk("template");

chunk.set("cluster", node.getNamecluster());

List<Nodes> listN= db.getNode(node.getNamecluster());
chunk.set("nodeA",listN.get(0).getNamenodes());
chunk.set("nodeB",listN.get(1).getNamenodes());
chunk.set("serviceipA",listN.get(0).getServiceip());
chunk.set("serviceipB",listN.get(1).getServiceip());
chunk.set("bootipA",listN.get(0).getBootip());
chunk.set("bootipB",listN.get(1).getBootip());
```

Obrázek 6.12: Uložení clusteru a nodů do šablony

```
cluster_name={$cluster:null}
node_list="{$nodeA:null},{$nodeB:null}"
/usr/es/sbin/cluster/utilities/cllsclstr -cwng >/dev/null 2>&1
if [ $? -ne 0 ]; then
        node_list="$node_list"
fi
/usr/es/sbin/cluster/utilities/clmodclstr -n $cluster_name -p"$node_list"
/usr/es/sbin/clus
```

Obrázek 6.11: Ukázka šablony

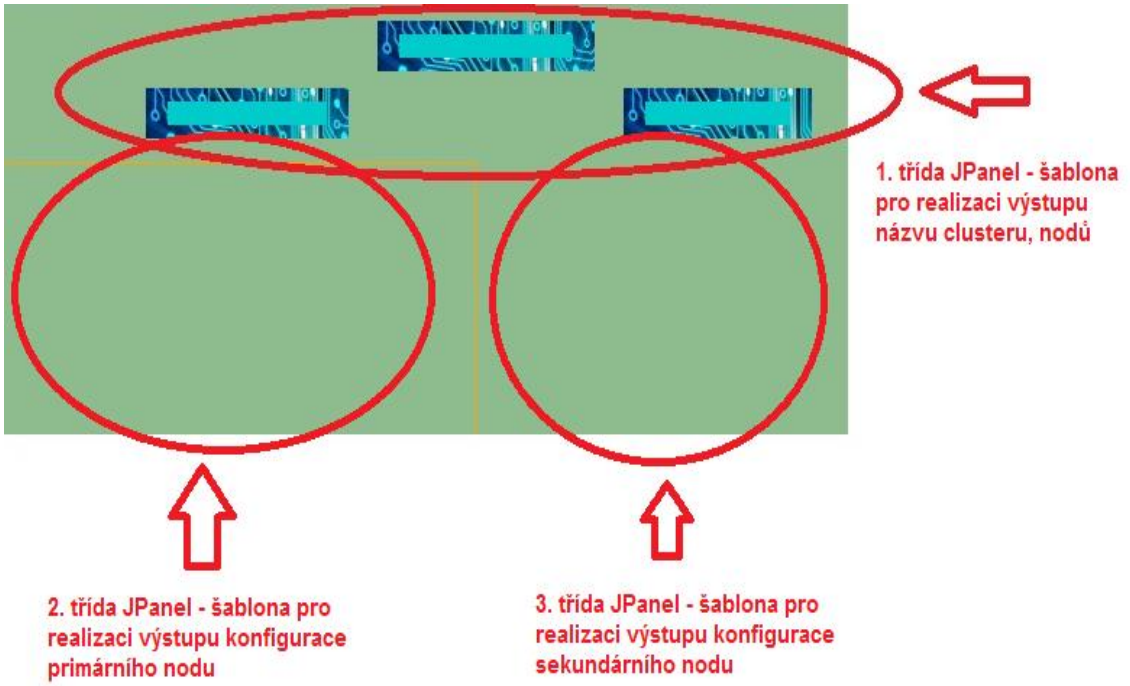
Na závěr se uloží šablona (viz Obrázek 6.13) do souboru.

