VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

GRAFICKÉ UŽIVATELSKÉ PROSTŘEDÍ V JAVAFX

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

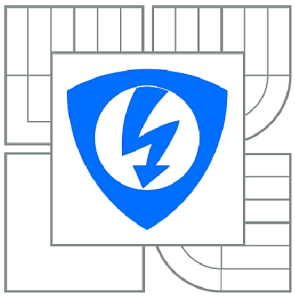
AUTOR PRÁCE
AUTHOR

ZDENĚK FUSEK

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

GRAFICKÉ UŽIVATELSKÉ PROSTŘEDÍ V JAVAFX

GRAPHICAL USER INTERFACE IN JAVAFX

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

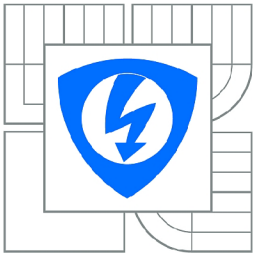
AUTOR PRÁCE — ZDENĚK FUSEK

AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN KARÁSEK

SUPERVISOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student: Zdeněk Fusek ***ID:*** 136512
Ročník: 3 ***Akademický rok:*** 2012/2013

NÁZEV TÉMATU:

Grafické uživatelské prostředí v JavaFX

POKYNY PRO VYPRACOVÁNÍ:

V rámci bakalářské práce student navrhne modulární jádro aplikace, které bude možné rozšiřovat libovolnou funkčností zásuvnými moduly. Modulární jádro student implementuje v programovacím jazyce JAVA. Dále student navrhne a implementuje grafické uživatelské prostředí tak, aby byla aplikace jednoduše a intuitivně ovladatelná. Grafické uživatelské prostředí student implementuje v programovacím jazyce JavaFX. Implementovaná aplikace bude prezentována jako jednoduchý výpočetní systém.

DOPORUČENÁ LITERATURA:

[1] Weaver, James L. et al. Pro JavaFX™ Platform: Script, Desktop and Mobile RIA with Java™ Technology. Apress, 1st edition. 586 s. 2009. ISBN 978-1430218753.

[2] Jordan, L. JavaFX Special Effects: Taking Java™ RIA to the Extreme with Animation, Multimedia, and Game Elements (Beginning). Apress, 1st edition. 300 s. 2009. ISBN 978-1430226239.

Termín zadání: 11.2.2013 ***Termín odevzdání:*** 5.6.2013

Vedoucí práce: Ing. Jan Karásek
Konzultanti bakalářské práce:

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Cílem této bakalářské práce je vytvoření jednoduchého výpočetního systému. V první části bude zpracována rešerše týkající se porovnání RIA platforem, jejímž výsledkem je analýza výhod a nevýhod jednotlivých platforem. Z důvodu zaměření práce bude platformě JavaFX věnován širší prostor k jejímu podrobnému popisu a možnostem využití.

K vytvoření aplikace, jež je prezentována jako jednoduchý výpočetní systém je zapotřebí nejprve vytvořit modulární jádro, následně navrhnout uživatelské rozhraní předepisující základní vzhled celé aplikace. Posléze dojde k vytvoření funkce drag & drop, pomocí které budou přenášeny moduly aritmetických operací a jejich vstupů. Po propojení vstupů s aritmetickou operací je možné vypočítat výsledek. Vytvoření výpočetního systému je podrobně popsáno v praktické části.

Bakalářská práce bude implementována ve vývojářském prostředí e(fx)clipse, a ke stylování práce budou použity CSS kaskádové styly.

KLÍČOVÁ SLOVA

JavaFX, RIA, Modulární systém, GUI

ABSTRACT

The aim