```
FileWriter out;
try{
        out = new FileWriter(name);
        chunk.render( out );
        out.flush();
        out.close();
} catch (IOException e){
        System.out.println("Nelze zapsat do souboru.");
}
```

Obrázek 6.13: Zápis do souboru

6.15 Implementace pomocné třídy RepaintingN

Pomocná třída RepaintingN (viz A.16) realizuje samotnou grafickou reprezentaci výstupních dat do pravého dolního výstupního content panelu (viz Obrázek 5.4). Tato třída obsahuje funkční implementaci třídy JPanel, která je realizována ve třech částech (viz Obrázek 6.14). První část graficky interpretuje základní výstupní hodnoty, mezi které patří název clusteru a nodů. Druhá část zobrazuje kompletní konfiguraci primárního nodu a třetí část zobrazuje konfiguraci sekundárního nodu.



Obrázek 6.14: Rozdělení pravého dolního výstupního panelu na tři části a způsob grafické interpretace jednotlivých částí pro naplnění odlišných konfigurací implementace clusteru.

Samotná realizace této třídy je rozdělena do dvou částí. První je design část, která se automaticky vygeneruje programem Netbeans a pomocí pluginu SWING a na základě návrhu provedeného v grafické části pluginu SWING (viz 4.1). Druhá část je vlastní realizační část programu, která v konstruktoru třídy iniciuje spuštění grafické části programu a poté pomocí funkce initCustom() nejprve naplní části grafické třídy JPanel vhodnými daty. V první části načte a zobrazí názvy nodů a clusteru. V druhé části iniciuje pozadí primárního a sekundárního nodu a poté do ní realizuje samotné načtení a vykreslení konfigurace. Ve vlastní konfiguraci nodu nejprve načte z databáze Cluster servisní a boot IP adresy nodu, posléze načte konfiguraci heartbeatu a resource groupy a vše překreslí do pravého dolního výstupního content panelu (viz 6.15).

7 Metodika provozu aplikace

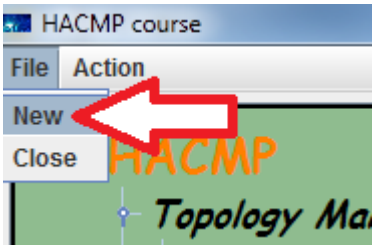
Tato kapitola je metodická příručka pro uživatele. Příručka nejprve navede, kde aplikaci najít a jak ji nainstalovat. Dalšími kroky uživateli ukáže, jak má postupovat s manipulací této aplikace.

7.1 Instalace programu

Vlastní provoz programu je realizován vhodným spuštěním souboru ***dipl.exe*** z adresáře C:/HACMP. V případě nekorektního chování programu lze realizovat i spuštěním příkazu ***{JAVA_HOME}/bin/java -jar C:/HACMP/dipl.jar*** (viz. README.txt na DVD). Pro běh programu je potřeba mít nainstalovaný program Java runtime s minimální verzí 7u67. Instalační soubor javy je součástí instalačního CD. Dále je potřeba mít práva zápisu na svazek C:/. Program vytváří adresář C:/clderby, kde se ukládá databáze a konfigurační soubory. Prvotní spuštění a samotný chod aplikace může mít delší odezvy. Během prvního spuštění aplikace se realizuje instalace databáze včetně všech potřebných ovladačů. Samotný běh aplikace má delší odezvy při přístupu do databáze při spouštění akcí FAILOVER a TAKEOVER.

7.2 Aktivace aplikace

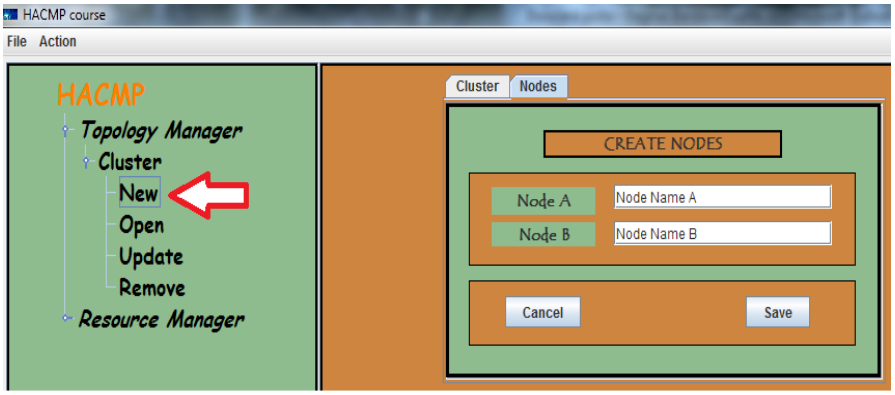
Pro spuštění a vyvolání samotné funkcionality programu je potřeba v menu File zadat volbu New.



Obrázek 7.1: Zahájení provozu aplikace

7.3 Založení nového clusteru

Při vstupu do programu je spuštěn prováděcí manuál v levém dolním výstupním content panelu (viz Obrázek 5.3). Jako první krok je doporučeno vytvořit si nový cluster.
V levém horním vstupním content panelu je potřeba v menu HACMP vybrat volbu Topology Manager / Cluster / New.



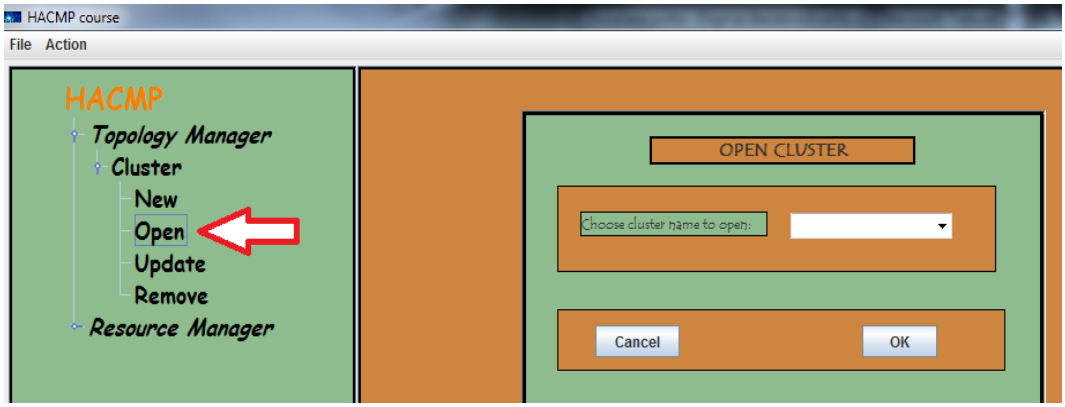
Obrázek 7.2: Vytvoření nového clusteru

Po vybrání této volby je uživatel vyzván k zadání vstupních údajů. Nejprve je požádán, aby v záložce Cluster definoval název clusteru a poté se přesune do záložky Nodes, kde definuje názvy nodů. Jelikož se jedná jen o podpůrný program výuky, byla v konfiguraci nastavena možnost definovat jen dva nody clusteru. Student zadá názvy nových nodů a potvrdí zapsané údaje. Na základě této interakce program vygeneruje v pravém dolním výstupním content panelu image, která zobrazí definovaný cluster (viz Obrázek 7.3).

Obrázek 7.3: Vytvoření nového clusteru se dvěma nody

7.4 Otevření existujícího clusteru

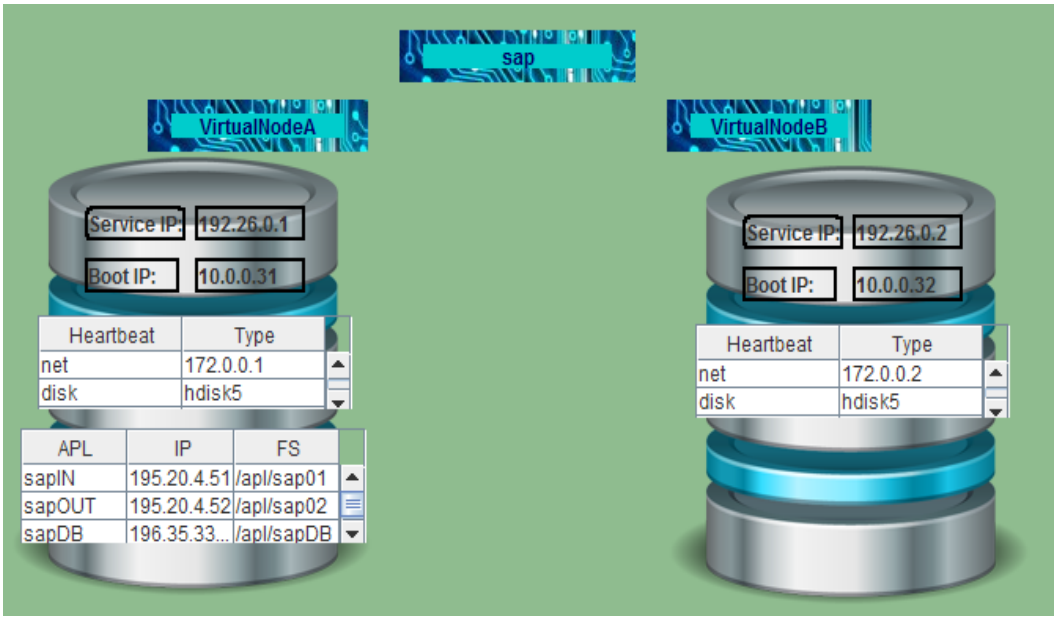
Pro otevření stávající konfigurace je potřeba přejít do levého horního vstupního content panelu (viz Obrázek 5.1), kde je potřeba vybrat v menu HACMP pořadí Topology Manager / Cluster / Open.



Obrázek 7.4: Otevření stávající konfigurace clusteru

Po vybrání této volby je uživatel vyzván k zadání již existující clusterové konfigurace. Pro ulehčení samotného ovládání, program proskenuje celou databázi a nabídne sám již vytvořené konfigurace clusteru. Vybranou existující konfiguraci stačí již jen potvrdit tlačítkem OK.

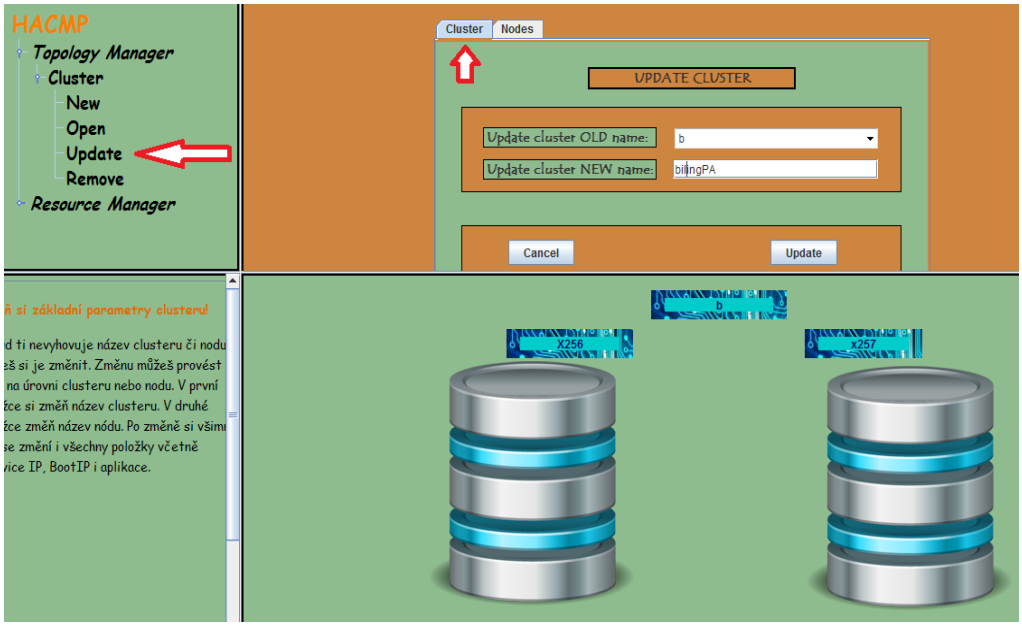
V případě kladného potvrzení se zobrazí v pravém dolním výstupním content panelu kompletní konfigurace již definovaného clusteru (viz Obrázek 7.5).



Obrázek 7.5: Zobrazení kompletní konfigurace clusteru

7.5 Změna názvu clusteru

V případě, že je potřeba provést změnu názvu clusteru je možné toto provést otevřením voleb Topology Manager / Cluster / Update v levém horním vstupním content panelu (viz Obrázek 7.6).

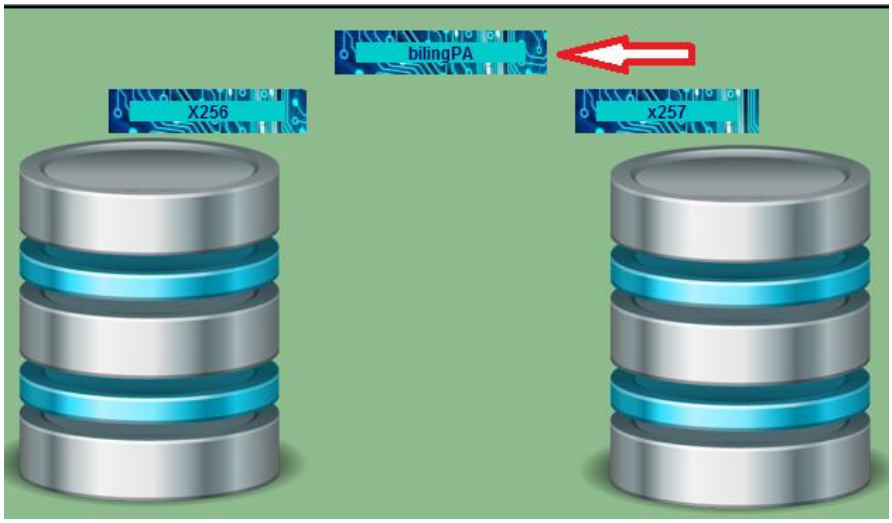


Obrázek 7.6:.Změna názvu clusteru

Po vybrání této volby se v pravém horním vstupním content panelu spustí prvek záložkový panel (JTabbedPane[1]), kde je potřeba provést výběr záložky Cluster. V nabídce Cluster je nejprve potřeba zvolit název clusteru, na kterém se bude změna provádět. Pro realizaci výběru clusteru byl zvolen SWING prvek JChoice, který jako výstup předá výsledek z prohledání databáze na existující konfiguraci clusteru. Po vybrání stávající konfigurace clusteru je potřeba uvést nový název clusteru a potvrdit.

Potvrzením se provedou požadované změny ve všech tabulkách databáze a v pravém dolním výstupním content panelu se zobrazí nová konfigurace clusteru (viz Obrázek 7.7).

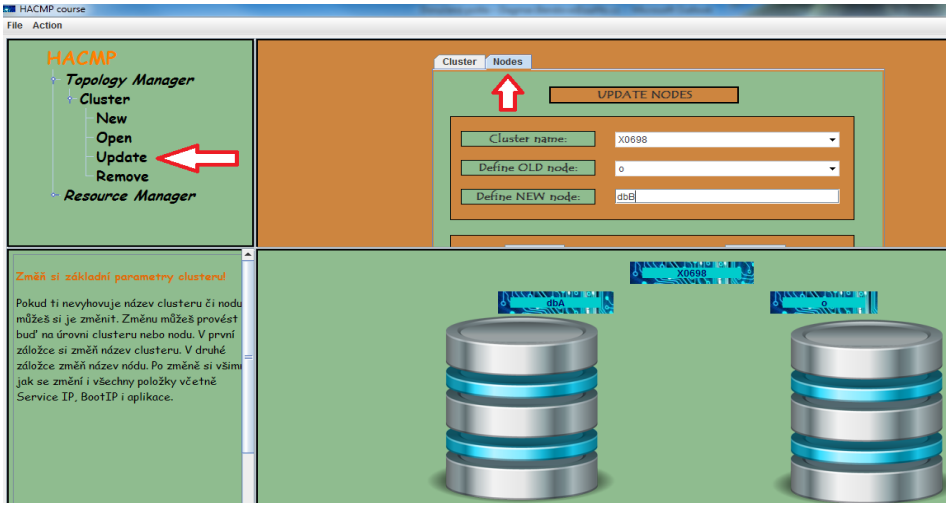
[1] JTabbedPane – prvek knihovny SWING, způsob realizace více záložek v jednom okně.



Obrázek 7.7: Změna názvu clusteru

7.6 Změna názvu nodů

V případě, že je potřeba provést změnu názvu nodu je možné toto provést otevřením voleb Topology Manager / Cluster / Update v levém horním vstupním content panelu (viz Obrázek 5.1).



Obrázek 7.8: Změna názvu nodu(vstup)

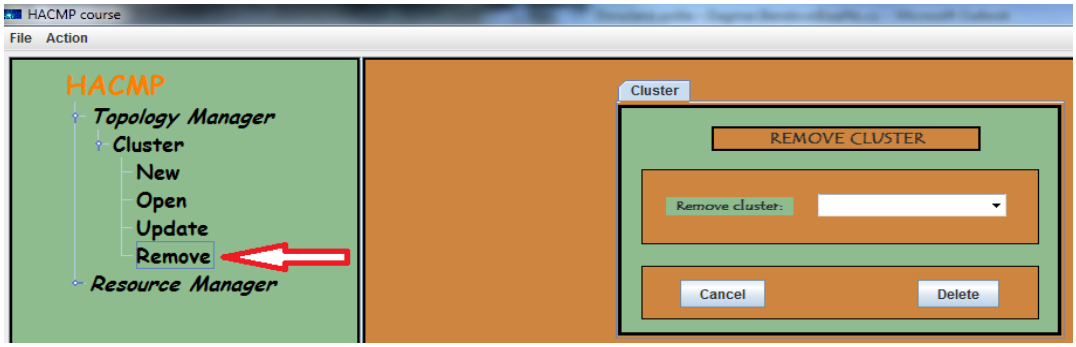
Po vybrání této volby se v pravém horním vstupním content panelu spustí prvek záložkový panel (JTabbedPane), kde je potřeba provést výběr záložky Nodes. V nabídce Nodes je nejprve vhodné zvolit na jakém clusteru se bude změna provádět. Pro realizaci výběru clusteru byl zvolen prvek JChoice knihovny SWING, který jako výstup předá výsledek z prohledání databáze na existující konfigurace clusteru. Po vybrání stávající konfigurace clusteru se automaticky objeví možné dostupné uzly tohoto clusteru v dalším SWING prvku JChoice, který nabídne studentovi možnost výběru nodu, jehož název míní měnit. (viz 6.4). Pak už zbývá doplnit nový název nódu a potvrdit.

Potvrzením se provedou požadované změny ve všech tabulkách databáze a zobrazí se v pravém dolním výstupním content panelu nová konfigurace clusteru (viz Obrázek 7.9).

Obrázek 7.9: Změna názvu nodu (výstup)

7.7 Smazání clusteru

Kompletní smazání již nepotřebné konfigurace clusteru je možné provést řetězcem voleb Topology Manager / Cluster / Remove v levém horním vstupním content panelu (viz Obrázek 7.10).



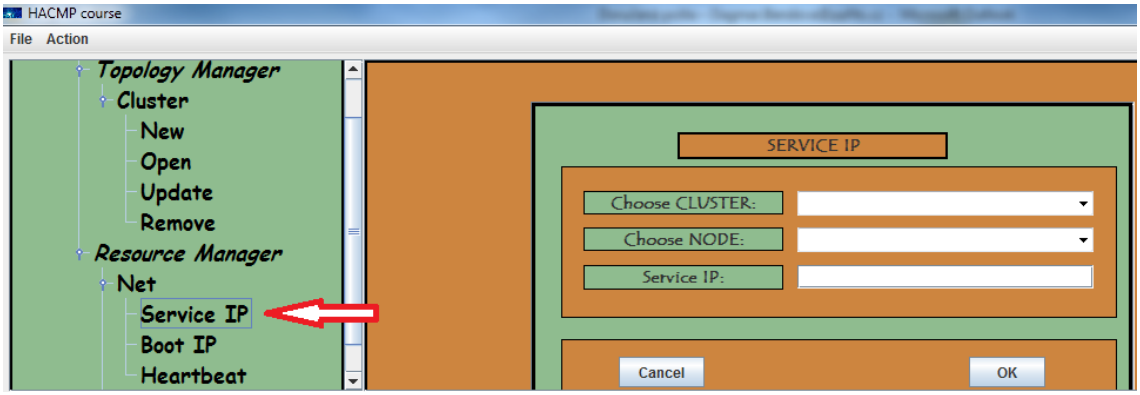
Obrázek 7.10: Smazání clustrové konfigurace

Po vybrání této volby se v pravém horním vstupním content panelu spustí záložkový panel Cluster, kde je potřeba zadat název vstupní konfigurace clusteru, kterou chceme smazat. Pro ulehčení program proskenuje celou databázi a nabídne sám již vytvořené konfigurace. Po vybrání stačí již potvrdit jen tlačítkem OK (viz 6.5).

V případě kladného potvrzení se z databáze derby vymaže kompletní konfigurace zadaného clusteru.

7.8 Vložení Service IP nodu

Vložení servisní IP adresy nodu provedeme volbou HACMP / Resource Manager / Net / Service IP (viz Obrázek 7.11).



Obrázek 7.11: Vložení Service IP (vstup)

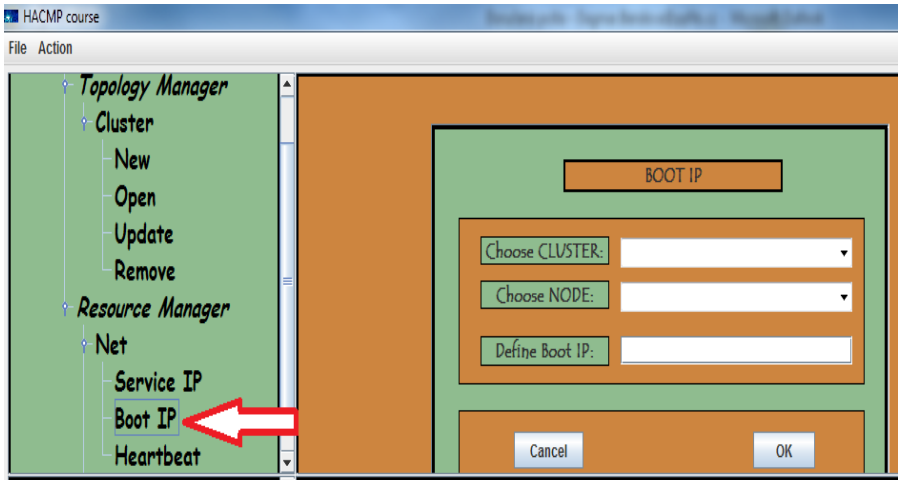
Po vybrání této volby se v pravém horním vstupním content panelu spustí JPanel, kde je možné zadat novou service IP pro daný nod. V nabídce cluster je potřeba nejprve zvolit pro jaký cluster se bude service IP zadávat. Pro realizaci výběru clusteru byl zvolen SWING prvek JChoice, který jako výstup předá výsledek z prohledání databáze na existující konfigurace clusteru. Po vybrání stávající konfigurace clusteru se automaticky objeví možné dostupné uzly tohoto clusteru v dalším SWING prvku JChoice, který nabídne studentovi možnost výběru nodu, jehož název míní měnit. Pak už zbývá doplnit samotnou IP adresu a potvrdit (viz 6.6).

Po potvrzení se překreslí celá konfigurace clusteru s doplněním o nový prvek Service IP (viz Obrázek 7.12).

Obrázek 7.12: Vložení Service IP (výstup)

7.9 Vložení Boot IP

Vložení Boot IP adresy nodu (viz 2.5.3), provedeme volbou HACMP / Resource Manager / Net / Boot IP (viz Obrázek 7.13).



Obrázek 7.13: Vložení Boot IP

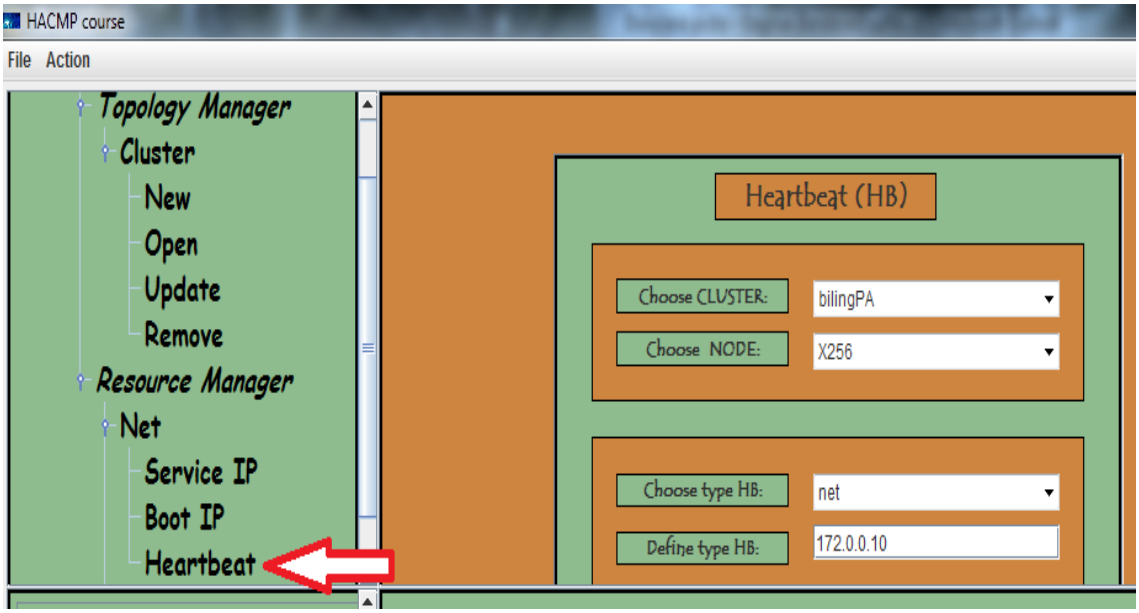
Po vybrání této volby se v pravém horním vstupním content panelu spustí implementace třídy JPanel, kde je možné zadat novou bootovací IP adresu pro daný nod. V nabídce cluster je potřeba nejprve zvolit pro jaký cluster se bude bootovací IP adresa zadávat. Pro realizaci výběru clusteru byl zvolen prvek JChoice knihovny SWING, který jako výstup předá výsledek z prohledání databáze na existující konfigurace clusteru. Po vybrání stávající konfigurace clusteru se automaticky objeví možné dostupné uzly tohoto clusteru v dalším SWING prvku JChoice, který nabídne studentovi možnost výběru nodu, pro který hodlá novou bootovací IP adresu zadat. Pak už zbývá doplnit samotnout bootovací IP adresu a potvrdit (viz 6.7).

Po potvrzení se překreslí celá konfigurace clusteru s doplněním o novou bootovací IP adresu (viz Obrázek 7.14).

Obrázek 7.14: Výstup z vložení bootovací IP adresy do clusterové konfigurace

7.10 Heartbeat

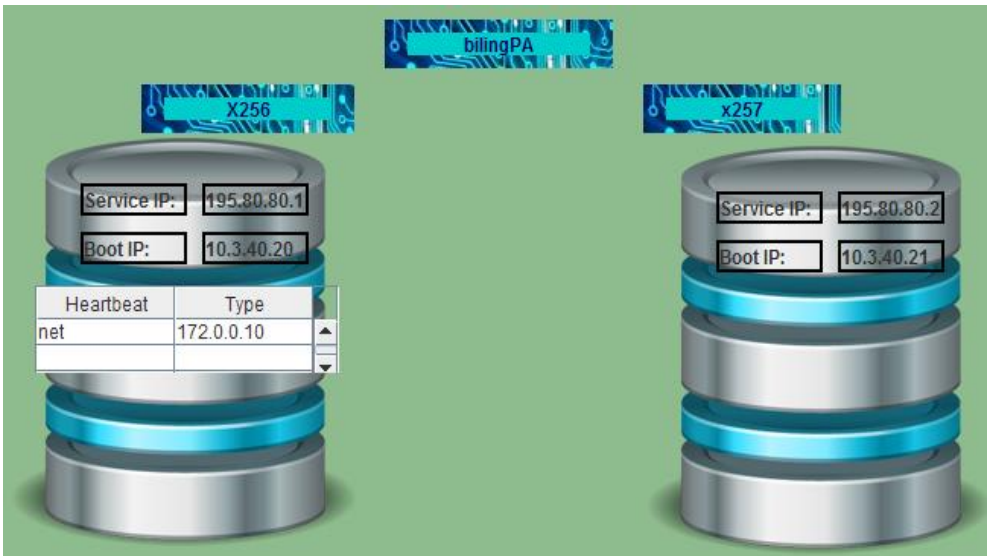
Vložení Heartbeatu pro konkrétní uzel, provedeme volbou HACMP / Resource Manager / Net / Heartbeat (viz Obrázek 7.15).



Obrázek 7.15: Vložení Heartbeatu

Po vybrání této volby se v pravém horním vstupním content panelu spustí JPanel, kde je možné zadat nový Heartbeat pro vybraný nod. V nabídce cluster je potřeba nejprve zvolit clusterovou konfiguraci. Pro realizaci výběru clusteru byl zvolen SWING prvek JChoice, který jako výstup předá výsledek z prohledání databáze na existující konfigurace clusteru. Po vybrání stávající konfigurace clusteru se automaticky objeví možné dostupné uzly tohoto clusteru v dalším SWING prvku JChoice, který nabídne studentovi možnost výběru nodu, pro který se bude Heartbeat definovat. Dalším SWING prvkem JChoice se vybere typ Heartbeatu. Program nabídne studentovi možnost výběru dvou typů a to typ Disk a typ Net. Poté už zbývá jen dodefinovat správnou hodnotu vybraného typu heartbeatu a potvrdit (viz 6.8).

Po potvrzení se překreslí celá konfigurace clusteru s doplněním o nový prvek Heartbeat.



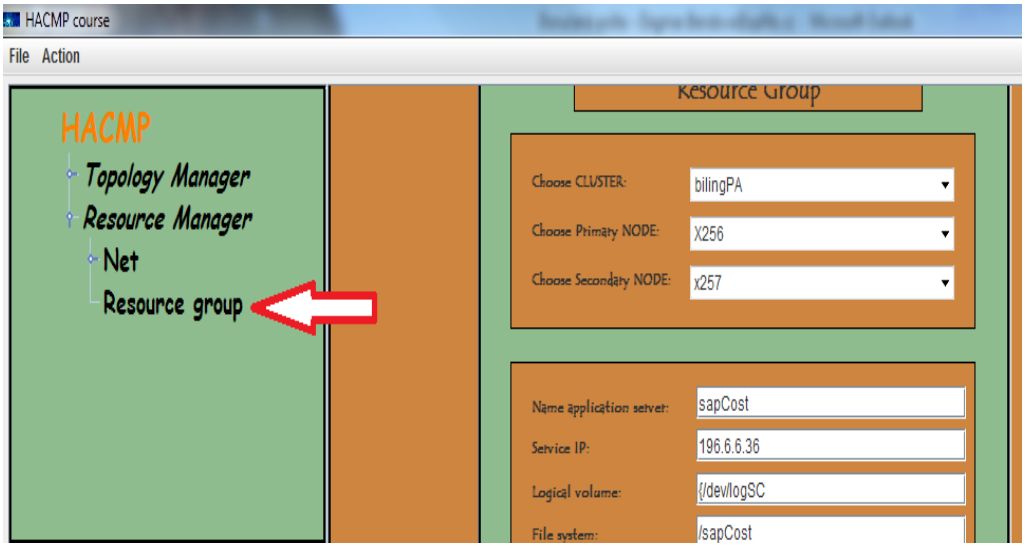
Obrázek 7.16: Výstup z vložení Heartbeatu

Aby byla clustrová konfigurace kompletní nesmíme zapomenout definovat Heartbeat pro oba nody. Z Obrázek 7.16 je patrné, že uživatel zadal pro primární nod clusteru x256 jen jeden heartbeat typu net. Tato konfigurace je nepřístupná. Chybí totiž definice heartbeatu pro sekundární nod x257. Tato konfigurace v reálném nastavení

clusteru nebude akceptována a neproběhne verifikace clusteru, a tudíž nebude možné cluster provozovat.

7.11 Vytvoření nové resource groupy

Vložení nové resource groupy pro konkrétní uzel provedeme volbou HACMP / Resource Manager / Net / Resource group (viz Obrázek 7.17).



Obrázek 7.17: Vytvoření nové resource groupy

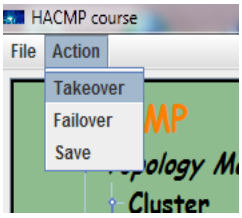
Po vybrání této volby se v pravém horním vstupním content panelu spustí JPanel, kde je možné zadat novou Resource groupu. V nabídce cluster je potřeba nejprve zvolit clusterovou konfiguraci. Pro realizaci výběru clusteru byl zvolen SWING prvek JChoice, který jako výstup předá výsledek z prohledání databáze na existující konfigurace clusteru. Po vybrání stávající konfigurace clusteru se automaticky objeví možné dostupné uzly tohoto clusteru v dalším SWING prvku JChoice, kde budeme definovat primární nod clusteru na kterém aplikace poběží. Po vybrání primárního nodu program automaticky nabídne vyplnit sekundární nod clusteru. Dále je potřeba vyplnit název aplikačního serveru a servisní IP adresy resource groupy. Pokud bude resource groupa využívat i filesystémovou strukturu, je potřeba vyplnit i LV a FS. Poté už zbývá jen nadefinovanou konfiguraci potvrdit.

Po potvrzení se překreslí celá konfigurace clusteru s doplněním o nový prvek Resource groupy, která se vydefinuje ze zadaných vstupních údajů (viz Obrázek 7.18).

Obrázek 7.18: Přehled vytvořené resource groupy

7.12 Akce TAKEOVER (migrace) resource groupy

Vizualizaci migrace (přepnutí) resource groupy lze realizovat spuštěním volby Action / Takeover v hlavním menu aplikace.



Obrázek 7.19: Spuštění migrace resource groupy

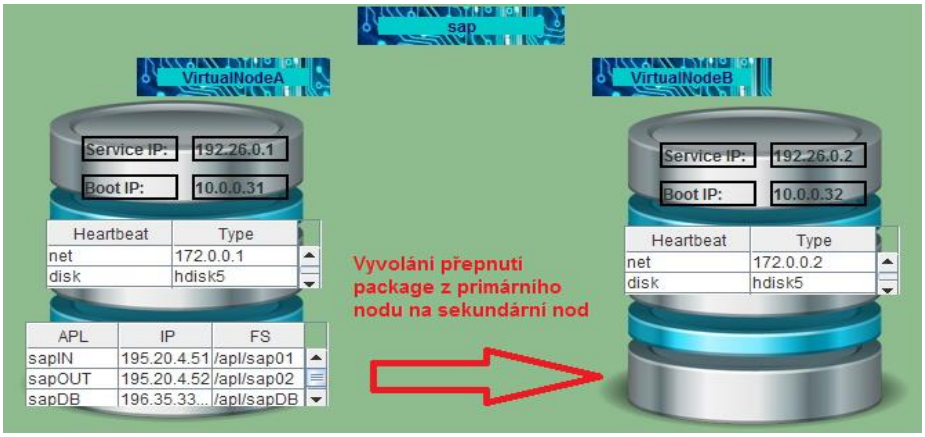
Před spuštěním migrace je nutné otevřít funkční clusterovou konfiguraci. Pro správný průběh vizualizace je potřeba zajistit, aby clusterová konfigurace obsahovala i definici resource groupy. Jinak není možné přepínání resource groupy předvést.

Pokud je clusterová konfigurace nastavená správně, dojde ke grafické simulaci přepnutí resource groupy z primárního nodu na sekundární nod. Na modelu můžeme studentovi předvést, jak je resource groupa odejmuta primárnímu nodu, a to smazáním tabulky s obsahem resource groupy z primárního nodu a jeho překreslením na sekundární nód. Tuto funkcionalitu je i doporučeno vyzkoušet v případě předchozí provedené akce FAILOVER. Tímto může vyučující studentům předvést zpětné navrácení resource groupy na primární nod. Stav výpadku nodu v clusteru se v tomto programu simuluje pomocí funkci FAILOVER. Po realizaci funkce FAILOVER je pak možné pomocí funkcí TAKEOVER vrátit resource groupu do původního stavu. Simulace TAKEOVER pak představuje buď migraci ruční (viz 2.7.5) nebo automatickou (viz 2.7.3). Tuto aktivitu lze studentům předvést, pro demonstraci této základní funkcionality clusteru, opakovaně.

Scénář simulace přepnutí resource groupy na sekundární nód na základě stavu TAKEOVER

Simulace přepnutí resource groupy (ruční vynucení přepnutí) je rozdělena do dílčích kroků, které jsou popsány níže:

1. Resource groupa běží na primárním nodu. Uživatel spuštěním sekvence Action/Takeover vyvolá přepnutí resource groupy. V reálném prostředí Cluster manager ukončí činnost resource groupy na primárním nodu. Provede ukončení aplikace pomocí stopovacího skriptu aplikace, odmountuje FS, deaktivuje volume groupu a připraví RG na její aktivaci na sekundárním nodu clusteru (viz Obrázek 7.20).



Obrázek 7.20: Vyvolání akce TAKEOVER. Simulace ukončení resource groupy na primárním nodu[1]

2. Resource groupa již neběží na primárním nodu. Její běh byl převzat sekundárním nodem. V reálném prostředí Cluster manager nastartuje resource groupu na sekundárním nodu. Provede aktivaci síťových služeb, volume groupy,

[1] Package – jiné označení pro resource groupu

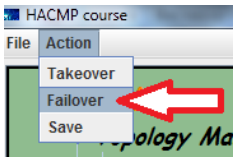
namountuje FS a nastartuje aplikaci pomocí start skritptu. Po těchto krocích nastaví resource groupu jako online (viz Obrázek 7.21).

Obrázek 7.21: Činnost resource groupy je ukončena a běží na sekundárním nodu

Z Obrázek 7.21 je patrné, že se skutečně jednalo o ruční přepnutí resource groupy, neboť na primárním nodu zůstaly aktivní heartbeaty a síťové interfaces.

7.13 Akce FAILOVER

Vizualizaci migrace resource groupy, z důvodu havárie nodu, lze spustit spuštěním volby Action / Failover v hlavním menu aplikace.



Obrázek 7.22: Spuštění simulace havárie uzlu

Před spuštěním migrace je nutné otevřít funkční clusterovou konfiguraci. Pro správný průběh vizualizace je potřeba zajistit, aby clusterová konfigurace obsahovala i definici resource groupy. Jinak není možné předvést simulaci havárie nodu. Přepínání lze ladně kombinovat i s operací TAKEOVER Takto je možné studentovi předvést, co se stane s clusterem v případě havárie jednoho z nodů clusteru. Opakovaným použitím

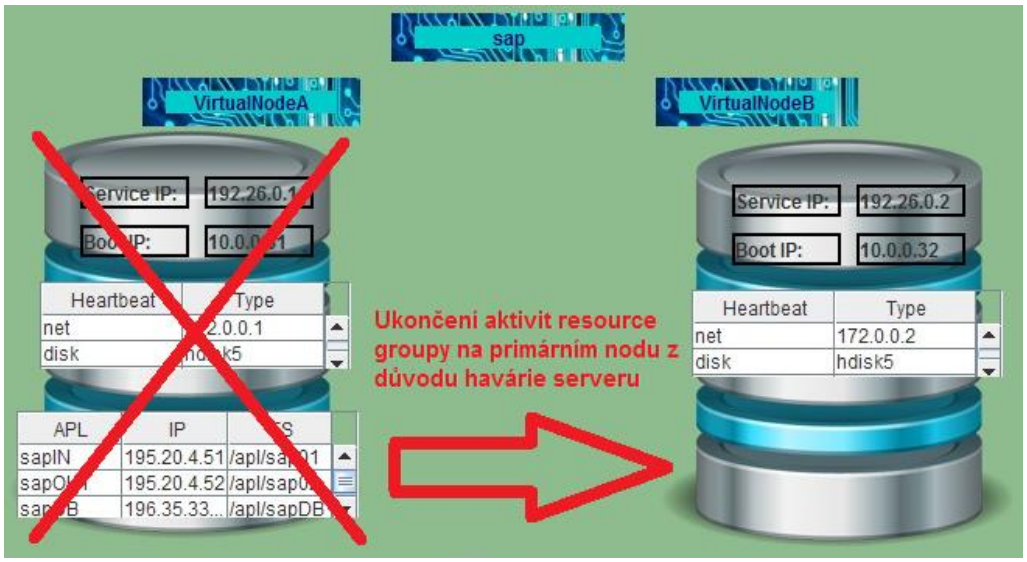
si student zažije hlavní funkcionalitu clusteru. Je možné použít i scénář demostrace zásahu administrátora v případě výpadku nodu. Výpadek nodu student simuluje pomocí funkce FAILOVER. Po opravě nodu student spustí sekvenci Action\Takeover (viz 7.12), čímž simuluje důležitou funkci FALLBACK, kdy na základě nastavených politik si opravený nod přebírá běh resource groupy (viz 2.7.3).

Pokud je clusterová konfigurace nastavená správně, tak po vyvolání simulace havárie aktivního nodu, na kterém je RG online, dojde k označní vadného nodu a k přepnutí resource groupy na funkční nod (v konfiguraci uvedeného jako sekundární nod). Po dokončení oprav lze provést simulaci přepnutí resource groupy zpět na primární nod nebo ponechat resource groupu běžet na sekundárním nodu dle předpokládaných politik dané resource groupy. Ruční přepnutí do konzistentního modu se provede volbou Action/Takeover

Scénář simulace přepnutí resource groupy na sekundární nód na zákl. stavu FAILOVER

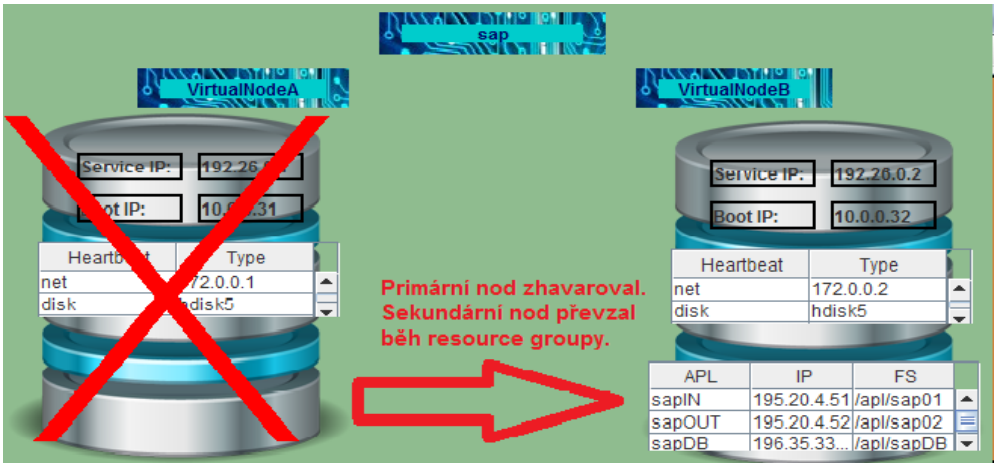
Simulace přepnutí resource groupy v důsledku havárie nodu je rozdělena do dílčích kroků, které jsou popsány níže:

1. Resource groupa běží na primárním nodu. Uživatel spuštěním sekvence Action/Failover vyvolá simulaci havárie aktivního nodu (nod s běžící resource groupou). V reálném prostředí Cluster manager detekuje aktivní nod a vyhledá další nod uvedený v seznamu nodu dané resource groupy (viz Obrázek 7.23).



Obrázek 7.23: Simulace detekce havárie nodu clusteru a vyhledání dostupného nodu, kam dojde k přesunutí resource groupy.

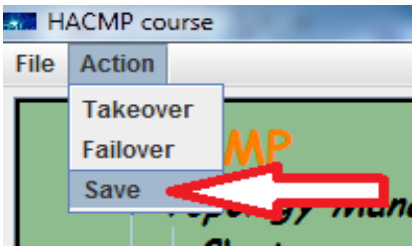
2. V dalším kroku resource groupa neběží na primárním nodu, náhradní nod pro běh resource groupy je vyhledán a resource groupa je na něm aktivována.
   V reálném prostředí Cluster manager aktivuje další nod v řadě seznamu nodů dané resource groupy a resource groupu na něm nastartuje. Provede aktivaci síťových služeb a volume groupy, namountuje FS a nastartuje aplikační server pomocí start skriptu (viz Obrázek 7.24).



Obrázek 7.24: Primární nod je neaktivní z důvodu havárie a čeká na opravu. Sekundární nod je aktivní a realizuje provoz zmigrované resource groupy.

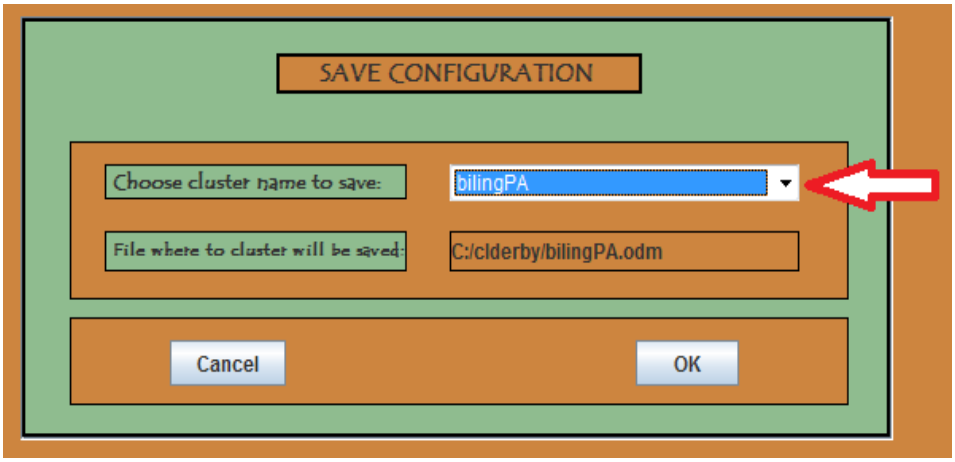
7.14 Uložení konfigurace

Vybranou konfiguraci clusteru je možné uložit spuštěním volby Action / Save v hlavním menu aplikace.



Obrázek 7.25: Uložení konfigurace clusteru

Po spuštění volby SAVE je potřeba zadat existující clusterovou konfiguraci, která je pro pohodlý uživatele automaticky vyhledána a nabídnuta (viz Obrázek 7.26).



Obrázek 7.26: Výběr konfigurace clusteru pro akci Save

Po vybrání této volby se v pravém horním vstupním content panelu spustí JPanel, kde je možné vybrat existující clusterovou konfiguraci. Pro realizaci výběru clusteru byl zvolen SWING prvek JChoice, který jako výstup předá výsledek z prohledání databáze na existující konfigurace clusteru. Po vybrání clusterové konfigurace se automaticky nabídne lokace uložení této clusterové konfigurace. Po potvrzení se vytvoří v adresáři C:\clderby soubor, jehož jméno bude stejné jako jméno clusteru s příponou odm. Soubor bude obsahovat, seznam příkazů spustitelných v bashi OS AIX. Po spuštění tohoto souboru na serveru se softwarem HACMP se naimplementuje funkční clusterová konfigurace (viz 6.14).

8 Realizace projektu

8.1 z hlediska časové náročnosti.

Pro měření času a orientace v dané problematice byly využity zjednodušené principy metody SCRUM[1,2] z předmětu Agilní metodiky programování. Snahou bylo jednotlivé dílčí úkoly rozčlenit do jednotlivých tasků a ty řešit a zmapovat jak dlouho řešení trvalo. Díky rozčlenění projektu do jednotlivých tasků probíhal vývoj aplikace rychleji a na konci bylo možné snadněji zesumarizovat prováděné činnosti. Bylo možné i odhadnou celkovou pracnost projektu, která která byla vyčíslena na 160 hodin.

[1] SCRUM – techniky pomocné vývojovým teamům zkrátit dobu tvorby projektu, vývoj rozdělen do dílčích úkolů (tasků)

[2] *Home - Scrum Alliance.* [online]. Nedatováno. [cit. 2015-03-14]. Dostupné z Home - Scrum Alliance: https://www.scrumalliance.org/

Scrum tasks - Leden

DEN	TERMÍN SPLNĚNÍ	CO	HODIN	PROBÍHÁ	HOTOVO
		Návrh designu	3		
		Výběr Software	1,2,5		
		Základní vize, rozbor	0		
		Startup	2		
		Wysiwyg, Window Builder	0		
		wix	1	NO GOOD	END
		Swing designer – instalace, požívání	2		
		Tvary, posouvání objektů, dynamické vyplňování	4+4		
		Na popředí velká ikona resource group	1		
		Po vyplnění se ikona zmenší	3		
		Knihovna graphics	1		
		Animation in java	2		
		Java 3D, own widget	2		
		Startup - Vytvoření základní obrazovky	2		
		Přidat ikonu server	3		
		Uchopit objekt a přesunout ho	4		
14.1.		mockup	4		
15.1		mockup	1		
15.1		STS – JFRAME, JPanel	4		
16.1		JMenu - technology	6		
19.1.		JPanel – multipanel	4		
20.1.		JPanel - multipanel	4		
20.1.		JPanel – resize multi window	3		
20.1		JTree - RightMenuPanel	1		
21.1.		Cluster new image	2		

Obrázek 8.1: Příklad Scrum tasks včetně hodin vyčleněných na jeden dílčí úkol za měsíc leden.

8.2 Z pedagogického hlediska

Software je navržen tak, aby vhodně doprovázel pedagogický výklad. Jednotlivé části se zobrazují postupně dle interaktivního vstupu a na pokyn vyučujícího. Školení na HACMP cluster trvá v průměru 5 MD[1]. Každý den je možné postupně doplňovat výklad ukázkou z tohoto podpůrného softwaru. Na závěr kurzu student získá ucelený obraz clusterové infrastruktury, kterou si může naimplementovat a použít popř. dále rozvíjet na provozním serveru. Tento program je možné i včelenit do výuky na střední škole.

[1] man-days

Bez nutnosti znát širší problematiku clusteru si může student na základě tohoto programu představit, jaký je princip clusteru a jeho funkcionality.

9 Test aplikace

Software byl otestován a kladně zhodnocen ve školním prostředí na střední škole[1] v rámci mé pedagogické praxe. Tato pedagogická pomůcka přispěla k rychlému pochopení dané problematiky a pomohla aktivně studenty zapojit do výuky. Dále byla pomůcka laděna a anonymně otestována s projektovým managerem a se zástupci datového centra[2], kteří mají na starosti úsek sítě, SAN a administraci AIX. Jejich případné připomínky jsem zapracovala a výsledný produkt byl již použit při rozhodování o zařazení provozované aplikace do úrovně provozu CRITICAL[3].

[1] SPŠ SE Dukelská, České Budějovice

[2] Název datového centra není uveden, protože v době publikace této práce nebylo vydáno povolení, aby bylo možné publikovat konkrétní osoby popř. názvy oddělení, názvy datového centra apod.

[3] Způsob označení provozu aplikací, Critical – kritická aplikace pro zajištění provozu aplikace v modu 24x7

10 Návrhy pro budoucí rozvoj aplikace

V této kapitole budou nastíněny slabiny aplikace a možné návrhy, jak dále rozvíjet tuto aplikaci.

V první fázi bylo vytvořeno funkční programové rozhraní, které vzniklo účelově pro implementaci clusterové problematiky. Během vývoje vyplynul velký potenciál využít toto API pro realizaci dalších didaktických pomůcek. Bylo by velmi žádoucí doladit toto prostředí do formy knihovny, kde by jako hlavní parametry byly předávány odkazy na jednotlivé části GUI (viz 5.1.1).

Další žádoucí změna by mohla přijít v API samotném a to předěláním celého grafického prostředí. Celé GUI by mohlo být přetvořeno z pohledu vzhledu a funkčnosti. Statické ovládání celého GUI by mohlo být realizováno dynamickou formou. Rozměry jednotlivých content panelů (viz 5.1.1) by poté bylo možné dynamicky měnit.

Další návrhy budoucího rozvoje patří úrovni clusterové logiky aplikace. Další verze aplikace by mohla být věnována optimalizaci výkonu celé aplikace a hlavně na prověření databázové části aplikace. Konkrétně zda neexistují optimálnější realizace přístupů do databáze. Momentálně je připojení pomalé a je žádoucí toto připojení zefektivnit.

Další návrh by se mohl věnovat analýze způsobu předávání parametrů mezi třídami aplikace přes vícenásobnou dědičnost, kdy momentální realizace předávání parametrů v programu je nepřehledné. Z pohledu autorky se ale bohužel jedná o vlastnost programovacího jazyka Java a zatím nenašla vhodnější způsob implementace této funkcionality.

11 Závěr

Tento projekt hodnotím z mého pohledu jako velmi přínosný. Podařilo se mi uskutečnit všechny cíle, které jsem si při plánování tohoto projektu stanovila, a dokonce se zdařilo zrealizovat cíl, který nebyl na začátku projektu stanoven a vyplynul z vývoje samotného softwaru.

Ve výsledném produktu bylo zrealizováno vytvoření funkční implementace clusterové konfigurace v architektuře standby (active/passive). Budoucí uživatel má k dispozici vhodnou didaktickou pomůcku pro znázornění zásadních funkčností clusteru. Podařilo se graficky znázornit jednotlivé funkce clusteru jako TAKEOVER a FAILOVER. Dále program dokáže přetransformovat existující clusterovou konfiguraci do spustitelného souboru využitelnou pro implementaci do provozovaného clusterového softwaru HACMP. V rámci vývoje aplikace bylo také neočekávaně vytvořeno programové rozhraní, které je dále využitelné pro implementaci dalších didaktických pomůcek použitelných jak na úrovni školního prostředí, tak i na úrovni firemního prostředí.

Literatura

[1] *Apache Derby*. [online]. Nedatováno. [cit. 2015-03-15]. Dostupné z Apache: http://db.apache.org/derby/papers/DerbyTut/embedded_intro.html

[2] *Cluster Manager and Clinfo*. [online]. Nedatováno. [cit. 2015-03-21]. Dostupné z Cluster Manager and Clinfo: http://www-01.ibm.com/support/knowledgecenter/SSPHQG_6.1.0/com.ibm.hacmp.progc/ha_clients_cluster_manager.htm

[3] *Eclipse*. [online]. Nedatováno. [cit. 2015-02-21]. Dostupné z Eclipse: https://eclipse.org/

[4] *EntityManagerFactory* (Java EE 6). [online]. Nedatováno. [cit. 2015-03-05]. Dostupné z:http://docs.oracle.com/javaee/6/api/javax/persistence/EntityManagerFactory.htm

[5] HA Storage Cluster | Open-E. [online]. Nedatováno. [cit. 2015-03-11]. Dostupné z: http://www.open-e.com/solutions/ha-storage-cluster/

[6] *HACMP*. [online]. Nedatováno. [cit. 2015-03-02]. Dostupné z TheFreeDictionary.com: http://acronyms.thefreedictionary.com/HACMP

[7] *HACMP*. *TheFreeDictionary.com*. [online]. Nedatováno. [cit. 2015-03-03]. Dostupné z: http://acronyms.thefreedictionary.com/HACMP

[8] *Heartbeat*. [online]. Nedatováno. [cit. 2015-03-08]. Dostupné z IBM: http://www-01.ibm.com/support/knowledgecenter/SSPHQG_7.1.0/com.ibm.powerha.admngd/ha_admin_config_hacmp_heartbeat.htm

[9] *Hibernate ORM*. [online]. Nedatováno. [cit. 2015-03-02]. Dostupné z Hibernate: http://hibernate.org/orm/

[10] *High-availability cluster*. [online]. Nedatováno. [cit. 2015-03-09]. Dostupné z: http://en.wikipedia.org/wiki/High-availability_cluster

[11] *High-Performance Computing* (HPC). [online]. Nedatováno. [cit. 2015-03-12]. Dostupné z: http://www.dell.com/learn/us/en/555/high-performance-computing

[12] *How to use GridBagLayout*. [online]. Nedatováno. [cit. 2015-03-16]. Dostupné z Laying Out Components Within a Container: https://docs.oracle.com/javase/tutorial/uiswing/layout/gridbag.html

[13] *How to use Scroll Panes*. [online]. Nedatováno. [cit. 2015-02-02]. Dostupné z Using Swing components: https://docs.oracle.com/javase/tutorial/uiswing/components/scrollpane.html

[14] *IBM Knowledge Center*. [online]. Nedatováno. [cit. 2015-02-08]. Dostupné z IBM Knowledge Center: http://www-01.ibm.com/support/knowledgecenter/SGVKBA_3.1.4/com.ibm.rsct314.aixcmds/idrsct_aixcmds_kickoff.htm

[15] *Java Persistence API*. [online]. Nedatováno. [cit. 2015-02-09]. Dostupné z Java Persitence API: http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html

[16] Load balancing. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-04-06]. Dostupné z: http://en.wikipedia.org/wiki/Load_balancing_%28computing%29

[17] *Mockup*. [online]. Nedatováno. [cit. 2014-10-02]. Dostupné z Balsamiq: https://balsamiq.com/products/mockups/

[18] *Netbeans*. [online]. Nedatováno. [cit. 2015-03-02]. Dostupné z Netbeans: https://Netbeans.org/

[19] Počítačový cluster. *Wikipedie* [online]. Nedatováno. [cit. 2015-04-05]. Dostupné z: https://cs.wikipedia.org/wiki/Po%C4%8D%C3%ADta%C4%8Dov%C3%BD_cluster

[20] *Rouse, M.* (Nedatováno). What is high-performance computing (HPC)?. WhatIs.com. [online]. [cit. 2015-03-21]. Dostupné z: http://searchenterpriselinux.techtarget.com/definition/high-performance-computing

[21] Swing (Java™ Foundation Classes). [online]. Nedatováno. [cit. 2015-03-28]. Dostupné z: http://docs.oracle.com/javase/7/docs/technotes/guides/swing/

[22] *The Table Strategy*. [online]. Nedatováno. [cit. 2015-03-29]. Dostupné z ObjectDB: http://www.objectdb.com/java/jpa/entity/generated#The_Table_Strategy_

[23] *Top-Level container*. [online]. Nedatováno. [cit. 2015-03-26]. Dostupné z web-feats.com: http://www.web-feats.com/classes/javaprog/lessons/swing_gui_intro/containment.htm

[24] *Using Layout Manager*. [online]. Nedatováno. [cit. 2015-03-26]. Dostupné z Oracle: https://docs.oracle.com/javase/tutorial/uiswing/layout/using.html

[25] *Using SWING Components*. [online]. Nedatováno. [cit. 2015-03-26]. Dostupné z SWING: http://docs.oracle.com/javase/tutorial/uiswing/components/

[26] Výpočetní cluster. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-04-06]. Dostupné z:http://cs.wikipedia.org/wiki/V%C3%BDpo%C4%8Detn%C3%AD_cluster

Seznam obrázků

REJSTŘÍK

Příloha A

Programová realizace jednotlivých tříd aplikace

Tato příloha obsahuje výpis zdrojového kodu všech tříd aplikace, které byly použity a vysvětleny v textu této práce.

A.1 Třída PanelNew

```
package dipl2;
import db.Cluster;
import db.Nodes;
import java.awt.Event;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.sql.Connection;
import java.sql.DriverManager;
import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.Vector;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.Query;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class PanelNew extends javax.swing.JPanel {

    int x,y;
    JPanel repaintPanel;
    Object o;
    List<JTextField> list = new Vector<JTextField>();

    /**
     * Creates new form Panel1
     */
    public PanelNew(JPanel tempPanelLeftAnswer) {
        initComponents();
        repaintPanel=tempPanelLeftAnswer;

    }

    // definice designu stránky
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jTabbedPane1 = new javax.swing.JTabbedPane();
        jScrollPane1 = new javax.swing.JScrollPane();
```

```
        jPanel1 = new javax.swing.JPanel();
        jLabel5 = new javax.swing.JLabel();
        jPanel5 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
        jScrollPane2 = new javax.swing.JScrollPane();
        jPanel2 = new javax.swing.JPanel();
        jLabel4 = new javax.swing.JLabel();
        jPanel3 = new javax.swing.JPanel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jTextField2 = new javax.swing.JTextField();
        jTextField3 = new javax.swing.JTextField();
        jPanel4 = new javax.swing.JPanel();
        jButton4 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();

        setOpaque(false);
        setPreferredSize(new java.awt.Dimension(800, 600));

        jTabbedPane1.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mousePressed(java.awt.event.MouseEvent evt) {
                jTabbedPane1MousePressed(evt);
            }
        });
        jTabbedPane1.addMouseMotionListener(new
java.awt.event.MouseMotionAdapter() {
            public void mouseDragged(java.awt.event.MouseEvent evt) {
                jTabbedPane1MouseDragged(evt);
            }
        });

        jPanel1.setBackground(new java.awt.Color(143, 188, 143));
        jPanel1.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0), 3));
        jPanel1.setPreferredSize(new java.awt.Dimension(40, 40));

        jLabel5.setBackground(new java.awt.Color(205, 133, 63));
        jLabel5.setFont(new java.awt.Font("Tempus Sans ITC", 1, 14)); // NOI18N
        jLabel5.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel5.setText("CREATE CLUSTER");
        jLabel5.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0), 2));
        jLabel5.setOpaque(true);

        jPanel5.setBackground(new java.awt.Color(205, 133, 63));
        jPanel5.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));

        jLabel1.setBackground(new java.awt.Color(143, 188, 143));
        jLabel1.setFont(new java.awt.Font("Tempus Sans ITC", 1, 14)); // NOI18N
        jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel1.setText("Cluster Name");
        jLabel1.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
        jLabel1.setOpaque(true);

        jTextField1.setText("hostname");

        javax.swing.GroupLayout jPanel5Layout = new
javax.swing.GroupLayout(jPanel5);
        jPanel5.setLayout(jPanel5Layout);
        jPanel5Layout.setHorizontalGroup(

jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel5Layout.createSequentialGroup()
                .addGap(35, 35, 35)
                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE,
147, javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                .addGap(18, 18, 18)
                .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE, 160,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(30, Short.MAX_VALUE))
        );
        jPanel5Layout.setVerticalGroup(

jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel5Layout.createSequentialGroup()
                .addGap(26, 26, 26)

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE)
                    .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap(32, Short.MAX_VALUE))
        );

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(0, 0, Short.MAX_VALUE)
                .addComponent(jLabel5, javax.swing.GroupLayout.PREFERRED_SIZE,
230, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(89, 89, 89))
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jPanel5, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(12, Short.MAX_VALUE))
        );
        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(22, 22, 22)
                .addComponent(jLabel5)
                .addGap(33, 33, 33)
                .addComponent(jPanel5, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(62, Short.MAX_VALUE))
        );

        jScrollPane1.setViewportView(jPanel1);

        jTabbedPane1.addTab("Cluster", jScrollPane1);

        jScrollPane2.setBackground(new java.awt.Color(153, 153, 0));

        jPanel2.setBackground(new java.awt.Color(143, 188, 143));
        jPanel2.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0), 3));
        jPanel2.setPreferredSize(new java.awt.Dimension(40, 40));

        jLabel4.setBackground(new java.awt.Color(205, 133, 63));
        jLabel4.setFont(new java.awt.Font("Tempus Sans ITC", 1, 14)); // NOI18N
        jLabel4.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel4.setText("CREATE NODES");
        jLabel4.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0), 2));
```

```
        jLabel4.setOpaque(true);

        jPanel3.setBackground(new java.awt.Color(205, 133, 63));
        jPanel3.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
        jPanel3.setLayout(new org.Netbeans.lib.awtextra.AbsoluteLayout());

        jLabel2.setBackground(new java.awt.Color(143, 188, 143));
        jLabel2.setFont(new java.awt.Font("Tempus Sans ITC", 1, 14)); // NOI18N
        jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel2.setText("Node A");
        jLabel2.setOpaque(true);
        jPanel3.add(jLabel2, new
org.Netbeans.lib.awtextra.AbsoluteConstraints(22, 12, 100, 23));

        jLabel3.setBackground(new java.awt.Color(143, 188, 143));
        jLabel3.setFont(new java.awt.Font("Tempus Sans ITC", 1, 14)); // NOI18N
        jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel3.setText("Node B");
        jLabel3.setOpaque(true);
        jPanel3.add(jLabel3, new
org.Netbeans.lib.awtextra.AbsoluteConstraints(22, 41, 100, 20));

        jTextField2.setText("Node Name A");
        jPanel3.add(jTextField2, new
org.Netbeans.lib.awtextra.AbsoluteConstraints(140, 10, 210, 20));

        jTextField3.setText("Node Name B");
        jPanel3.add(jTextField3, new
org.Netbeans.lib.awtextra.AbsoluteConstraints(140, 40, 210, -1));

        jPanel4.setBackground(new java.awt.Color(205, 133, 63));
        jPanel4.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));

        jButton4.setText("Cancel");
        jButton4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton4ActionPerformed(evt);
            }
        });

        jButton2.setText("Save");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel4Layout = new
javax.swing.GroupLayout(jPanel4);
        jPanel4.setLayout(jPanel4Layout);
        jPanel4Layout.setHorizontalGroup(

jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel4Layout.createSequentialGroup()
                .addGap(34, 34, 34)
                .addComponent(jButton4)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jButton2)
                .addGap(43, 43, 43))
        );
        jPanel4Layout.setVerticalGroup(

jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel4Layout.createSequentialGroup()
                .addContainerGap()
```

```
.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE)
                    .addComponent(jButton4)
                    .addComponent(jButton2))
                .addContainerGap(14, Short.MAX_VALUE))
        );

        javax.swing.GroupLayout jPanel2Layout = new
javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING)
                    .addGroup(jPanel2Layout.createSequentialGroup()
                        .addGap(91, 91, 91)
                        .addComponent(jLabel4,
javax.swing.GroupLayout.PREFERRED_SIZE, 230,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(jPanel2Layout.createSequentialGroup()
                        .addGap(19, 19, 19)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING, false)
                            .addComponent(jPanel3,
javax.swing.GroupLayout.DEFAULT_SIZE, 373, Short.MAX_VALUE)
                            .addComponent(jPanel4,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))))
                .addContainerGap(22, Short.MAX_VALUE))
        );
        jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addGap(19, 19, 19)
                .addComponent(jLabel4)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE,
77, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jPanel4, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(30, Short.MAX_VALUE))
        );

        jScrollPane2.setViewportView(jPanel2);

        jTabbedPane1.addTab("Nodes", jScrollPane2);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addComponent(jTabbedPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 427,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(0, 373, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
```

```
                .addComponent(jTabbedPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 257,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(0, 343, Short.MAX_VALUE))
        );
    }// </editor-fold>

    private void jTabbedPane1MousePressed(java.awt.event.MouseEvent evt) {
        x = evt.getX();
    y = evt.getY();// TODO add your handling code here:
    }

    private void jTabbedPane1MouseDragged(java.awt.event.MouseEvent evt) {
        evt.translatePoint(evt.getComponent().getLocation().x-x,
evt.getComponent().getLocation().y-y);
    evt.getComponent().setLocation(evt.getX(), evt.getY());// TODO add your
handling code here:
    }

    // v případě potvrzení definovaných hodnot tlačítkem OK se provedou níže
uvedené aktivity
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

    // připojení do db derby
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("dipl2PU");
    EntityManager em = emf.createEntityManager();
    Nodes node = new Nodes();
    Cluster cluster = new Cluster();

    //provedení transakce vložení
    em.getTransaction().begin();

    cluster.setName(jTextField1.getText());
    em.persist(cluster);

    node.setIdnode(null);
    node.setNamenodes(jTextField2.getText());
    node.setNamecluster(jTextField1.getText());
    node.setTypnode('A');
    em.persist(node);
    em.getTransaction().commit();
    em.close();

    DbQuery db = new DbQuery();
    db.removeFO();
    db.setFO(db.getANode(node.getNamecluster()));

    em = emf.createEntityManager();
    em.getTransaction().begin();
    node.setIdnode(null);
    node.setNamenodes(jTextField3.getText());
    node.setNamecluster(jTextField1.getText());
    node.setTypnode('B');

    em.persist(node);
    em.getTransaction().commit();

    //výstup z volžení dat se zobrazí ve čtvrtém dolním výstupním panelu
    repaintPanel.removeAll();
    repaintPanel.add(new RepaintingN(node));
    repaintPanel.repaint();
    repaintPanel.revalidate();

    //vyčištění pravého horního vstupního panelu
    this.setVisible(false);
    this.repaint();
    this.revalidate();

    //uzavření databáze
```

```
    em.close();
    emf.close();

    }
    //v případě zmáčknutí tlačítka Cancel
    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    this.setVisible(false);
    this.repaint();
    this.revalidate();
    }


    // Variables declaration - do not modify
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton4;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JPanel jPanel5;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JTabbedPane jTabbedPane1;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JTextField jTextField2;
    private javax.swing.JTextField jTextField3;
    // End of variables declaration

    public String myGetObject() {

        return jTextField1.getName();
    }
```

A.2 Třída PanelOpen

```
/**
 * @author Dagmar Bendova
 */

package dipl2;

import db.Cluster;
import db.Nodes;
import java.util.Iterator;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.Query;
import javax.swing.JPanel;

public class PanelOpen extends javax.swing.JPanel {

     final private JPanel repaintPanel;

    //konstruktor třídy PanelOpen
    public PanelOpen(JPanel tempPanelLeftAnswer) {

        //design vsuptního content panelu
        initComponents();
        //nahraje vstupní data, známé konfigurace clusteru
        initCustom();
        repaintPanel=tempPanelLeftAnswer;

    }

    //design vstupní obrazovky
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jScrollPane1 = new javax.swing.JScrollPane();
        jPanel1 = new javax.swing.JPanel();
        jLabel3 = new javax.swing.JLabel();
        jPanel2 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        choice1 = new java.awt.Choice();
        jPanel3 = new javax.swing.JPanel();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();

        setOpaque(false);
        setPreferredSize(new java.awt.Dimension(800, 600));

        jPanel1.setBackground(new java.awt.Color(143, 188, 143));
        jPanel1.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0), 3));
        jPanel1.setPreferredSize(new java.awt.Dimension(40, 40));
        jPanel1.setLayout(new org.Netbeans.lib.awtextra.AbsoluteLayout());

        jLabel3.setBackground(new java.awt.Color(205, 133, 63));
        jLabel3.setFont(new java.awt.Font("Tempus Sans ITC", 1, 14)); // NOI18N
        jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel3.setText("OPEN CLUSTER");
        jLabel3.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0), 2));
        jLabel3.setOpaque(true);
        jPanel1.add(jLabel3, new org.Netbeans.lib.awtextra.AbsoluteConstraints(110, 20,
230, -1));
```

```
        jPanel2.setBackground(new java.awt.Color(205, 133, 63));
        jPanel2.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));

        jLabel1.setBackground(new java.awt.Color(143, 188, 143));
        jLabel1.setFont(new java.awt.Font("Tempus Sans ITC", 0, 12)); // NOI18N
        jLabel1.setText("Choose cluster name to open:");
        jLabel1.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
        jLabel1.setOpaque(true);

        javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addGap(19, 19, 19)
                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 162,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(20, 20, 20)
                .addComponent(choice1, javax.swing.GroupLayout.PREFERRED_SIZE, 140,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(37, Short.MAX_VALUE))
        );
        jPanel2Layout.setVerticalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addGap(20, 20, 20)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(choice1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap(28, Short.MAX_VALUE))
        );

        jPanel1.add(jPanel2, new org.Netbeans.lib.awtextra.AbsoluteConstraints(30, 60,
380, 70));

        jPanel3.setBackground(new java.awt.Color(205, 133, 63));
        jPanel3.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));

        jButton1.setText("Cancel");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setText("OK");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
        jPanel3.setLayout(jPanel3Layout);
        jPanel3Layout.setHorizontalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addGap(32, 32, 32)
                .addComponent(jButton1)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
158, Short.MAX_VALUE)
```

```
                .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 65,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(58, 58, 58))
        );
        jPanel3Layout.setVerticalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jButton1)
                    .addComponent(jButton2))
                .addContainerGap(14, Short.MAX_VALUE))
        );

        jPanel1.add(jPanel3, new org.Netbeans.lib.awtextra.AbsoluteConstraints(30, 160,
-1, 50));

        jScrollPane1.setViewportView(jPanel1);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(23, 23, 23)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 454,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(323, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(28, 28, 28)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 250,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(322, Short.MAX_VALUE))
        );
    }// </editor-fold>
    //tlačítko Cancel - vyčistí pravý dolní výstupní content panel od stávajících
konfigurací
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        this.setVisible(false);
        this.repaint();
        this.revalidate();
    }

    // button OK
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

    //smaže stávající konfiguraci
    repaintPanel.removeAll();

    EntityManagerFactory emf = Persistence.createEntityManagerFactory("dipl2PU");
    EntityManager em = emf.createEntityManager();
    em.getTransaction().begin();

    DbQuery db = new DbQuery();
    Nodes node = new Nodes();
    node.setNamecluster(choice1.getSelectedItem());

    //překreslí stávající clusterovou konfiguraci
    repaintPanel.removeAll();
    repaintPanel.add(new RepaintingN(node));
    repaintPanel.repaint();
    repaintPanel.revalidate();

    //promaže tabulku FO
    db.removeFO();
```

```
    //zaznamená se informace o aktivní clusterové konfigurace
    db.setFO(db.getANode(node.getNamecluster()));

    //smaže JPanel v pravém horním content panelu
    this.setVisible(false);
    this.repaint();
    this.revalidate();
    }

    //výběr všech clusterových konfigurací z tabulky CLUSTER
    private void initCustom() {
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("dipl2PU");
    EntityManager em = emf.createEntityManager();
    em.getTransaction().begin();

    choice1.addItem("");

    Query query = em.createNativeQuery(
                    "Select * FROM Cluster",Cluster.class);
    List<Nodes> results = query.getResultList();

    Iterator itr = results.iterator();

      while(itr.hasNext()) {
         Cluster element = (Cluster)itr.next();
         choice1.addItem(element.getName());
      }

    }


    // Variables declaration - do not modify
    private java.awt.Choice choice1;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JScrollPane jScrollPane1;
    // End of variables declaration

    public String myGetObject() {

        return choice1.getName();
    }
}
```

A.3 Třída Panel Remove

```
package dipl2;

import db.Cluster;
import db.Nodes;
import java.util.Iterator;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.Query;
import javax.swing.JPanel;

public class PanelRemove extends javax.swing.JPanel {

    int x,y;
    final private JPanel repaintPanel;

    public PanelRemove(JPanel tempPanelLeftAnswer) {

        //zobrazení vstupního okna pro smazání clusterové konfigurace
        initComponents();
        repaintPanel=tempPanelLeftAnswer;

        //pro naplnění vstupních hodnot pro prvek JChoice
        initCustom();
    }

    //grafická část pro zobrazení panelu remove
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jTabbedPane1 = new javax.swing.JTabbedPane();
        jPanel1 = new javax.swing.JPanel();
        jLabel3 = new javax.swing.JLabel();
        jPanel2 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        choice1 = new java.awt.Choice();
        jPanel3 = new javax.swing.JPanel();
        jButton4 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();

        setBackground(new java.awt.Color(205, 133, 63));

        jTabbedPane1.setBackground(new java.awt.Color(143, 188, 143));
        jTabbedPane1.addMouseMotionListener(new java.awt.event.MouseMotionAdapter() {
            public void mouseDragged(java.awt.event.MouseEvent evt) {
                jTabbedPane1MouseDragged(evt);
            }
        });
        jTabbedPane1.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mousePressed(java.awt.event.MouseEvent evt) {
                jTabbedPane1MousePressed(evt);
            }
        });

        jPanel1.setBackground(new java.awt.Color(143, 188, 143));
        jPanel1.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0), 3));
        jPanel1.setLayout(new org.Netbeans.lib.awtextra.AbsoluteLayout());

        jLabel3.setBackground(new java.awt.Color(205, 133, 63));
        jLabel3.setFont(new java.awt.Font("Tempus Sans ITC", 1, 14)); // NOI18N
        jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel3.setText("REMOVE CLUSTER");
```

```
        jLabel3.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0), 2));
        jLabel3.setOpaque(true);
        jPanel1.add(jLabel3, new org.Netbeans.lib.awtextra.AbsoluteConstraints(80, 20,
230, -1));

        jPanel2.setBackground(new java.awt.Color(205, 133, 63));
        jPanel2.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));

        jLabel1.setBackground(new java.awt.Color(143, 188, 143));
        jLabel1.setFont(new java.awt.Font("Tempus Sans ITC", 1, 12)); // NOI18N
        jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel1.setText("Remove cluster:");
        jLabel1.setOpaque(true);

        javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addGap(20, 20, 20)
                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 109,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 21,
Short.MAX_VALUE)
                .addComponent(choice1, javax.swing.GroupLayout.PREFERRED_SIZE, 161,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(27, 27, 27))
        );
        jPanel2Layout.setVerticalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addGap(21, 21, 21)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(choice1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel1))
                .addContainerGap(27, Short.MAX_VALUE))
        );

        jPanel1.add(jPanel2, new org.Netbeans.lib.awtextra.AbsoluteConstraints(20, 60,
340, 70));

        jPanel3.setBackground(new java.awt.Color(205, 133, 63));
        jPanel3.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));

        jButton4.setText("Cancel");
        jButton4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton4ActionPerformed(evt);
            }
        });

        jButton2.setText("Delete");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
        jPanel3.setLayout(jPanel3Layout);
        jPanel3Layout.setHorizontalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addGap(32, 32, 32)
```

```
                .addComponent(jButton4)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
145, Short.MAX_VALUE)
                .addComponent(jButton2)
                .addGap(33, 33, 33))
        );
        jPanel3Layout.setVerticalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jButton4)
                    .addComponent(jButton2))
                .addContainerGap(14, Short.MAX_VALUE))
        );

        jPanel1.add(jPanel3, new org.Netbeans.lib.awtextra.AbsoluteConstraints(20, 150,
340, 50));

        jTabbedPane1.addTab("Cluster", jPanel1);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jTabbedPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 384,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(232, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jTabbedPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 243,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(105, Short.MAX_VALUE))
        );
    }// </editor-fold>

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

    //otevření a připojení do databáze Cluster
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("dipl2PU");
    EntityManager em = emf.createEntityManager();
    Nodes node = new Nodes();

    em.getTransaction().begin();

    //smazání clusterové konfigurace ze všech tabulek databáze CLUSTER
    node.setNamecluster(choice1.getSelectedItem());
     Query query = em.createNativeQuery(
      "DELETE FROM Cluster WHERE name=?");
     query.setParameter(1, node.getNamecluster()).executeUpdate();

     query = em.createNativeQuery(
      "DELETE FROM Nodes WHERE NAMECLUSTER=?");
     query.setParameter(1, node.getNamecluster());//.executeUpdate();
     query.executeUpdate ();

     query = em.createNativeQuery(
      "DELETE FROM Apl WHERE NAMECLUSTER=?");
     query.setParameter(1, node.getNamecluster());//.executeUpdate();
     query.executeUpdate ();

     query = em.createNativeQuery(
      "DELETE FROM Fo WHERE NAMECLUFO=?");
```

```
     query.setParameter(1, node.getNamecluster());//.executeUpdate();
     query.executeUpdate ();

     query = em.createNativeQuery(
      "DELETE FROM Heartbeat WHERE NAMECLUSTER=?");
     query.setParameter(1, node.getNamecluster());//.executeUpdate();
     query.executeUpdate ();

     em.flush();

    em.getTransaction().commit();
    em.close();
    emf.close();

    //smazání pravého výstupního content panelu od aktivní clusterové konfigurace
    repaintPanel.removeAll();
    repaintPanel.repaint();
    repaintPanel.revalidate();

    //zavření vstupního okna Panel Remove
    this.setVisible(false);
    this.repaint();
    this.revalidate();
    }

    private void jTabbedPane1MousePressed(java.awt.event.MouseEvent evt) {
       x = evt.getX();
       y = evt.getY(); // TODO add your handling code here:
    }

    private void jTabbedPane1MouseDragged(java.awt.event.MouseEvent evt) {
        evt.translatePoint(evt.getComponent().getLocation().x-x,
evt.getComponent().getLocation().y-y);
        evt.getComponent().setLocation(evt.getX(), evt.getY());// TODO add your handling
code here:
    }

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    this.setVisible(false);
    this.repaint();
    this.revalidate();
    }

    private void initCustom() {
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("dipl2PU");
    final EntityManager em = emf.createEntityManager();
    em.getTransaction().begin();

    Query query = em.createNativeQuery(
                    "Select * FROM Cluster",Cluster.class);
    List<Cluster> results = query.getResultList();

    Iterator itr = results.iterator();
    choice1.addItem("");

    //volba choice je naplněna již existujícími jmény clusterů z tabulky CLUSTER
      while(itr.hasNext()) {
         Cluster element = (Cluster)itr.next();
         choice1.addItem(element.getName());
      }
       choice1.revalidate();
      choice1.repaint();

}

    // Variables declaration - do not modify
    private java.awt.Choice choice1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton4;
```

```
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JTabbedPane jTabbedPane1;
    // End of variables declaration
}
```

A.4 Třída Panel Update

```
package dipl2;

import db.Cluster;
import db.Nodes;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.util.Iterator;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.Query;
import javax.swing.JPanel;

public class PanelUpdate extends javax.swing.JPanel {

    private int x,y;
    final private JPanel repaintPanel;

    //konstruktor vytvoření nové třídy Panel Update s parametrem content panelu kam se
zobrazí požadovaný výstup
    public PanelUpdate(JPanel tempPanelLeftAnswer) {
        //zobrazení vstupního okna pro smazání clusterové konfigurace
        initComponents();
        repaintPanel=tempPanelLeftAnswer;
        //pro naplnění vstupních hodnot pro prvek JChoice
        initCustom();
    }

    //grafická část pro zobrazení panelu update
    //kod automaticky generovaný programem Netbeans na základě návrhu v design složce
knihovny SWING
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jTabbedPane1 = new javax.swing.JTabbedPane();
        jPanel1 = new javax.swing.JPanel();
        jLabel7 = new javax.swing.JLabel();
        jPanel5 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        choice1 = new java.awt.Choice();
        jTextField4 = new javax.swing.JTextField();
        jPanel4 = new javax.swing.JPanel();
        jButton4 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
        jPanel3 = new javax.swing.JPanel();
        jLabel2 = new javax.swing.JLabel();
        jPanel2 = new javax.swing.JPanel();
        jLabel3 = new javax.swing.JLabel();
        choice2 = new java.awt.Choice();
        jLabel6 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        choice3 = new java.awt.Choice();
        jTextField6 = new javax.swing.JTextField();
        jLabel8 = new javax.swing.JLabel();
        jPanel6 = new javax.swing.JPanel();
        jButton3 = new javax.swing.JButton();
        jButton1 = new javax.swing.JButton();

        setBackground(new java.awt.Color(205, 133, 63));

        jTabbedPane1.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mousePressed(java.awt.event.MouseEvent evt) {
```

```
                jTabbedPane1MousePressed(evt);
            }
        });
        jTabbedPane1.addMouseMotionListener(new java.awt.event.MouseMotionAdapter() {
            public void mouseDragged(java.awt.event.MouseEvent evt) {
                jTabbedPane1MouseDragged(evt);
            }
        });

        jPanel1.setBackground(new java.awt.Color(143, 188, 143));

        jLabel7.setBackground(new java.awt.Color(205, 133, 63));
        jLabel7.setFont(new java.awt.Font("Tempus Sans ITC", 1, 14)); // NOI18N
        jLabel7.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel7.setText("UPDATE CLUSTER");
        jLabel7.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0), 2));
        jLabel7.setOpaque(true);

        jPanel5.setBackground(new java.awt.Color(205, 133, 63));
        jPanel5.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));

        jLabel1.setBackground(new java.awt.Color(143, 188, 143));
        jLabel1.setFont(new java.awt.Font("Tempus Sans ITC", 1, 14)); // NOI18N
        jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel1.setText("Update cluster OLD name: ");
        jLabel1.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
        jLabel1.setOpaque(true);

        jLabel4.setBackground(new java.awt.Color(143, 188, 143));
        jLabel4.setFont(new java.awt.Font("Tempus Sans ITC", 1, 14)); // NOI18N
        jLabel4.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel4.setText("Update cluster NEW name:");
        jLabel4.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
        jLabel4.setOpaque(true);

        javax.swing.GroupLayout jPanel5Layout = new javax.swing.GroupLayout(jPanel5);
        jPanel5.setLayout(jPanel5Layout);
        jPanel5Layout.setHorizontalGroup(
            jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel5Layout.createSequentialGroup()
                .addGap(23, 23, 23)

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                    .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jLabel4, javax.swing.GroupLayout.DEFAULT_SIZE, 192,
Short.MAX_VALUE))

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                    .addGroup(jPanel5Layout.createSequentialGroup()
                        .addGap(20, 20, 20)
                        .addComponent(choice1, javax.swing.GroupLayout.PREFERRED_SIZE,
225, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(jPanel5Layout.createSequentialGroup()
                        .addGap(18, 18, 18)
                        .addComponent(jTextField4)))
                .addContainerGap(25, Short.MAX_VALUE))
        );
        jPanel5Layout.setVerticalGroup(
            jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel5Layout.createSequentialGroup()
                .addGap(20, 20, 20)
```

```
.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jLabel1)
                    .addComponent(choice1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel4)
                    .addComponent(jTextField4, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

        jPanel4.setBackground(new java.awt.Color(205, 133, 63));
        jPanel4.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));

        jButton4.setText("Cancel");
        jButton4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton4ActionPerformed(evt);
            }
        });

        jButton2.setText("Update");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel4Layout = new javax.swing.GroupLayout(jPanel4);
        jPanel4.setLayout(jPanel4Layout);
        jPanel4Layout.setHorizontalGroup(
            jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel4Layout.createSequentialGroup()
                .addGap(51, 51, 51)
                .addComponent(jButton4)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jButton2)
                .addGap(70, 70, 70))
        );
        jPanel4Layout.setVerticalGroup(
            jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel4Layout.createSequentialGroup()
                .addContainerGap(14, Short.MAX_VALUE)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jButton4)
                    .addComponent(jButton2))
                .addContainerGap())
        );

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jLabel7, javax.swing.GroupLayout.PREFERRED_SIZE, 230,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(146, 146, 146))
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(28, 28, 28)
```

```
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                    .addComponent(jPanel5, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jPanel4, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addContainerGap(29, Short.MAX_VALUE))
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(26, 26, 26)
                .addComponent(jLabel7)
                .addGap(18, 18, 18)
                .addComponent(jPanel5, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(34, 34, 34)
                .addComponent(jPanel4, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(51, Short.MAX_VALUE))
        );

        jTabbedPane1.addTab("Cluster", jPanel1);

        jPanel3.setBackground(new java.awt.Color(143, 188, 143));
        jPanel3.setLayout(new org.Netbeans.lib.awtextra.AbsoluteLayout());
        jPanel3.add(jLabel2, new org.Netbeans.lib.awtextra.AbsoluteConstraints(10, 116,
-1, -1));

        jPanel2.setBackground(new java.awt.Color(205, 133, 63));
        jPanel2.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));

        jLabel3.setBackground(new java.awt.Color(143, 188, 143));
        jLabel3.setFont(new java.awt.Font("Tempus Sans ITC", 1, 14)); // NOI18N
        jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel3.setText("Cluster name:");
        jLabel3.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
        jLabel3.setOpaque(true);

        jLabel6.setBackground(new java.awt.Color(143, 188, 143));
        jLabel6.setFont(new java.awt.Font("Tempus Sans ITC", 1, 14)); // NOI18N
        jLabel6.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel6.setText("Define OLD node:");
        jLabel6.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
        jLabel6.setOpaque(true);

        jLabel5.setBackground(new java.awt.Color(143, 188, 143));
        jLabel5.setFont(new java.awt.Font("Tempus Sans ITC", 1, 14)); // NOI18N
        jLabel5.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel5.setText("Define NEW node:");
        jLabel5.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
        jLabel5.setOpaque(true);

        javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel3, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
```

```
                    .addComponent(jLabel6, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jLabel5, javax.swing.GroupLayout.DEFAULT_SIZE, 165,
Short.MAX_VALUE))
                .addGap(24, 24, 24)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                    .addComponent(choice3, javax.swing.GroupLayout.DEFAULT_SIZE, 272,
Short.MAX_VALUE)
                    .addComponent(choice2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jTextField6))
                .addGap(17, 17, 17))
        );
        jPanel2Layout.setVerticalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addGap(19, 19, 19)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(choice2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel3))
                .addGap(18, 18, 18)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel6)
                    .addComponent(choice3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel5)
                    .addComponent(jTextField6, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

        jPanel3.add(jPanel2, new org.Netbeans.lib.awtextra.AbsoluteConstraints(20, 60,
490, 140));

        jLabel8.setBackground(new java.awt.Color(205, 133, 63));
        jLabel8.setFont(new java.awt.Font("Tempus Sans ITC", 1, 14)); // NOI18N
        jLabel8.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel8.setText("UPDATE NODES");
        jLabel8.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0), 2));
        jLabel8.setOpaque(true);
        jPanel3.add(jLabel8, new org.Netbeans.lib.awtextra.AbsoluteConstraints(140, 20,
230, -1));

        jPanel6.setBackground(new java.awt.Color(205, 133, 63));
        jPanel6.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));

        jButton3.setText("Cancel");
        jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton3ActionPerformed(evt);
            }
        });

        jButton1.setText("Update");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });
```

```
        javax.swing.GroupLayout jPanel6Layout = new javax.swing.GroupLayout(jPanel6);
        jPanel6.setLayout(jPanel6Layout);
        jPanel6Layout.setHorizontalGroup(
            jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel6Layout.createSequentialGroup()
                .addGap(65, 65, 65)
                .addComponent(jButton3)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
209, Short.MAX_VALUE)
                .addComponent(jButton1)
                .addGap(82, 82, 82))
        );
        jPanel6Layout.setVerticalGroup(
            jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel6Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jButton3)
                    .addComponent(jButton1))
                .addContainerGap(14, Short.MAX_VALUE))
        );

        jPanel3.add(jPanel6, new org.Netbeans.lib.awtextra.AbsoluteConstraints(20, 220,
490, 50));

        jTabbedPane1.addTab("Nodes", jPanel3);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jTabbedPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 549,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(72, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jTabbedPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 316,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(35, Short.MAX_VALUE))
        );
    }// </editor-fold>

    //tlačítko OK, provede změnu clusteru
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

    //otevření a připojení do databáze Cluster
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("dipl2PU");
    EntityManager em = emf.createEntityManager();

    Nodes node = new Nodes();
    em.getTransaction().begin();
    String nameOld = choice1.getSelectedItem();
    node.setNamecluster(jTextField4.getText());

    //změna se musí provést ve všech tabulkách databáze CLUSTER
     Query query = em.createNativeQuery(
      "UPDATE Cluster SET name=? WHERE name=?");
     query.setParameter(1, node.getNamecluster());
     query.setParameter(2, nameOld);
     query.executeUpdate();

     query = em.createNativeQuery(
```

```
      "UPDATE Nodes SET namecluster=? WHERE namecluster=?");
     query.setParameter(1, node.getNamecluster());
     query.setParameter(2, nameOld);
     query.executeUpdate();

    query = em.createNativeQuery(
      "UPDATE Fo SET nameclufo=? WHERE nameclufo=?");
     query.setParameter(1, node.getNamecluster());
     query.setParameter(2, nameOld);
     query.executeUpdate();

     query = em.createNativeQuery(
      "UPDATE Apl SET namecluster=? WHERE namecluster=?");
     query.setParameter(1, node.getNamecluster());
     query.setParameter(2, nameOld);
     query.executeUpdate();

    query = em.createNativeQuery(
      "UPDATE Heartbeat SET namecluster=? WHERE namecluster=?");
     query.setParameter(1, node.getNamecluster());
     query.setParameter(2, nameOld);
     query.executeUpdate();

     em.flush();

    em.getTransaction().commit();
    em.close();
    emf.close();

    this.setVisible(false);
    this.repaint();
    this.revalidate();

    //smazání staré clusterové konfigurace, nahrazení novou a překreslení
    repaintPanel.removeAll();
    repaintPanel.add(new RepaintingN(node));
    repaintPanel.repaint();
    repaintPanel.revalidate();

    //zaznamenání aktivní clusterové konfigurace do tabulky FO pro případ simulací nad
clusterem
    DbQuery db = new DbQuery();
    db.removeFO();
    db.setFO(db.getANode(node.getNamecluster()));
    }

    private void jTabbedPane1MousePressed(java.awt.event.MouseEvent evt) {
       x = evt.getX();
       y = evt.getY(); // TODO add your handling code here:
    }

    private void jTabbedPane1MouseDragged(java.awt.event.MouseEvent evt) {
        evt.translatePoint(evt.getComponent().getLocation().x-x,
evt.getComponent().getLocation().y-y);
        evt.getComponent().setLocation(evt.getX(), evt.getY());// TODO add your handling
code here:
    }

    //tlačítko OK, provede změnu názvu nodu
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

    //otevření a připojení do databáze Cluster
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("dipl2PU");
    EntityManager em = emf.createEntityManager();

    Nodes node = new Nodes();
    em.getTransaction().begin();

    //získání vstupních dat
```

```
node.setNamecluster(choice2.getSelectedItem());
node.setNamenodes(jTextField6.getText());
String nameOld = choice3.getSelectedItem();

//změna na zákl. vstupních dat v tabulkách NODES a Cluster
 Query query = em.createNativeQuery(
  "UPDATE Nodes SET namenodes=? WHERE namenodes=? AND namecluster=?");
 query.setParameter(1, node.getNamenodes());
 query.setParameter(2, nameOld);
 query.setParameter(3, node.getNamecluster());
 query.executeUpdate();

 query = em.createNativeQuery(
  "UPDATE Apl SET nameprimnode=? WHERE nameprimnode=? AND namecluster=?");
 query.setParameter(1, node.getNamenodes());
 query.setParameter(2, nameOld);
 query.setParameter(3, node.getNamecluster());
 query.executeUpdate();

 query = em.createNativeQuery(
  "UPDATE Apl SET namesecnode=? WHERE namesecnode=? AND namecluster=?");
 query.setParameter(1, node.getNamenodes());
 query.setParameter(2, nameOld);
 query.setParameter(3, node.getNamecluster());
 query.executeUpdate();

 query = em.createNativeQuery(
  "UPDATE Fo SET namenodfo=? WHERE namenodfo=? AND nameclufo=?");
 query.setParameter(1, node.getNamenodes());
 query.setParameter(2, nameOld);
 query.setParameter(3, node.getNamecluster());
 query.executeUpdate();

 query = em.createNativeQuery(
  "UPDATE Heartbeat SET namenode=? WHERE namenode=? AND namecluster=?");
 query.setParameter(1, node.getNamenodes());
 query.setParameter(2, nameOld);
 query.setParameter(3, node.getNamecluster());
 query.executeUpdate();

 em.flush();

 em.getTransaction().commit();

 //zavření vstupního okna Panel update
    this.setVisible(false);
    this.repaint();
    this.revalidate();

//smazání pravého výstupního content panelu od aktivní clusterové konfigurace
repaintPanel.removeAll();
repaintPanel.add(new RepaintingN(node));
repaintPanel.repaint();
repaintPanel.revalidate();

//zavření připojení do databáze
em.close();
emf.close();


DbQuery db = new DbQuery();
//promaže tabulku FO
db.removeFO();
//zaznamená se informace o aktivní clusterové konfigurace
db.setFO(db.getANode(node.getNamecluster()));

}

//tlačítko Cancel v záložce Cluster, smazání vstupního okna pavenl update
```

```
    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        this.setVisible(false);
        this.repaint();
        this.revalidate();
    }

    //tlačítko Cancel v záložce Nodes, smazání vstupního okna pavenl update
    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
        this.setVisible(false);
        this.repaint();
        this.revalidate();
    }

    private void initCustom() {

    //připojení do databáze Cluster
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("dipl2PU");
    final EntityManager em = emf.createEntityManager();
    em.getTransaction().begin();

    //získání položek cluster a nodes na základě vstupních údajů
    Query query = em.createNativeQuery(
                    "Select * FROM Cluster",Cluster.class);
    List<Cluster> results = query.getResultList();

    Iterator itr = results.iterator();
    choice2.addItem("");
    choice1.addItem("");

      while(itr.hasNext()) {
         Cluster element = (Cluster)itr.next();
         choice2.addItem(element.getName());
         choice1.addItem(element.getName());
      }

      choice2.revalidate();
      choice2.repaint();
      choice1.revalidate();
      choice1.repaint();

      //podle uživatelem zadané konfigurace clusteru, se automaticky vyhledají možné
konfigurace nodů, které patří vybranému clusteru
      choice2.addItemListener(new ItemListener() {

        @Override
        public void itemStateChanged(ItemEvent ie) {

            choice3.removeAll();
            choice3.addItem("");

            Query query = em.createNativeQuery(
                    "Select * FROM Nodes WHERE Namecluster=?",Nodes.class);
            query.setParameter(1, ie.getItem());
            List<Nodes> results = query.getResultList();

            //podle vybraného jména clusteru se automaticky vyberou nody a automaticky
se vloži do prvku JChoice
            Iterator itr = results.iterator();

                while(itr.hasNext()) {
                    Nodes element = (Nodes)itr.next();
                    choice3.addItem(element.getNamenodes());
            }
            choice3.revalidate();
            choice3.repaint();
        }

       }
      );
```

```
}

    // Variables declaration - do not modify
    private java.awt.Choice choice1;
    private java.awt.Choice choice2;
    private java.awt.Choice choice3;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton4;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JPanel jPanel5;
    private javax.swing.JPanel jPanel6;
    private javax.swing.JTabbedPane jTabbedPane1;
    private javax.swing.JTextField jTextField4;
    private javax.swing.JTextField jTextField6;
    // End of variables declaration
}
```

A.5 Třída Panel SIPNew

```
/**
 * @author Dagmar Bendova
 * Třída PanelSIP New generuje vstupní okno pro zadání Servisní IP adresy
 */

package dipl2;

import db.Cluster;
import db.Nodes;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.util.Iterator;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.Query;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class PanelSIPNew extends javax.swing.JPanel {

    final private JPanel repaintPanel;

    //konstruktor vytvoření nové třídy Panel SIP New s parametrem content panelu kam se
zobrazí požadovaný výstup
    public PanelSIPNew(JPanel tempPanelLeftAnswer) {
        //zobrazení vstupního okna
        initComponents();
        repaintPanel=tempPanelLeftAnswer;
        //pro naplnění vstupních hodnot pro prvek JChoice
        initCustom();
    }

    //grafická část pro zobrazení panelu SIPNew
    //kod automaticky generovaný programem Netbeans na základě návrhu v design složce
knihovny SWING
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jScrollPane1 = new javax.swing.JScrollPane();
        jPanel1 = new javax.swing.JPanel();
        jLabel5 = new javax.swing.JLabel();
        jPanel2 = new javax.swing.JPanel();
        jLabel4 = new javax.swing.JLabel();
        choice2 = new java.awt.Choice();
        jLabel1 = new javax.swing.JLabel();
        choice1 = new java.awt.Choice();
        jLabel3 = new javax.swing.JLabel();
        jTextField2 = new javax.swing.JTextField();
        jPanel3 = new javax.swing.JPanel();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();

        setOpaque(false);
        setPreferredSize(new java.awt.Dimension(800, 600));

        jPanel1.setBackground(new java.awt.Color(143, 188, 143));
        jPanel1.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0), 3));
        jPanel1.setPreferredSize(new java.awt.Dimension(40, 40));

        jLabel5.setBackground(new java.awt.Color(205, 133, 63));
        jLabel5.setFont(new java.awt.Font("Tempus Sans ITC", 1, 14)); // NOI18N
```

```
        jLabel5.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel5.setText("SERVICE IP");
        jLabel5.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0), 2));
        jLabel5.setOpaque(true);

        jPanel2.setBackground(new java.awt.Color(205, 133, 63));
        jPanel2.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));

        jLabel4.setBackground(new java.awt.Color(143, 188, 143));
        jLabel4.setFont(new java.awt.Font("Tempus Sans ITC", 1, 14)); // NOI18N
        jLabel4.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel4.setText("Choose CLUSTER:");
        jLabel4.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
        jLabel4.setOpaque(true);

        jLabel1.setBackground(new java.awt.Color(143, 188, 143));
        jLabel1.setFont(new java.awt.Font("Tempus Sans ITC", 1, 14)); // NOI18N
        jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel1.setText("Choose NODE:");
        jLabel1.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
        jLabel1.setOpaque(true);

        jLabel3.setBackground(new java.awt.Color(143, 188, 143));
        jLabel3.setFont(new java.awt.Font("Tempus Sans ITC", 1, 14)); // NOI18N
        jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel3.setText("Service IP:");
        jLabel3.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
        jLabel3.setOpaque(true);

        javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addGap(18, 18, 18)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel4, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jLabel3, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                    .addComponent(choice2, javax.swing.GroupLayout.DEFAULT_SIZE, 252,
Short.MAX_VALUE)
                    .addComponent(choice1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jTextField2))
                .addGap(19, 19, 19))
        );
        jPanel2Layout.setVerticalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addGap(19, 19, 19)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(choice2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(choice1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel3)
                    .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap(23, Short.MAX_VALUE))
        );

        jPanel3.setBackground(new java.awt.Color(205, 133, 63));
        jPanel3.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));

        jButton1.setText("Cancel");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setText("OK");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
        jPanel3.setLayout(jPanel3Layout);
        jPanel3Layout.setHorizontalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addGap(48, 48, 48)
                .addComponent(jButton1)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
225, Short.MAX_VALUE)
                .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 65,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(60, 60, 60))
        );
        jPanel3Layout.setVerticalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel3Layout.createSequentialGroup()
                .addContainerGap(14, Short.MAX_VALUE)

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jButton1)
                    .addComponent(jButton2))
                .addContainerGap())
        );

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(121, 121, 121)
                .addComponent(jLabel5, javax.swing.GroupLayout.PREFERRED_SIZE, 230,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(155, Short.MAX_VALUE))
            .addGroup(jPanel1Layout.createSequentialGroup()
```

```
                .addGap(22, 22, 22)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                    .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jPanel3, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addGap(0, 0, Short.MAX_VALUE))
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(22, 22, 22)
                .addComponent(jLabel5)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(44, 44, 44))
        );

        jScrollPane1.setViewportView(jPanel1);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(23, 23, 23)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 514,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(263, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(28, 28, 28)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 278,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(294, Short.MAX_VALUE))
        );
    }// </editor-fold>

    //tlačítko Cancel, zrušení vstupního okna
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        this.setVisible(false);
        this.repaint();
        this.revalidate();
    }

    //tlačítko OK, provede uložení servisní IP adresy
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

    repaintPanel.removeAll();

    //připojení do databáze Cluster
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("dipl2PU");
    EntityManager em = emf.createEntityManager();

    Nodes node = new Nodes();
    em.getTransaction().begin();

    //získání vstupních dat
    node.setNamecluster(choice2.getSelectedItem());
    node.setNamenodes(choice1.getSelectedItem());
```

```
    node.setServiceip(jTextField2.getText());

    //vložení vstupních dat do tabulky NODES
     Query query = em.createNativeQuery(
      "UPDATE Nodes SET serviceip=? WHERE namenodes=? AND namecluster=?");
     query.setParameter(1, node.getServiceip());
     query.setParameter(2, node.getNamenodes());
     query.setParameter(3, node.getNamecluster());
     int updated = query.executeUpdate();

    em.getTransaction().commit();

    //zavření spojení s databází
    em.close();
    emf.close();

      //smazání pravého výstupního content panelu od aktivní clusterové konfigurace
    repaintPanel.removeAll();
    repaintPanel.add(new RepaintingN(node));
    repaintPanel.repaint();
    repaintPanel.revalidate();

      //zavření vstupního okna Panel SIPNew
    this.setVisible(false);
    this.repaint();
    this.revalidate();

    DbQuery db = new DbQuery();
    //promaže tabulku FO a zaznamená se informace o aktivní clusterové konfigurace
    db.removeFO();
    db.setFO(db.getANode(node.getNamecluster()));
    }

private void initCustom() {

    //připojení do databáze Cluster a naplnění položek všemi existujícími názvy clusteru
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("dipl2PU");
    final EntityManager em = emf.createEntityManager();
    em.getTransaction().begin();

    choice2.addItem("");

    Query query = em.createNativeQuery(
                    "Select * FROM Cluster",Cluster.class);
    List<Nodes> results = query.getResultList();

    Iterator itr = results.iterator();

      while(itr.hasNext()) {
         Cluster element = (Cluster)itr.next();
         choice2.addItem(element.getName());
      }

      //podle uživatelem zadané konfigurace clusteru, se automaticky vyhledají možné
konfigurace nodů, které patří vybranému clusteru
      choice2.addItemListener(new ItemListener() {

        @Override
        public void itemStateChanged(ItemEvent ie) {
            //EntityManagerFactory emfPod =
Persistence.createEntityManagerFactory("dipl2PU");
            //EntityManager emPod = emfPod.createEntityManager();
            //em.getTransaction().begin();

            choice1.removeAll();

            Query query = em.createNativeQuery(
                    "Select * FROM Nodes WHERE Namecluster=?",Nodes.class);
            query.setParameter(1, ie.getItem());
```

```
            List<Nodes> results = query.getResultList();

            choice1.addItem("");
            Iterator itr = results.iterator();

                while(itr.hasNext()) {
                    Nodes element = (Nodes)itr.next();
                    choice1.addItem(element.getNamenodes());
            }
            choice1.revalidate();
            choice1.repaint();
           //em.close();
           //emf.close();


        }

       }
      );
}

    // Variables declaration - do not modify
    private java.awt.Choice choice1;
    private java.awt.Choice choice2;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextField jTextField2;
    // End of variables declaration
}
```

A.6 Třída Panel BIP

```
/**
 * @author Dagmar Bendova
 * Třída PanelBIP generuje vstupní okno pro zadání Boot IP adresy
 */

package dipl2;

import db.Cluster;
import db.Nodes;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.util.Iterator;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.Query;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class PanelBIPNew extends javax.swing.JPanel {

    final private JPanel repaintPanel;

    //konstruktor vytvoření nové třídy Panel BIP s parametrem content panelu kam se
zobrazí požadovaný výstup
    public PanelBIPNew(JPanel tempPanelLeftAnswer) {
        //zobrazení vstupního okna
        initComponents();
        repaintPanel=tempPanelLeftAnswer;
        //pro naplnění vstupních hodnot pro prvek JChoice
        initCustom();
    }

    //grafická část pro zobrazení panelu BIP
    //kod automaticky generovaný programem Netbeans na základě návrhu v design složce
knihovny SWING
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jScrollPane1 = new javax.swing.JScrollPane();
        jPanel1 = new javax.swing.JPanel();
        jLabel5 = new javax.swing.JLabel();
        jPanel3 = new javax.swing.JPanel();
        jLabel4 = new javax.swing.JLabel();
        choice2 = new java.awt.Choice();
        jLabel1 = new javax.swing.JLabel();
        choice1 = new java.awt.Choice();
        jLabel3 = new javax.swing.JLabel();
        jTextField2 = new javax.swing.JTextField();
        jPanel4 = new javax.swing.JPanel();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();

        setOpaque(false);
        setPreferredSize(new java.awt.Dimension(800, 600));

        jPanel1.setBackground(new java.awt.Color(143, 188, 143));
        jPanel1.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0), 3));
        jPanel1.setPreferredSize(new java.awt.Dimension(40, 40));

        jLabel5.setBackground(new java.awt.Color(205, 133, 63));
        jLabel5.setFont(new java.awt.Font("Tempus Sans ITC", 1, 14)); // NOI18N
```

```
        jLabel5.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel5.setText("BOOT IP");
        jLabel5.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0), 2));
        jLabel5.setOpaque(true);

        jPanel3.setBackground(new java.awt.Color(205, 133, 63));
        jPanel3.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));

        jLabel4.setBackground(new java.awt.Color(143, 188, 143));
        jLabel4.setFont(new java.awt.Font("Tempus Sans ITC", 1, 14)); // NOI18N
        jLabel4.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel4.setText("Choose CLUSTER:");
        jLabel4.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
        jLabel4.setOpaque(true);

        jLabel1.setBackground(new java.awt.Color(143, 188, 143));
        jLabel1.setFont(new java.awt.Font("Tempus Sans ITC", 1, 14)); // NOI18N
        jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel1.setText("Choose NODE:");
        jLabel1.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
        jLabel1.setOpaque(true);

        jLabel3.setBackground(new java.awt.Color(143, 188, 143));
        jLabel3.setFont(new java.awt.Font("Tempus Sans ITC", 1, 14)); // NOI18N
        jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel3.setText("Define Boot IP:");
        jLabel3.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
        jLabel3.setOpaque(true);

        javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
        jPanel3.setLayout(jPanel3Layout);
        jPanel3Layout.setHorizontalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addGap(22, 22, 22)

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel3, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jLabel1, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                    .addComponent(jLabel4, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 135, Short.MAX_VALUE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(choice2, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 243, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(choice1, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 243, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jTextField2,
javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED_SIZE, 243,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap())
        );
        jPanel3Layout.setVerticalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(choice2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                    .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(choice1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel3)
                    .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

        jPanel4.setBackground(new java.awt.Color(205, 133, 63));
        jPanel4.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));

        jButton1.setText("Cancel");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setText("OK");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel4Layout = new javax.swing.GroupLayout(jPanel4);
        jPanel4.setLayout(jPanel4Layout);
        jPanel4Layout.setHorizontalGroup(
            jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel4Layout.createSequentialGroup()
                .addGap(57, 57, 57)
                .addComponent(jButton1)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 65,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(51, 51, 51))
        );
        jPanel4Layout.setVerticalGroup(
            jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel4Layout.createSequentialGroup()
                .addContainerGap(14, Short.MAX_VALUE)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jButton1)
                    .addComponent(jButton2))
                .addContainerGap())
        );

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(25, 25, 25)
```

```
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                    .addComponent(jPanel3, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jPanel4, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addGap(0, 26, Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jLabel5, javax.swing.GroupLayout.PREFERRED_SIZE, 230,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(107, 107, 107))
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(21, 21, 21)
                .addComponent(jLabel5)
                .addGap(18, 18, 18)
                .addComponent(jPanel3, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGap(18, 18, 18)
                .addComponent(jPanel4, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(44, 44, 44))
        );

        jScrollPane1.setViewportView(jPanel1);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(23, 23, 23)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 481,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(296, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(28, 28, 28)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 285,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(287, Short.MAX_VALUE))
        );
    }// </editor-fold>

    //tlačítko Cancel, zrušení vstupního okna
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        this.setVisible(false);
        this.repaint();
        this.revalidate();
    }

    //tlačítko OK, provede uložení Boot IP adresy
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

    repaintPanel.removeAll();

    //připojení do databáze Cluster
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("dipl2PU");
    EntityManager em = emf.createEntityManager();

    Nodes node = new Nodes();
    em.getTransaction().begin();
```

```
    //získání vstupních dat
    node.setNamecluster(choice2.getSelectedItem());
    node.setNamenodes(choice1.getSelectedItem());
    node.setBootip(jTextField2.getText());

    //vložení vstupních dat do tabulky NODES, záznam již existuje proto UPDATE
     Query query = em.createNativeQuery(
      "UPDATE Nodes SET bootip=? WHERE namenodes=? AND namecluster=?");
     query.setParameter(1, node.getBootip());
     query.setParameter(2, node.getNamenodes());
     query.setParameter(3, node.getNamecluster());
     query.executeUpdate();

    // potvrzení transakce a zavření spojení s databází
    em.getTransaction().commit();
    em.close();
    emf.close();

     //smazání pravého výstupního content panelu od aktivní clusterové konfigurace
    repaintPanel.removeAll();
    repaintPanel.add(new RepaintingN(node));
    repaintPanel.repaint();
    repaintPanel.revalidate();

    //zavření vstupního okna Panel BIP
    this.setVisible(false);
    this.repaint();
    this.revalidate();

    //promaže tabulku FO a zaznamená se informace o aktivní clusterové konfigurace
    DbQuery db = new DbQuery();
    db.removeFO();
    db.setFO(db.getANode(node.getNamecluster()));

    }

private void initCustom() {

    //připojení do databáze Cluster a naplnění položek všemi existujícími názvy clusteru
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("dipl2PU");
    final EntityManager em = emf.createEntityManager();
    em.getTransaction().begin();

    Query query = em.createNativeQuery(
                    "Select * FROM Cluster",Cluster.class);
    List<Cluster> results = query.getResultList();

    Iterator itr = results.iterator();
    choice2.addItem("");

      while(itr.hasNext()) {
         Cluster element = (Cluster)itr.next();
         choice2.addItem(element.getName());
      }

      //podle uživatelem zadané konfigurace clusteru, se automaticky vyhledají možné
konfigurace nodů, které patří vybranému clusteru
      choice2.addItemListener(new ItemListener() {

        @Override
        public void itemStateChanged(ItemEvent ie) {
            //EntityManagerFactory emfPod =
Persistence.createEntityManagerFactory("dipl2PU");
            //EntityManager emPod = emfPod.createEntityManager();
            //em.getTransaction().begin();

            choice1.removeAll();
            choice1.addItem("");
```

```
            Query query = em.createNativeQuery(
                    "Select * FROM Nodes WHERE Namecluster=?",Nodes.class);
            query.setParameter(1, ie.getItem());
            List<Nodes> results = query.getResultList();

            Iterator itr = results.iterator();

                while(itr.hasNext()) {
                    Nodes element = (Nodes)itr.next();
                    choice1.addItem(element.getNamenodes());
            }
            choice1.revalidate();
            choice1.repaint();
           //em.close();
           //emf.close();


        }

       }
      );
}

    // Variables declaration - do not modify
    private java.awt.Choice choice1;
    private java.awt.Choice choice2;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextField jTextField2;
    // End of variables declaration

}
```

A.7 Třída Panel HBnew

```
/**
 * @author Dagmar Bendova
 * Třída PanelHBNew generuje vstupní okno pro zadání HeartBeatu
 */

package dipl2;

import db.*;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.util.Iterator;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.Query;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class PanelHBNew extends javax.swing.JPanel {

    final private JPanel repaintPanel;

    //konstruktor vytvoření nové třídy PanelHBNew s parametrem content panelu kam se
zobrazí požadovaný výstup
    public PanelHBNew(JPanel tempPanelLeftAnswer) {
        //zobrazení vstupního okna
        initComponents();
        repaintPanel=tempPanelLeftAnswer;
        //pro naplnění vstupních hodnot pro prvek JChoice
        initCustom();

        //iniciuje statické hodnoty prvkku choice3
        choice3.add("");
        choice3.add("disk over SAN/NAS");
        choice3.add("net");

    }

    //grafická část pro zobrazení paneluHBNew
    //kod automaticky generovaný programem Netbeans na základě návrhu v design složce
knihovny SWING
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jScrollPane1 = new javax.swing.JScrollPane();
        jPanel1 = new javax.swing.JPanel();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
        jLabel2 = new javax.swing.JLabel();
        jPanel2 = new javax.swing.JPanel();
        jLabel4 = new javax.swing.JLabel();
        jLabel8 = new javax.swing.JLabel();
        choice2 = new java.awt.Choice();
        choice1 = new java.awt.Choice();
        jPanel3 = new javax.swing.JPanel();
        jLabel5 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jTextField2 = new javax.swing.JTextField();
        choice3 = new java.awt.Choice();

        setOpaque(false);
        setPreferredSize(new java.awt.Dimension(800, 600));
```

```
        jPanel1.setBackground(new java.awt.Color(143, 188, 143));
        jPanel1.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0), 2));
        jPanel1.setPreferredSize(new java.awt.Dimension(40, 40));
        jPanel1.setLayout(new org.Netbeans.lib.awtextra.AbsoluteLayout());

        jButton1.setText("Cancel");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });
        jPanel1.add(jButton1, new org.Netbeans.lib.awtextra.AbsoluteConstraints(30, 270,
-1, -1));

        jButton2.setText("OK");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });
        jPanel1.add(jButton2, new org.Netbeans.lib.awtextra.AbsoluteConstraints(370,
270, 65, -1));

        jLabel2.setBackground(new java.awt.Color(205, 133, 63));
        jLabel2.setFont(new java.awt.Font("Tempus Sans ITC", 1, 18)); // NOI18N
        jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel2.setText("Heartbeat (HB)");
        jLabel2.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
        jLabel2.setOpaque(true);
        jPanel1.add(jLabel2, new org.Netbeans.lib.awtextra.AbsoluteConstraints(130, 10,
180, -1));

        jPanel2.setBackground(new java.awt.Color(205, 133, 63));
        jPanel2.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
        jPanel2.setLayout(new org.Netbeans.lib.awtextra.AbsoluteLayout());

        jLabel4.setBackground(new java.awt.Color(143, 188, 143));
        jLabel4.setFont(new java.awt.Font("Tempus Sans ITC", 1, 12)); // NOI18N
        jLabel4.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel4.setText("Choose CLUSTER:");
        jLabel4.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
        jLabel4.setOpaque(true);
        jPanel2.add(jLabel4, new org.Netbeans.lib.awtextra.AbsoluteConstraints(20, 20,
140, 20));

        jLabel8.setBackground(new java.awt.Color(143, 188, 143));
        jLabel8.setFont(new java.awt.Font("Tempus Sans ITC", 1, 12)); // NOI18N
        jLabel8.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel8.setText("Choose  NODE:");
        jLabel8.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
        jLabel8.setOpaque(true);
        jPanel2.add(jLabel8, new org.Netbeans.lib.awtextra.AbsoluteConstraints(20, 50,
140, 20));
        jPanel2.add(choice2, new org.Netbeans.lib.awtextra.AbsoluteConstraints(180, 50,
200, -1));
        jPanel2.add(choice1, new org.Netbeans.lib.awtextra.AbsoluteConstraints(180, 20,
200, -1));

        jPanel1.add(jPanel2, new org.Netbeans.lib.awtextra.AbsoluteConstraints(30, 50,
400, 90));

        jPanel3.setBackground(new java.awt.Color(205, 133, 63));
        jPanel3.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
```

```
        jPanel3.setLayout(new org.Netbeans.lib.awtextra.AbsoluteLayout());

        jLabel5.setBackground(new java.awt.Color(143, 188, 143));
        jLabel5.setFont(new java.awt.Font("Tempus Sans ITC", 1, 12)); // NOI18N
        jLabel5.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel5.setText("Define type HB:");
        jLabel5.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
        jLabel5.setOpaque(true);
        jPanel3.add(jLabel5, new org.Netbeans.lib.awtextra.AbsoluteConstraints(20, 54,
140, -1));

        jLabel3.setBackground(new java.awt.Color(143, 188, 143));
        jLabel3.setFont(new java.awt.Font("Tempus Sans ITC", 1, 12)); // NOI18N
        jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel3.setText("Choose type HB:");
        jLabel3.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
        jLabel3.setOpaque(true);
        jPanel3.add(jLabel3, new org.Netbeans.lib.awtextra.AbsoluteConstraints(20, 21,
140, -1));
        jPanel3.add(jTextField2, new org.Netbeans.lib.awtextra.AbsoluteConstraints(180,
50, 200, -1));
        jPanel3.add(choice3, new org.Netbeans.lib.awtextra.AbsoluteConstraints(180, 20,
200, -1));

        jPanel1.add(jPanel3, new org.Netbeans.lib.awtextra.AbsoluteConstraints(30, 160,
400, 90));

        jScrollPane1.setViewportView(jPanel1);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(23, 23, 23)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 463,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(314, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(28, 28, 28)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 310,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(262, Short.MAX_VALUE))
        );
    }// </editor-fold>

    //tlačítko Cancel, zrušení vstupního okna
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        this.setVisible(false);
        this.repaint();
        this.revalidate();// TODO add your handling code here:
    }

    //tlačítko OK, provede uložení Heartbeatu
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

    repaintPanel.removeAll();

    Nodes node = new Nodes();
    Heartbeat hb = new Heartbeat();
    DbQuery dbQuery = new DbQuery();

    //získání vstupních dat
    node.setNamecluster(choice1.getSelectedItem());
```

```
    node.setNamenodes(choice2.getSelectedItem());
    hb.setTypehb(choice3.getSelectedItem());
    hb.setValuehb(jTextField2.getText());

    //vložení info o heartbeatu do tabulky Heartbeat
    dbQuery.insertHB(node,hb);

    //zavření vstupního okna Panel HBNew
    this.setVisible(false);
    this.repaint();
    this.revalidate();

    //promaže tabulku FO a zaznamená se informace o aktivní clusterové konfigurace
    repaintPanel.removeAll();
    repaintPanel.add(new RepaintingN(node));
    repaintPanel.repaint();
    repaintPanel.revalidate();

    //promaže tabulku FO a zaznamená se informace o aktivní clusterové konfigurace
    dbQuery.removeFO();
    dbQuery.setFO(dbQuery.getANode(node.getNamecluster()));
    }

    private void initCustom() {

    //připojení do databáze Cluster a naplnění položek všemi existujícími názvy clusteru
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("dipl2PU");
    final EntityManager em = emf.createEntityManager();
    em.getTransaction().begin();

    choice1.addItem("");

    Query query = em.createNativeQuery(
                    "Select * FROM Cluster",Cluster.class);
    List<Nodes> results = query.getResultList();

    Iterator itr = results.iterator();

      while(itr.hasNext()) {
         Cluster element = (Cluster)itr.next();
         choice1.addItem(element.getName());
      }

      //podle uživatelem zadané konfigurace clusteru, se automaticky vyhledají možné
konfigurace nodů, které patří vybranému clusteru
      choice1.addItemListener(new ItemListener() {

        @Override
        public void itemStateChanged(ItemEvent ie) {
            //EntityManagerFactory emfPod =
Persistence.createEntityManagerFactory("dipl2PU");
            //EntityManager emPod = emfPod.createEntityManager();
            //em.getTransaction().begin();

            choice2.removeAll();

            Query query = em.createNativeQuery(
                    "Select * FROM Nodes WHERE Namecluster=?",Nodes.class);
            query.setParameter(1, ie.getItem());
            List<Nodes> results = query.getResultList();

            choice2.addItem("");
            Iterator itr = results.iterator();

                while(itr.hasNext()) {
                    Nodes element = (Nodes)itr.next();
                    choice2.addItem(element.getNamenodes());
            }
            choice2.revalidate();
```

```
             choice2.repaint();
            //em.close();
            //emf.close();


         }

        }
       );
    }

    // Variables declaration - do not modify
    private java.awt.Choice choice1;
    private java.awt.Choice choice2;
    private java.awt.Choice choice3;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextField jTextField2;
    // End of variables declaration


}
```

A.8 Třída PanelHBTable

```
/**
 *
 * @author Dagmar Bendova
 * Třída PanelHBTable zobrazí prvek JTable s informacemi o heartbeatech
 */

package dipl2;

import db.Nodes;
import db.Heartbeat;
import java.util.Iterator;
import java.util.List;
import javax.swing.table.DefaultTableModel;

public class PanelHBTable extends javax.swing.JPanel {

    private DbQuery dbQuery;
    private List<Heartbeat> hb;

    //konstruktor vytvoření nové třídy PanelHBTable, který naplní dle uvedeného nodu
tabulku JTable informacemi o heartbeatech
    public PanelHBTable(Nodes node) {
        initComponents();

        //získá informace o heartbeatech pro uvedený nod
        dbQuery = new DbQuery();
        hb = dbQuery.getQueryHB(node);

        //naplní jednotlivé prvky získanými položkami
        if (!hb.isEmpty()) {
            Iterator itr = hb.iterator();
            int i=0;
            while(itr.hasNext()) {
                Heartbeat element = (Heartbeat)itr.next();
                Object[] row = new Object[2];
                row[0] = element.getTypehb();
                row[1] = element.getValuehb();

                ((DefaultTableModel) jTable1.getModel()).insertRow(i,row);
                i++;
            }
        //v případě že ještě nejsou nadefinovány údaje o heartbeatech prvek JTable se
nebude zobrazovat
        } else {
            jScrollPane1.setVisible(false);
            jScrollPane1.repaint();
            jScrollPane1.revalidate();
        }

        jTable1.repaint();
        jTable1.revalidate();
    }

    //grafická část pro zobrazení paneluHBNew
    //kod automaticky generovaný programem Netbeans na základě návrhu v design složce
knihovny SWING
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jScrollPane1 = new javax.swing.JScrollPane();
        jTable1 = new javax.swing.JTable();

        setOpaque(false);
        setLayout(new org.Netbeans.lib.awtextra.AbsoluteLayout());
```

```
        jScrollPane1.setOpaque(false);

        jTable1.setAutoCreateRowSorter(true);
        jTable1.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null},
                {null, null}
            },
            new String [] {
                "Heartbeat", "Type"
            }
        ) {
            Class[] types = new Class [] {
                java.lang.String.class, java.lang.String.class
            };
            boolean[] canEdit = new boolean [] {
                false, false
            };

            public Class getColumnClass(int columnIndex) {
                return types [columnIndex];
            }

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
        jTable1.setOpaque(false);
        jScrollPane1.setViewportView(jTable1);

        add(jScrollPane1, new org.Netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 200,
60));
    }// </editor-fold>


    // Variables declaration - do not modify
    private javax.swing.JScrollPane jScrollPane1;
    public javax.swing.JTable jTable1;
    // End of variables declaration
}
```

A.9 Třída PanelAPLNew

```
/**
 * @author Dagmar Bendova
 * Třída PanelAplNew generuje vstupní okno pro zadání informací o resource groupě
 */

package dipl2;

import db.Apl;
import db.Cluster;
import db.Nodes;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.util.Iterator;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.Query;
import javax.swing.JPanel;

public class PanelAplNew extends javax.swing.JPanel {

    final private JPanel repaintPanel;
    private String nameCluster;


    //konstruktor vytvoření nové třídy PanelAplNew s parametrem content panelu, kam se
zobrazí požadovaný výstup
    public PanelAplNew(JPanel tempPanelLeftAnswer) {
        //zobrazení vstupního okna
        initComponents();
        repaintPanel=tempPanelLeftAnswer;

        //pro naplnění vstupních hodnot pro prvek JChoice
        initCustom();
    }

    //grafická část pro zobrazení paneluAplNew
    //kod automaticky generovaný programem Netbeans na základě návrhu v design složce
knihovny SWING
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jScrollPane1 = new javax.swing.JScrollPane();
        jPanel1 = new javax.swing.JPanel();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
        jLabel2 = new javax.swing.JLabel();
        jPanel2 = new javax.swing.JPanel();
        jLabel4 = new javax.swing.JLabel();
        jLabel8 = new javax.swing.JLabel();
        jLabel1 = new javax.swing.JLabel();
        choice2 = new java.awt.Choice();
        choice3 = new java.awt.Choice();
        choice1 = new java.awt.Choice();
        jPanel3 = new javax.swing.JPanel();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jLabel7 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jTextField3 = new javax.swing.JTextField();
        jTextField2 = new javax.swing.JTextField();
        jTextField4 = new javax.swing.JTextField();
        jTextField5 = new javax.swing.JTextField();
```

```
        setOpaque(false);
        setPreferredSize(new java.awt.Dimension(800, 600));

        jPanel1.setBackground(new java.awt.Color(143, 188, 143));
        jPanel1.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0), 2));
        jPanel1.setPreferredSize(new java.awt.Dimension(40, 40));
        jPanel1.setLayout(new org.Netbeans.lib.awtextra.AbsoluteLayout());

        jButton1.setText("Cancel");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });
        jPanel1.add(jButton1, new org.Netbeans.lib.awtextra.AbsoluteConstraints(30, 350,
-1, -1));

        jButton2.setText("OK");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });
        jPanel1.add(jButton2, new org.Netbeans.lib.awtextra.AbsoluteConstraints(400,
350, 65, -1));

        jLabel2.setBackground(new java.awt.Color(205, 133, 63));
        jLabel2.setFont(new java.awt.Font("Tempus Sans ITC", 1, 18)); // NOI18N
        jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel2.setText("Resource Group");
        jLabel2.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
        jLabel2.setOpaque(true);
        jPanel1.add(jLabel2, new org.Netbeans.lib.awtextra.AbsoluteConstraints(90, 10,
330, -1));

        jPanel2.setBackground(new java.awt.Color(205, 133, 63));
        jPanel2.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
        jPanel2.setLayout(new org.Netbeans.lib.awtextra.AbsoluteLayout());

        jLabel4.setFont(new java.awt.Font("Tempus Sans ITC", 1, 10)); // NOI18N
        jLabel4.setText("Choose CLUSTER:");
        jPanel2.add(jLabel4, new org.Netbeans.lib.awtextra.AbsoluteConstraints(20, 20,
130, 20));

        jLabel8.setFont(new java.awt.Font("Tempus Sans ITC", 1, 10)); // NOI18N
        jLabel8.setText("Choose Primary NODE:");
        jPanel2.add(jLabel8, new org.Netbeans.lib.awtextra.AbsoluteConstraints(20, 50,
130, 20));

        jLabel1.setFont(new java.awt.Font("Tempus Sans ITC", 1, 10)); // NOI18N
        jLabel1.setText("Choose Secondary NODE:");
        jPanel2.add(jLabel1, new org.Netbeans.lib.awtextra.AbsoluteConstraints(20, 80,
140, 20));
        jPanel2.add(choice2, new org.Netbeans.lib.awtextra.AbsoluteConstraints(170, 50,
250, -1));
        jPanel2.add(choice3, new org.Netbeans.lib.awtextra.AbsoluteConstraints(170, 80,
250, -1));
        jPanel2.add(choice1, new org.Netbeans.lib.awtextra.AbsoluteConstraints(170, 20,
250, -1));

        jPanel1.add(jPanel2, new org.Netbeans.lib.awtextra.AbsoluteConstraints(30, 50,
440, 120));

        jPanel3.setBackground(new java.awt.Color(205, 133, 63));
```

```
        jPanel3.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));

        jLabel5.setFont(new java.awt.Font("Tempus Sans ITC", 1, 10)); // NOI18N
        jLabel5.setText("Service IP:");

        jLabel6.setFont(new java.awt.Font("Tempus Sans ITC", 1, 10)); // NOI18N
        jLabel6.setText("Logical volume:");

        jLabel7.setFont(new java.awt.Font("Tempus Sans ITC", 1, 10)); // NOI18N
        jLabel7.setText("File system:");

        jLabel3.setFont(new java.awt.Font("Tempus Sans ITC", 1, 10)); // NOI18N
        jLabel3.setText("Name application server:");

        javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
        jPanel3.setLayout(jPanel3Layout);
        jPanel3Layout.setHorizontalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addGap(19, 19, 19)

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel3Layout.createSequentialGroup()

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                            .addComponent(jLabel6, javax.swing.GroupLayout.DEFAULT_SIZE,
138, Short.MAX_VALUE)
                            .addComponent(jLabel7, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                        .addGap(18, 18, 18)

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jTextField5,
javax.swing.GroupLayout.PREFERRED_SIZE, 255, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jTextField4,
javax.swing.GroupLayout.PREFERRED_SIZE, 255, javax.swing.GroupLayout.PREFERRED_SIZE)))
                    .addGroup(jPanel3Layout.createSequentialGroup()

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                            .addComponent(jLabel5, javax.swing.GroupLayout.DEFAULT_SIZE,
138, Short.MAX_VALUE)
                            .addComponent(jLabel3, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jTextField3,
javax.swing.GroupLayout.PREFERRED_SIZE, 255, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jTextField2,
javax.swing.GroupLayout.PREFERRED_SIZE, 255, javax.swing.GroupLayout.PREFERRED_SIZE))))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );
        jPanel3Layout.setVerticalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addGap(14, 14, 14)

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jLabel3)
                    .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
```

```
                    .addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel5))
                .addGap(7, 7, 7)

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jTextField4, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel6))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jTextField5, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel7))
                .addContainerGap(25, Short.MAX_VALUE))
        );

        jPanel1.add(jPanel3, new org.Netbeans.lib.awtextra.AbsoluteConstraints(30, 190,
440, 140));

        jScrollPane1.setViewportView(jPanel1);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(23, 23, 23)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 505,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(272, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(28, 28, 28)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 399,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(173, Short.MAX_VALUE))
        );
    }// </editor-fold>

    //tlačítko Cancel, zrušení vstupního okna
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        this.setVisible(false);
        this.repaint();
        this.revalidate();
    }

    //tlačítko OK, provede uložení Resource Groupy
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

    repaintPanel.removeAll();

    EntityManagerFactory emf = Persistence.createEntityManagerFactory("dipl2PU");
    EntityManager em = emf.createEntityManager();

    Apl apl = new Apl();
    Nodes node = new Nodes();
    em.getTransaction().begin();

    //získání vstupních dat
    apl.setNamecluster(choice1.getSelectedItem());
    apl.setNameapl(jTextField2.getText());
    apl.setNamePrimNode(choice2.getSelectedItem());
    apl.setNameSecNode(choice3.getSelectedItem());
    apl.setIp(jTextField3.getText());
    apl.setLv(jTextField4.getText());
```

```
    apl.setFs(jTextField5.getText());

    //uložení dat do databáze
    em.persist(apl);
    em.getTransaction().commit();

    em.close();
    emf.close();

    //překreslí stávající clusterovou konfiguraci
    repaintPanel.removeAll();
    node.setNamenodes(apl.getNamePrimNode());
    node.setNamecluster(apl.getNamecluster());
    repaintPanel.add(new RepaintingN(node));

    repaintPanel.repaint();
    repaintPanel.revalidate();

    //zavření vstupního okna Panel APLNew
    this.setVisible(false);
    this.repaint();
    this.revalidate();

    //promaže tabulku FO a zaznamená se informace o aktivní clusterové konfigurace
    DbQuery dbQuery = new DbQuery();
    dbQuery.removeFO();
    dbQuery.setFO(dbQuery.getANode(node.getNamecluster()));
    }

private void initCustom() {

    choice1.addItem("");

    //připojení do databáze Cluster a naplnění položek všemi existujícími názvy clusteru
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("dipl2PU");
    final EntityManager em = emf.createEntityManager();
    em.getTransaction().begin();

    Query query = em.createNativeQuery(
                    "Select * FROM Cluster",Cluster.class);
    List<Nodes> results = query.getResultList();

    Iterator itr = results.iterator();
      while(itr.hasNext()) {
         Cluster element = (Cluster)itr.next();
         choice1.addItem(element.getName());
      }

      //podle uživatelem zadané konfigurace clusteru, se automaticky vyhledají možné
konfigurace nodů, které patří vybranému clusteru
      choice1.addItemListener(new ItemListener() {

        @Override
        public void itemStateChanged(ItemEvent ie) {

            nameCluster=ie.getItem().toString();
            choice2.removeAll();
            choice2.addItem("");

            Query query = em.createNativeQuery(
                    "Select * FROM Nodes WHERE Namecluster=?",Nodes.class);
            query.setParameter(1, ie.getItem().toString());
            List<Nodes> results = query.getResultList();

            Iterator itr = results.iterator();

                while(itr.hasNext()) {
                    Nodes element = (Nodes)itr.next();
                    choice2.addItem(element.getNamenodes());
```

```
            }
            //podle uživatelem zadaný název nodu, se automaticky vyhledají možné
konfigurace nodů. Pokud uživatel zvolil přimární nod nabídne se sekundární a nopak.
            choice2.addItemListener(new ItemListener() {

                @Override
                public void itemStateChanged(ItemEvent ie) {
                    choice3.removeAll();
                    choice3.addItem("");

                    Query query = em.createNativeQuery(
                        "Select * FROM Nodes WHERE Namecluster=?",Nodes.class);
                    query.setParameter(1, nameCluster);
                    List<Nodes> results = query.getResultList();

                    Iterator itr = results.iterator();

                    while(itr.hasNext()) {
                        Nodes element = (Nodes)itr.next();
                        if (!element.getNamenodes().equals(ie.getItem())) {
                            choice3.addItem(element.getNamenodes());
                        }
                    }
                }
            }
            );
        }
       }
      );
}

    // Variables declaration - do not modify
    private java.awt.Choice choice1;
    private java.awt.Choice choice2;
    private java.awt.Choice choice3;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextField jTextField2;
    private javax.swing.JTextField jTextField3;
    private javax.swing.JTextField jTextField4;
    private javax.swing.JTextField jTextField5;
    // End of variables declaration

}
```

A.10 Třída PanelAplTable

```
/**
 * @author Dagmar Bendova
 * Třída PanelAplTable zobrazí prvek JTable a naplního obsahem databázové tabulky Apl
 * Tabulka jTable bude obsahovat informace o resource groupě
 */

package dipl2;

import db.Apl;
import db.Nodes;
import java.util.Iterator;
import java.util.List;
import javax.swing.table.DefaultTableModel;

public class PanelAplTable extends javax.swing.JPanel {

    final private DbQuery dbQuery;
    final private List<Apl> apl;
    private boolean failover;

    //konstruktor vytvoření nové třídy PanelAplTable, který naplní dle uvedeného nodu
tabulku JTable položkami resource groupy
    public PanelAplTable(Nodes node) {
        failover = false;
        initComponents();

        // získání resource groupy z tabulky Apl
        dbQuery = new DbQuery();
        apl = dbQuery.getApl(node);

        Iterator itr = apl.iterator();
        int i=0;
        //naplní jednotlivé řádky tabulky položkami resource groupy
        while(itr.hasNext()) {
             Apl element = (Apl)itr.next();
            Object[] row = new Object[3];
            row[0] = element.getNameapl();
            row[1] = element.getIp();
            row[2] = element.getFs();

            ((DefaultTableModel) jTable1.getModel()).insertRow(i,row);
            i++;
        }

        //pokud není ještě resource groupa definovaná tabulka se nebude zobrazovat
        if (apl.isEmpty()) {
            jScrollPane1.setVisible(false);
            failover=true;
        }

        jTable1.repaint();
        jTable1.revalidate();
        }

    public boolean getAplFO(){
     return failover;
    }

    //grafická část pro zobrazení paneluAplNew
    //kod automaticky generovaný programem Netbeans na základě návrhu v design složce
knihovny SWING
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
```

```
        jScrollPane1 = new javax.swing.JScrollPane();
        jTable1 = new javax.swing.JTable();

        setOpaque(false);
        setLayout(new org.Netbeans.lib.awtextra.AbsoluteLayout());

        jScrollPane1.setOpaque(false);

        jTable1.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null, null},
                {null, null, null},
                {null, null, null}
            },
            new String [] {
                "APL", "IP", "FS"
            }
        ) {
            Class[] types = new Class [] {
                java.lang.String.class, java.lang.String.class, java.lang.String.class
            };
            boolean[] canEdit = new boolean [] {
                false, false, false
            };

            public Class getColumnClass(int columnIndex) {
                return types [columnIndex];
            }

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
        jTable1.setOpaque(false);
        jScrollPane1.setViewportView(jTable1);

        add(jScrollPane1, new org.Netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 220,
70));
    }// </editor-fold>


    // Variables declaration - do not modify
    private javax.swing.JScrollPane jScrollPane1;
    public javax.swing.JTable jTable1;
    // End of variables declaration
}
```

A.11 Třída PanelSave

```
/**
 * @author Dagmar Bendova
 * Třída PanelSave, vstupní okno pro definici souboru
 * Uložení vybrané clusterové konfigurace do specifikovaného souboru ve vstupním okně
 */

package dipl2;

import db.*;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.io.*;
import java.util.Iterator;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.Query;
import javax.swing.JPanel;
import com.x5.template.Theme;
import com.x5.template.Chunk;

public class PanelSave extends javax.swing.JPanel {

    final private JPanel repaintPanel;

    //konstruktor vytvoření nové třídy PanelSave s parametrem content panelu, kam se
zobrazí požadovaný výstup
    public PanelSave(JPanel tempPanelLeftAnswer) {

        //zobrazení vstupního okna
        initComponents();
        //pro naplnění vstupních hodnot pro prvek JChoice
        initCustom();
        repaintPanel=tempPanelLeftAnswer;

    }

    //grafická část pro zobrazení panelSave
    //kod automaticky generovaný programem Netbeans na základě návrhu v design složce
knihovny SWING
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jScrollPane1 = new javax.swing.JScrollPane();
        jPanel1 = new javax.swing.JPanel();
        jPanel2 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        choice1 = new java.awt.Choice();
        jLabel2 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jPanel3 = new javax.swing.JPanel();
        jButton2 = new javax.swing.JButton();
        jButton1 = new javax.swing.JButton();
        jLabel3 = new javax.swing.JLabel();

        setOpaque(false);
        setPreferredSize(new java.awt.Dimension(800, 600));

        jPanel1.setBackground(new java.awt.Color(143, 188, 143));
        jPanel1.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0), 2));
        jPanel1.setPreferredSize(new java.awt.Dimension(40, 40));
        jPanel1.setLayout(new org.Netbeans.lib.awtextra.AbsoluteLayout());
```

```
        jPanel2.setBackground(new java.awt.Color(205, 133, 63));
        jPanel2.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));

        jLabel1.setBackground(new java.awt.Color(143, 188, 143));
        jLabel1.setFont(new java.awt.Font("Tempus Sans ITC", 1, 12)); // NOI18N
        jLabel1.setText(" Choose cluster name to save:");
        jLabel1.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
        jLabel1.setOpaque(true);

        jLabel2.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));

        jLabel4.setBackground(new java.awt.Color(143, 188, 143));
        jLabel4.setFont(new java.awt.Font("Tempus Sans ITC", 1, 10)); // NOI18N
        jLabel4.setText(" File where to cluster will be saved:");
        jLabel4.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
        jLabel4.setOpaque(true);

        javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addGap(21, 21, 21)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                    .addComponent(jLabel4, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 27,
Short.MAX_VALUE)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(choice1, javax.swing.GroupLayout.PREFERRED_SIZE, 220,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 220,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(30, 30, 30))
        );
        jPanel2Layout.setVerticalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(choice1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                    .addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jLabel4, javax.swing.GroupLayout.DEFAULT_SIZE, 23,
Short.MAX_VALUE))
                .addContainerGap(16, Short.MAX_VALUE))
        );

        jPanel1.add(jPanel2, new org.Netbeans.lib.awtextra.AbsoluteConstraints(30, 70,
490, 90));
```

```
        jPanel3.setBackground(new java.awt.Color(205, 133, 63));
        jPanel3.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));

        jButton2.setText("OK");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        jButton1.setText("Cancel");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
        jPanel3.setLayout(jPanel3Layout);
        jPanel3Layout.setHorizontalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel3Layout.createSequentialGroup()
                .addGap(62, 62, 62)
                .addComponent(jButton1)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
228, Short.MAX_VALUE)
                .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 65,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(68, 68, 68))
        );
        jPanel3Layout.setVerticalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jButton2)
                    .addComponent(jButton1))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

        jPanel1.add(jPanel3, new org.Netbeans.lib.awtextra.AbsoluteConstraints(30, 170,
490, 50));

        jLabel3.setBackground(new java.awt.Color(205, 133, 63));
        jLabel3.setFont(new java.awt.Font("Tempus Sans ITC", 1, 14)); // NOI18N
        jLabel3.setText("     SAVE CONFIGURATION");
        jLabel3.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0), 2));
        jLabel3.setOpaque(true);
        jPanel1.add(jLabel3, new org.Netbeans.lib.awtextra.AbsoluteConstraints(160, 20,
230, -1));

        jScrollPane1.setViewportView(jPanel1);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(23, 23, 23)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 549,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(228, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
            .addGroup(layout.createSequentialGroup()
                .addGap(19, 19, 19)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 241,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(340, Short.MAX_VALUE))
        );
    }// </editor-fold>

    //tlačítko Cancel, zrušení vstupního okna
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        this.setVisible(false);
        this.repaint();
        this.revalidate();
    }

    //tlačítko OK, provede uložení vybrané clusterové konfigurace
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

    repaintPanel.removeAll();

    //získání názvu clusteru
    DbQuery db = new DbQuery();
    Nodes node = new Nodes();
    node.setNamecluster(choice1.getSelectedItem());

    //vytvoření souboru, kde bude konfigurace uložena
    String name="C:/clderby/"+node.getNamecluster()+".odm";
    File f = new File(name);
    try{
    f.createNewFile();
    } catch (IOException e){
        System.out.println("Nelze vytvorit soubor.");
    }

     //doplnění do šablony informace ze všech tabulek databaze
     Theme theme = new Theme();
     Chunk chunk = theme.makeChunk("template");

     chunk.set("cluster", node.getNamecluster());

     List<Nodes> listN= db.getNode(node.getNamecluster());
     chunk.set("nodeA",listN.get(0).getNamenodes());
     chunk.set("nodeB",listN.get(1).getNamenodes());
     chunk.set("serviceipA",listN.get(0).getServiceip());
     chunk.set("serviceipB",listN.get(1).getServiceip());
     chunk.set("bootipA",listN.get(0).getBootip());
     chunk.set("bootipB",listN.get(1).getBootip());

     node.setNamenodes(listN.get(0).getNamenodes());
     List<Apl> listA = db.getApl(node);
     Iterator itr = listA.iterator();
     int i=0;
      while(itr.hasNext()) {
         Apl element = (Apl)itr.next();
         chunk.set("apl"+i,listA.get(i).getNameapl());
         chunk.set("aplSIP"+i,listA.get(i).getIp());
         chunk.set("apllv"+i,listA.get(i).getLv());
         chunk.set("aplfs"+i,listA.get(i).getFs());
         i++;
      }
    List<Heartbeat> listH = db.getQueryHB(node);
    itr = listH.iterator();
     int j=0;
      while(itr.hasNext()) {
         Heartbeat element = (Heartbeat)itr.next();
         chunk.set("node"+j,listN.get(0).getNamenodes());
         if (element.getTypehb().contains("net"))
                              chunk.set("net"+j,"ether");
         else
```

```
                            chunk.set("net"+j,"diskhb");
        chunk.set("nodeHIB"+j,listH.get(j).getValuehb());
        j++;
     }

    node.setNamenodes(listN.get(1).getNamenodes());
    listA = db.getApl(node);
    while(itr.hasNext()) {
        Apl element = (Apl)itr.next();
        chunk.set("apl"+i,listA.get(i).getNameapl());
        chunk.set("aplSIP"+i,listA.get(i).getIp());
        chunk.set("apllv"+i,listA.get(i).getLv());
        chunk.set("aplfs"+i,listA.get(i).getFs());
        i++;
     }

    listH = db.getQueryHB(node);
       itr = listH.iterator();
       i=0;
     while(itr.hasNext()) {
        Heartbeat element = (Heartbeat)itr.next();
       //if (element != null) {
        //String n=element.getTypehb();
        chunk.set("node"+j,listN.get(1).getNamenodes());
        if (element.getTypehb().contains("net"))
                            chunk.set("net"+j,"ether");
        else
                            chunk.set("net"+j,"diskhb");
        chunk.set("nodeHIB"+j,listH.get(i).getValuehb());
        j++;
        i++;

     }

     //zápis vytvořené šablony do souboru
    FileWriter out;
    try{
        out = new FileWriter(name);
       chunk.render( out );

       out.flush();
       out.close();
     } catch (IOException e){
        System.out.println("Nelze vytvorit soubor.");
    }

   //zavření vstupního okna PanelSave
   this.setVisible(false);
   this.repaint();
   this.revalidate();
   }

   private void initCustom() {
   EntityManagerFactory emf = Persistence.createEntityManagerFactory("dipl2PU");
   EntityManager em = emf.createEntityManager();
   em.getTransaction().begin();

   choice1.addItem("");

   //načtení všech existujících clusterových konfigurací do prvku JChoice
   Query query = em.createNativeQuery(
                   "Select * FROM Cluster",Cluster.class);
   List<Nodes> results = query.getResultList();

   Iterator itr = results.iterator();

     while(itr.hasNext()) {
        Cluster element = (Cluster)itr.next();
        choice1.addItem(element.getName());
```

```
      }

    //v případě výběru názvu clusteru, program automaticky doplní cestu, kde si ho
uživatel může vyzvednout
    choice1.addItemListener(new ItemListener() {

        @Override
        public void itemStateChanged(ItemEvent ie) {
            String item = choice1.getSelectedItem();
            item="C:/clderby/"+item+".odm";
            jLabel2.setText(item);
            jPanel2.repaint();
            jPanel2.revalidate();
        }
    }
    );

    }

    // Variables declaration - do not modify
    private java.awt.Choice choice1;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JScrollPane jScrollPane1;
    // End of variables declaration

    public String myGetObject() {

        return choice1.getName();
    }
}
```

A.12 Třída Mainy

```
/**
 * @author Dagmar Bendova
 * Třída Mainy spouští aplikaci, generuje JFrame
 */

package dipl2;

import java.awt.BorderLayout;
import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JMenuBar;
import javax.swing.JMenu;
import javax.swing.JMenuItem;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.SQLException;
import javax.swing.ImageIcon;

public class Mainy extends javax.swing.JFrame {

        final private JPanel contentPane;
         final private Desktop4 desktop;
         private boolean setContentPane;

        /**
         * Launch the application.
         */
        public static void main(String[] args) {
                EventQueue.invokeLater(new Runnable() {
                        public void run() {
                                try {
                                        //vytvoření rámce
                                    Mainy frame = new Mainy();
                                        frame.setIconImage(new
ImageIcon(getClass().getResource("cpu.jpg")).getImage());

frame.setExtendedState(java.awt.Frame.MAXIMIZED_BOTH);
                                        frame.setTitle("HACMP course");
                                        frame.pack();
                                    frame.setVisible(true);
                                } catch (Exception e) {
                                        e.printStackTrace();
                                        printSQLException((SQLException) e);
                                }
                        }
                });
        }

         public static void printSQLException(SQLException e)
    {
         // ošetření SQL chyb
         while (e != null)
         {
             System.err.println("\n----- SQLException -----");
             System.err.println("  SQL State:  " + e.getSQLState());
             System.err.println("  Error Code: " + e.getErrorCode());
             System.err.println("  Message:    " + e.getMessage());
             e = e.getNextException();
         }
    }

        public Mainy(){
```

```
                //definice jednotl. položek menu
                setContentPane=false;
                MenuActionListener mal = new MenuActionListener();

               setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
               setBounds(100, 100, 500, 521);

               JMenuBar menuBar = new JMenuBar();
               setJMenuBar(menuBar);

               JMenu mnFile = new JMenu("File");
               menuBar.add(mnFile);

                JMenu Act = new JMenu("Action");
               menuBar.add(Act);

                JMenuItem TO = new JMenuItem("Takeover");
                TO.addActionListener(mal);
               Act.add(TO);

                JMenuItem FO = new JMenuItem("Failover");
                FO.addActionListener(mal);
               Act.add(FO);

                JMenuItem save = new JMenuItem("Save");
                save.addActionListener(mal);
               Act.add(save);

               JMenuItem mntmNew = new JMenuItem("New");
                mntmNew.addActionListener(mal);
               mnFile.add(mntmNew);

                JMenuItem mntmClose = new JMenuItem("Clean");
                mntmClose.addActionListener(mal);

                JMenuItem mntmExit = new JMenuItem("Exit");
                mntmExit.addActionListener(mal);

               mnFile.add(mntmClose);

                //vytvoření content panelu
               contentPane = new JPanel();
               contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
               contentPane.setLayout(new BorderLayout(0, 0));
               setContentPane(contentPane);

                PanelUvod panelU = new PanelUvod();
                contentPane.add(panelU);

               desktop = new Desktop4();

}
         private class MenuActionListener implements ActionListener {
            public void actionPerformed(ActionEvent e) {

                //generuje nový contentpanel rozdělených do čtyř částí
                if (e.getActionCommand().equals("New")) {
                    contentPane.removeAll();
                   contentPane.add(desktop);
                    contentPane.repaint();
                    contentPane.revalidate();
                    setContentPane=true;
                }

                //vyčistí obsah čtvrtého (pravého dolního content panelu
                if ((e.getActionCommand().equals("Clean")) && (setContentPane)) {
                    desktop.getCleanPanel();
                }
```

```
                //spouští utilitu migrace package ruční
                if (e.getActionCommand().equals("Takeover")) {
                    desktop.runTO();
                }

                //spouští utilitu migrace package automaticku
                if (e.getActionCommand().equals("Failover")) {
                    desktop.runFO();
                }

                if (e.getActionCommand().equals("Save")) {
                    //desktop.getCleanPanel();
                    desktop.save();

                }

                if (e.getActionCommand().equals("Exit")) {
                    desktop.fin();
                }
            }


}
}
```

A.13 Třída Desktop4

```
/**
 *
 * @author Dagmar Bendova
 */

package dipl2;

import db.Fo;
import javax.swing.JPanel;
import java.awt.Color;
import java.awt.GridBagLayout;
import java.awt.GridBagConstraints;
import java.awt.Insets;
import java.awt.ComponentOrientation;
import org.apache.commons.io.FileUtils;
import javax.swing.JScrollPane;
import javax.swing.border.LineBorder;
import db.*;
import java.io.File;
import java.io.IOException;

public class Desktop4 extends JPanel  {

        final private JPanel panelLeftAnswer;
         final private JPanel panelLeftAsk;
         private Boolean existing = true;

        public Desktop4() {

                 panelLeftAnswer = new JPanel();
                 panelLeftAsk = new JPanel();

                 //pokud databáze neexistuje, vytvoří se adresář, kam se databáze uloží
                 createFile();
                 DbTest dbTest=new DbTest();
                 if (!new File(dbTest.getPath()).exists()) {
                     existing = false;
                 }
                 //otevření databáze, vyhledání správného driveru
                 DbQuery db = new DbQuery();
                 boolean dbO = dbTest.dbOpen(dbTest.getDriver());

                 //pokud db neexistuje
                 //spustí se procedura vytvoření databáze, naplnění dat tabulek databáze
s jedním příkladem clusterové konfigurace
                 if (!existing || !dbO) {
                     dbTest.dbNew(dbTest.getDriver());
                     db.init();
                     dbTest.createField();
                 }
                 // inicializuje se tabulka FO
                 db.removeFO();


               GridBagLayout gridBagLayout = new GridBagLayout();
               gridBagLayout.columnWidths = new int[]{110, 286, 0};
               gridBagLayout.rowHeights = new int[]{167, 313, 0};
               gridBagLayout.columnWeights = new double[]{0.2, 0.8, Double.MIN_VALUE};
               gridBagLayout.rowWeights = new double[]{0.5, 0.5, Double.MIN_VALUE};
               setLayout(gridBagLayout);

               JScrollPane scrollPaneRightMenu = new JScrollPane();
               GridBagConstraints gbc_scrollPaneRightMenu = new GridBagConstraints();
               gbc_scrollPaneRightMenu.fill = GridBagConstraints.BOTH;
               gbc_scrollPaneRightMenu.insets = new Insets(0, 0, 0, 0);
```

```
                gbc_scrollPaneRightMenu.gridx = 0;
                gbc_scrollPaneRightMenu.gridy = 0;
                add(scrollPaneRightMenu, gbc_scrollPaneRightMenu);

                 //vytvoření levého horního vstupního contentpanelu
                JPanel panelRightMenu = new JPanel();
                scrollPaneRightMenu.setViewportView(panelRightMenu);
                panelRightMenu.setBorder(new LineBorder(new Color(0, 0, 0), 2));

        panelRightMenu.setComponentOrientation(ComponentOrientation.LEFT_TO_RIGHT);
                 panelRightMenu.setBackground(new Color(143, 188, 143));


                JScrollPane scrollPaneRightInfo = new JScrollPane();
                GridBagConstraints gbc_scrollPaneRightInfo = new GridBagConstraints();
                gbc_scrollPaneRightInfo.fill = GridBagConstraints.BOTH;
                gbc_scrollPaneRightInfo.insets = new Insets(0, 0, 0, 0);
                gbc_scrollPaneRightInfo.gridx = 0;
                gbc_scrollPaneRightInfo.gridy = 1;
                add(scrollPaneRightInfo, gbc_scrollPaneRightInfo);

                 //vytvoření levého dolního content panelu
                JPanel panelRightInfo = new JPanel();
                scrollPaneRightInfo.setViewportView(panelRightInfo);
                panelRightInfo.setBorder(new LineBorder(new Color(0, 0, 0), 2));
                panelRightInfo.setBackground(new Color(143, 188, 143));

                JScrollPane scrollPaneLeftAsk = new JScrollPane();
                GridBagConstraints gbc_scrollPaneLeftAsk = new GridBagConstraints();
                gbc_scrollPaneLeftAsk.fill = GridBagConstraints.BOTH;
                gbc_scrollPaneLeftAsk.insets = new Insets(0, 0, 0, 0);
                gbc_scrollPaneLeftAsk.gridx = 1;
                gbc_scrollPaneLeftAsk.gridy = 0;
                add(scrollPaneLeftAsk, gbc_scrollPaneLeftAsk);

                //vytvoření pravého horního výstupního content panelu
                 //JPanel panelLeftAsk = new JPanel();
                scrollPaneLeftAsk.setViewportView(panelLeftAsk);
                panelLeftAsk.setBorder(new LineBorder(new Color(0, 0, 0), 2));
                panelLeftAsk.setBackground(new Color(205, 133, 63));

                JScrollPane scrollPaneLeftAnswer = new JScrollPane();
                GridBagConstraints gbc_scrollPaneLeftAnswer = new GridBagConstraints();
                gbc_scrollPaneLeftAnswer.fill = GridBagConstraints.BOTH;
                gbc_scrollPaneLeftAnswer.gridx = 1;
                gbc_scrollPaneLeftAnswer.gridy = 1;
                add(scrollPaneLeftAnswer, gbc_scrollPaneLeftAnswer);

                 //vytvoření pravého dolního výstupního content panelu
                //JPanel panelLeftAnswer = new JPanel();
                panelLeftAnswer.setBorder(new LineBorder(new Color(0, 0, 0), 2));
                scrollPaneLeftAnswer.setViewportView(panelLeftAnswer);
                panelLeftAnswer.setBackground(new Color(143, 188, 143));

                 //logiku roztřídění jednotl. částí řídí třída Dispacher
                panelRightMenu.add(new
Dispacher(panelRightMenu,panelRightInfo,panelLeftAsk,panelLeftAnswer));
        }
        public JPanel getLeftAsk() {
            return panelLeftAsk;
        }

        public JPanel getLeftAnsw() {
            return panelLeftAnswer;
        }

        //uložení konfigurace
        public void save() {
            PanelSave panelSave=new PanelSave(panelLeftAnswer);
```

```
            panelLeftAsk.add(panelSave);
            panelLeftAsk.repaint();
            panelLeftAsk.revalidate();
        }

        //smazání pravého dolního výstupního content panelu
        public void getCleanPanel() {
            panelLeftAnswer.removeAll();
            panelLeftAnswer.repaint();
            panelLeftAnswer.revalidate();
        }

        //spuštění ruční migrace
        public void runTO() {

            DbQuery db = new DbQuery();
            Nodes node = new Nodes();
            if (db.emptyFO()) {
            Fo fo = db.getFO();
            db.setTurnFo(true);
            node.setNamecluster(fo.getNameclufo());
                    panelLeftAnswer.removeAll();
                    panelLeftAnswer.add(new RepaintingN(node));
                    panelLeftAnswer.repaint();
                    panelLeftAnswer.revalidate();
            db.setTurnFo(false);
            }
        }

        //spuštění automatické migrace na zákl. failover
        public void runFO() {

            DbQuery db = new DbQuery();
            Nodes node = new Nodes();
            if (db.emptyFO()) {
            Fo fo = db.getFO();
            db.setTurnFo(true);
            node.setNamecluster(fo.getNameclufo());
                    panelLeftAnswer.removeAll();
                    panelLeftAnswer.add(new RepaintingN(node,false));
                    panelLeftAnswer.repaint();
                    panelLeftAnswer.revalidate();
            db.setTurnFo(false);
            }

        }
        //v případě prvního spuštění se vytvoří adresář, kam se uloží databáze a
konfigurační soubory clusteru
        public void createFile() {
           boolean b = false;
           try {
           File saveDir = new File("C:/clderby");
            if(!saveDir.exists()) {
                FileUtils.forceMkdir(saveDir);
                b=true;
            }
           } catch (IOException e){
               System.out.println("Adresar C:/clderby nelze vytvorit");
           }
           }


        public void initDB(){
        DbTest dbTest=new DbTest();
        }
```

A.14 Třída Panel Desktop4

```
/**
 *
 * @author Dagmar Bendova
 */

package dipl2;

import db.Fo;
import javax.swing.JPanel;
import java.awt.Color;
import java.awt.GridBagLayout;
import java.awt.GridBagConstraints;
import java.awt.Insets;
import java.awt.ComponentOrientation;
import org.apache.commons.io.FileUtils;
import javax.swing.JScrollPane;
import javax.swing.border.LineBorder;
import db.*;
import java.io.File;
import java.io.IOException;

public class Desktop4 extends JPanel  {

        final private JPanel panelLeftAnswer;
         final private JPanel panelLeftAsk;
         private Boolean existing = true;

        public Desktop4() {

                panelLeftAnswer = new JPanel();
                panelLeftAsk = new JPanel();

                //pokud databáze neexistuje, vytvoří se adresář, kam se databáze uloží
                createFile();
                DbTest dbTest=new DbTest();
                if (!new File(dbTest.getPath()).exists()) {
                    existing = false;
                }
                //otevření databáze, vyhledání správného driveru
                DbQuery db = new DbQuery();
                boolean dbO = dbTest.dbOpen(dbTest.getDriver());

                //pokud db neexistuje
                //spustí se procedura vytvoření databáze, naplnění dat tabulek databáze
s jedním příkladem clusterové konfigurace
                if (!existing || !dbO) {
                    dbTest.dbNew(dbTest.getDriver());
                    db.init();
                    dbTest.createField();
                }
                // inicializuje se tabulka FO
                db.removeFO();


               GridBagLayout gridBagLayout = new GridBagLayout();
               gridBagLayout.columnWidths = new int[]{110, 286, 0};
               gridBagLayout.rowHeights = new int[]{167, 313, 0};
               gridBagLayout.columnWeights = new double[]{0.2, 0.8, Double.MIN_VALUE};
               gridBagLayout.rowWeights = new double[]{0.5, 0.5, Double.MIN_VALUE};
               setLayout(gridBagLayout);

               JScrollPane scrollPaneRightMenu = new JScrollPane();
               GridBagConstraints gbc_scrollPaneRightMenu = new GridBagConstraints();
               gbc_scrollPaneRightMenu.fill = GridBagConstraints.BOTH;
               gbc_scrollPaneRightMenu.insets = new Insets(0, 0, 0, 0);
```

```
                gbc_scrollPaneRightMenu.gridx = 0;
                gbc_scrollPaneRightMenu.gridy = 0;
                add(scrollPaneRightMenu, gbc_scrollPaneRightMenu);

                 //vytvoření levého horního vstupního contentpanelu
                JPanel panelRightMenu = new JPanel();
                scrollPaneRightMenu.setViewportView(panelRightMenu);
                panelRightMenu.setBorder(new LineBorder(new Color(0, 0, 0), 2));

        panelRightMenu.setComponentOrientation(ComponentOrientation.LEFT_TO_RIGHT);
                 panelRightMenu.setBackground(new Color(143, 188, 143));


                JScrollPane scrollPaneRightInfo = new JScrollPane();
                GridBagConstraints gbc_scrollPaneRightInfo = new GridBagConstraints();
                gbc_scrollPaneRightInfo.fill = GridBagConstraints.BOTH;
                gbc_scrollPaneRightInfo.insets = new Insets(0, 0, 0, 0);
                gbc_scrollPaneRightInfo.gridx = 0;
                gbc_scrollPaneRightInfo.gridy = 1;
                add(scrollPaneRightInfo, gbc_scrollPaneRightInfo);

                 //vytvoření levého dolního content panelu
                JPanel panelRightInfo = new JPanel();
                scrollPaneRightInfo.setViewportView(panelRightInfo);
                panelRightInfo.setBorder(new LineBorder(new Color(0, 0, 0), 2));
                panelRightInfo.setBackground(new Color(143, 188, 143));

                JScrollPane scrollPaneLeftAsk = new JScrollPane();
                GridBagConstraints gbc_scrollPaneLeftAsk = new GridBagConstraints();
                gbc_scrollPaneLeftAsk.fill = GridBagConstraints.BOTH;
                gbc_scrollPaneLeftAsk.insets = new Insets(0, 0, 0, 0);
                gbc_scrollPaneLeftAsk.gridx = 1;
                gbc_scrollPaneLeftAsk.gridy = 0;
                add(scrollPaneLeftAsk, gbc_scrollPaneLeftAsk);

                //vytvoření pravého horního výstupního content panelu
                 //JPanel panelLeftAsk = new JPanel();
                scrollPaneLeftAsk.setViewportView(panelLeftAsk);
                panelLeftAsk.setBorder(new LineBorder(new Color(0, 0, 0), 2));
                panelLeftAsk.setBackground(new Color(205, 133, 63));

                JScrollPane scrollPaneLeftAnswer = new JScrollPane();
                GridBagConstraints gbc_scrollPaneLeftAnswer = new GridBagConstraints();
                gbc_scrollPaneLeftAnswer.fill = GridBagConstraints.BOTH;
                gbc_scrollPaneLeftAnswer.gridx = 1;
                gbc_scrollPaneLeftAnswer.gridy = 1;
                add(scrollPaneLeftAnswer, gbc_scrollPaneLeftAnswer);

                 //vytvoření pravého dolního výstupního content panelu
                //JPanel panelLeftAnswer = new JPanel();
                panelLeftAnswer.setBorder(new LineBorder(new Color(0, 0, 0), 2));
                scrollPaneLeftAnswer.setViewportView(panelLeftAnswer);
                panelLeftAnswer.setBackground(new Color(143, 188, 143));

                 //logiku roztřídění jednotl. částí řídí třída Dispacher
                panelRightMenu.add(new
Dispacher(panelRightMenu,panelRightInfo,panelLeftAsk,panelLeftAnswer));
        }
        public JPanel getLeftAsk() {
            return panelLeftAsk;
        }

        public JPanel getLeftAnsw() {
            return panelLeftAnswer;
        }

        //uložení konfigurace
        public void save() {
            PanelSave panelSave=new PanelSave(panelLeftAnswer);
```

```
            panelLeftAsk.add(panelSave);
            panelLeftAsk.repaint();
            panelLeftAsk.revalidate();
        }

        //smazání pravého dolního výstupního content panelu
        public void getCleanPanel() {
            panelLeftAnswer.removeAll();
            panelLeftAnswer.repaint();
            panelLeftAnswer.revalidate();
        }

        //spuštění ruční migrace
        public void runTO() {

            DbQuery db = new DbQuery();
            Nodes node = new Nodes();
            if (db.emptyFO()) {
            Fo fo = db.getFO();
            db.setTurnFo(true);
            node.setNamecluster(fo.getNameclufo());
                    panelLeftAnswer.removeAll();
                    panelLeftAnswer.add(new RepaintingN(node));
                    panelLeftAnswer.repaint();
                    panelLeftAnswer.revalidate();
            db.setTurnFo(false);
            }
        }

        //spuštění automatické migrace na zákl. failover
        public void runFO() {

            DbQuery db = new DbQuery();
            Nodes node = new Nodes();
            if (db.emptyFO()) {
            Fo fo = db.getFO();
            db.setTurnFo(true);
            node.setNamecluster(fo.getNameclufo());
                    panelLeftAnswer.removeAll();
                    panelLeftAnswer.add(new RepaintingN(node,false));
                    panelLeftAnswer.repaint();
                    panelLeftAnswer.revalidate();
            db.setTurnFo(false);
            }

        }
        //v případě prvního spuštění se vytvoří adresář, kam se uloží databáze a
konfigurační soubory clusteru
        public void createFile() {
           boolean b = false;
           try {
           File saveDir = new File("C:/clderby");
            if(!saveDir.exists()) {
                FileUtils.forceMkdir(saveDir);
                b=true;
            }
           } catch (IOException e){
               System.out.println("Adresar C:/clderby nelze vytvorit");
           }
           }


        public void initDB(){
        DbTest dbTest=new DbTest();
        }

}
```

A.15 Třída Dispacher

```
/**
 * @author Dagmar Bendova
 * Třída Dispacher se stará o obecnou funkční část programu a propojuje interakce na
jednotlivé podněty
 */

package dipl2;

import javax.swing.JPanel;
import javax.swing.UIManager;
import javax.swing.JTree;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.Rectangle;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import javax.swing.tree.DefaultMutableTreeNode;
import javax.swing.tree.DefaultTreeCellRenderer;

public class Dispacher extends JPanel {

   private PanelInfo panelInfo;
   final private JPanel askPanel;
   private JTree tree;
   public JPanel panelAsk;
   public JPanel panelAswr;
   public PanelNew panel1;
   public PanelRemove panelRemove;
   public PanelUpdate panelUpdate;
   public PanelOpen panelOpen;
   public PanelSIPNew panelSIPNew;
   private TreeInfo info;


   //konstruktor třídy se stárá o funkční logiku programu
   public Dispacher(final JPanel tempPanelRightMenu, final JPanel tempPanelRightInfo,
final JPanel tempPanelLeftAsk, final JPanel tempPanelLeftAnswer) {

       super(new GridLayout(1,0));

       UIManager.put("Tree.rendererFillBackground", false);

       //vytvoření menu v levém horním vstupním content panelu
       Tree tempTree = new Tree();
       tree = new JTree(tempTree.createNodes());
       tree.setBackground(new Color(143, 188, 143));

       DefaultTreeCellRenderer renderer = (DefaultTreeCellRenderer)
tree.getCellRenderer();
        renderer.setLeafIcon(null);
        renderer.setClosedIcon(null);
        renderer.setOpenIcon(null);

      //přidání listeneru, který se stará o o interakci od uživatele a reaguje na
vybrání konkrétní položky z menu
      tree.addMouseListener(new MouseListener() {
        @Override
        public void mouseReleased(final MouseEvent e) {}
        @Override
        public void mousePressed(final MouseEvent e) {

          DefaultMutableTreeNode node = (DefaultMutableTreeNode)
tree.getLastSelectedPathComponent();
```

```
          if(tree.getSelectionRows().length!=0){
            final Rectangle r = tree.getRowBounds(tree.getSelectionRows()[0]);

            if (node == null || !r.contains(e.getX(), e.getY())) {
                           return;
                       }
          }

          triggerBookmark(node);
        }
        @Override
        public void mouseExited(final MouseEvent e) {}
        @Override
        public void mouseEntered(final MouseEvent arg0) {}
        @Override
        public void mouseClicked(final MouseEvent arg0) {
        }

        // v případě vybrání konkrétní položky
        private void triggerBookmark(DefaultMutableTreeNode node) {

       if (node == null) return;

       //načte se jaká položka byla uživatelem vybrána
       String substr = node.toString();
       String subs = substr.substring(substr.indexOf("<b>")+3,substr.indexOf("</b>"));
       String nodePath = subs;

       //pokud se jedná o list tzn. poslední prvek v menu
       if (node.isLeaf()) {

           String po = ".htm";
           String pred = "html/";
           String retezec = pred+nodePath+po;
           info.treeURL = info.getURL(retezec);
           //spustí se hml stránka v levém výstupním content panelu a zobrazí help pro
uživatele
           panelInfo.display(info.treeURL);

       } else {
          // panelInfo.display(helpURL);
       }

       panelAsk.removeAll();

       // pokud uživatel zvolí jednu z následujících voleb spustí se požadovaná aktivita
např. vytvoření nové clusterové konfigurace apod.
       if (nodePath.contains("New"))
       {
         panelAsk.add(new PanelNew(tempPanelLeftAnswer));
       }

       if (nodePath.contains("Remove"))
       {
         panelAsk.add(new PanelRemove(tempPanelLeftAnswer));
       }

       if (nodePath.contains("Update"))
       {
         panelAsk.add(new PanelUpdate(tempPanelLeftAnswer));
       }

       if (nodePath.contains("Open"))
       {
         panelAsk.add(new PanelOpen(tempPanelLeftAnswer));
       }

       if (nodePath.contains("Service IP"))
       {
```

```
          panelAsk.add(new PanelSIPNew(tempPanelLeftAnswer));
        }

        if (nodePath.contains("Boot IP"))
        {
          panelAsk.add(new PanelBIPNew(tempPanelLeftAnswer));
        }

        if (nodePath.contains("Heartbeat"))
        {
          panelAsk.add(new PanelHBNew(tempPanelLeftAnswer));
        }

        if (nodePath.contains("Resource group"))
        {
          panelAsk.add(new PanelAplNew(tempPanelLeftAnswer));
        }

        panelAsk.repaint();
        panelAsk.revalidate();
            }

       });

       //nastavení úvodní stránky z helpem.
       panelInfo=new PanelInfo();
       panelInfo.setOpaque(false);
       info = new TreeInfo();
       info.treeURL = info.getURL("html/Uvod.htm");
       panelInfo.display(info.treeURL);
       tempPanelRightInfo.add(panelInfo);

        askPanel=new JPanel();
        askPanel.setBackground(new Color(205, 133, 63));
        askPanel.setPreferredSize(new Dimension(600,500));
        UIManager.put("TabbedPane.contentOpaque", Boolean.FALSE);
           askPanel.setOpaque(false);

        //přidání JMenu do levého horního vstupního content panelu
        add(tree);
            setPanel(tempPanelLeftAsk,tempPanelLeftAnswer);
    }

    private void setPanel(JPanel panelAsk, JPanel panelAswr) {
        this.panelAsk = panelAsk;
        this.panelAswr = panelAswr;
    }
}
```

A.16 Třída RepaintingN

```
/**
 *
 * @author Dagmar Bendova
 *  Třída RepaintíngN realizuje grafický výstup pravého dolního výstupního content
panelu
 */

package dipl2;

import db.*;
import java.util.Iterator;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.Query;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;

public class RepaintingN extends javax.swing.JPanel {


    private List<Pair> listOfPairs;
    private PanelAplTable panelAplTable;
    private char left;
    private char right;

    public RepaintingN() {
        left = 'A';
        right = 'B';
        initComponents();
    }

    //konstruktor realizuje překreslení všech prvků GUI
    public RepaintingN(Nodes node) {
        this();
        initCustom(node,true);
    }

    //konstruktor realizuje překreslení prvků GUI, které se netýkají resource groupy
    public RepaintingN(Nodes node, boolean b) {
        this();
        initCustom(node,b);
    }

        public void initCustom(Nodes node,boolean b) {

        PanelBasicDiscNfo pbdLeft;
        PanelBasicDiscNfo pbdRight;
        DbQuery dbQ = new DbQuery();

        List<Nodes> listNode =  dbQ.getNode(node.getNamecluster());

        Iterator itr = listNode.iterator();
      while(itr.hasNext()) {
         Nodes element = (Nodes)itr.next();
         if (element.getTypnode() == 'A') {
             jLabel2.setText(element.getNamenodes());
             node.setNamenodes(element.getNamenodes());
             node.setTypnode(left);
             //pbdLeft = new PanelBasicDiscN(node);
             pbdLeft = new PanelBasicDiscNfo(node,b);
            jPanel5.add(pbdLeft);
         }
```

```
         if (element.getTypnode() == 'B') {
             jLabel3.setText(element.getNamenodes());
             node.setNamenodes(element.getNamenodes());
             node.setTypnode(right);
             //pbdRight = new PanelBasicDisc();
             //pbdRight = new PanelBasicDiscN(node);
             pbdRight = new PanelBasicDiscNfo(node,b);
             jPanel6.add(pbdRight);
         }
       }

         jLabel1.setText(node.getNamecluster());
         jLabel1.setSize(jLabel1.getPreferredSize());
         jLabel1.invalidate();

         jLabel2.setSize(jLabel2.getPreferredSize());
         jLabel2.invalidate();

         jLabel3.setSize(jLabel3.getPreferredSize());
         jLabel3.invalidate();

     }

     //tato funkce realizuje naplnění a překreslení resource groupy
     public JTable repaintApl(String nameCluster) {

     JTable jTable1 = new JTable();
     //        while (panelAplTable.jTable1.getRowCount() > 0) {
     //           ((DefaultTableModel) panelAplTable.jTable1.getModel()).removeRow(0);
    //     }

     EntityManagerFactory emf = Persistence.createEntityManagerFactory("dipl2PU");
     EntityManager em = emf.createEntityManager();
     em.getTransaction().begin();

     Query query = em.createNativeQuery(
                     "Select * FROM Apl WHERE NameCluster=?",Apl.class);
     query.setParameter(1, nameCluster);
     List<Nodes> results = query.getResultList();

     Iterator itr = results.iterator();
     Object[] row = new Object[4];
     int i=0;
       while(itr.hasNext()) {
          Apl element = (Apl)itr.next();
          row[0] = element.getNameapl();
          row[1] = element.getIp();
          row[2] = element.getLv();
          row[3] = element.getFs();

          ((DefaultTableModel) jTable1.getModel()).insertRow(i,row);
          i++;
       }
      return jTable1;
     }

     //tato funkce realizuje naplnění a překreslení HB
     public JTable repaintHB(Nodes node) {

     JTable jTable1 = new JTable();
     EntityManagerFactory emf = Persistence.createEntityManagerFactory("dipl2PU");
     EntityManager em = emf.createEntityManager();
     em.getTransaction().begin();

     Query query = em.createNativeQuery(
                     "Select * FROM Heartbeat WHERE NameCluster=? AND
Namenode=?",Heartbeat.class);
     query.setParameter(1, node.getNamecluster());
     query.setParameter(2, node.getNamenodes());
```

```
    List<Nodes> results = query.getResultList();

    Iterator itr = results.iterator();
    Object[] row = new Object[2];
    int i=0;
      while(itr.hasNext()) {
         Heartbeat  element = (Heartbeat)itr.next();
         row[0] = element.getTypehb();
         row[1] = element.getValuehb();

         ((DefaultTableModel) jTable1.getModel()).insertRow(i,row);
         i++;
      }
     return jTable1;
    }

    //grafická část pro zobrazení panelu RapaintingN
    //kod automaticky generovaný programem Netbeans na základě návrhu v design složce
knihovny SWING
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        jPanel2 = new javax.swing.JPanel();
        jPanel5 = new javax.swing.JPanel();
        jPanel6 = new javax.swing.JPanel();
        jPanel7 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();

        setAlignmentX(0.0F);
        setAlignmentY(0.0F);

        jPanel2.setAlignmentX(0.0F);
        jPanel2.setAlignmentY(0.0F);
        jPanel2.setLayout(new java.awt.GridBagLayout());

        jPanel5.setBackground(new java.awt.Color(143, 188, 143));
        jPanel5.setAlignmentX(0.0F);
        jPanel5.setAlignmentY(0.0F);
        jPanel5.setMinimumSize(new java.awt.Dimension(120, 120));
        jPanel5.setName(""); // NOI18N
        jPanel5.setLayout(new java.awt.GridLayout(1, 0));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 1;
        gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
        gridBagConstraints.weightx = 0.6;
        gridBagConstraints.weighty = 0.9;
        jPanel2.add(jPanel5, gridBagConstraints);

        jPanel6.setBackground(new java.awt.Color(143, 188, 143));
        jPanel6.setAlignmentX(0.0F);
        jPanel6.setAlignmentY(0.0F);
        jPanel6.setPreferredSize(new java.awt.Dimension(120, 120));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 1;
        gridBagConstraints.gridy = 1;
        gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
        gridBagConstraints.weightx = 0.4;
        gridBagConstraints.weighty = 0.9;
        jPanel2.add(jPanel6, gridBagConstraints);

        jPanel7.setBackground(new java.awt.Color(143, 188, 143));
        jPanel7.setAlignmentX(0.0F);
        jPanel7.setAlignmentY(0.0F);
        jPanel7.setLayout(new org.Netbeans.lib.awtextra.AbsoluteLayout());
```

```
        jLabel1.setBackground(new java.awt.Color(0, 204, 204));
        jLabel1.setForeground(new java.awt.Color(0, 0, 102));
        jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel1.setBorder(javax.swing.BorderFactory.createMatteBorder(8, 15, 8, 15, new
javax.swing.ImageIcon(getClass().getResource("/dipl2/jpg/CPU1.jpg")))); // NOI18N
        jLabel1.setOpaque(true);
        jPanel7.add(jLabel1, new org.Netbeans.lib.awtextra.AbsoluteConstraints(260, 10,
150, 30));

        jLabel2.setBackground(new java.awt.Color(0, 204, 204));
        jLabel2.setForeground(new java.awt.Color(0, 0, 102));
        jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel2.setBorder(javax.swing.BorderFactory.createMatteBorder(8, 15, 8, 15, new
javax.swing.ImageIcon(getClass().getResource("/dipl2/jpg/CPU1.jpg")))); // NOI18N
        jLabel2.setOpaque(true);
        jPanel7.add(jLabel2, new org.Netbeans.lib.awtextra.AbsoluteConstraints(100, 50,
140, 30));

        jLabel3.setBackground(new java.awt.Color(0, 204, 204));
        jLabel3.setForeground(new java.awt.Color(0, 0, 102));
        jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel3.setBorder(javax.swing.BorderFactory.createMatteBorder(8, 15, 8, 15, new
javax.swing.ImageIcon(getClass().getResource("/dipl2/jpg/CPU1.jpg")))); // NOI18N
        jLabel3.setOpaque(true);
        jPanel7.add(jLabel3, new org.Netbeans.lib.awtextra.AbsoluteConstraints(430, 50,
130, 30));

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.gridwidth = 2;
        gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
        gridBagConstraints.weightx = 0.5;
        gridBagConstraints.weighty = 0.1;
        jPanel2.add(jPanel7, gridBagConstraints);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE, 665,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(0, 0, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE, 352,
javax.swing.GroupLayout.PREFERRED_SIZE)
        );
    }// </editor-fold>


    // Variables declaration - do not modify
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel5;
    private javax.swing.JPanel jPanel6;
    private javax.swing.JPanel jPanel7;
    // End of variables declaration
```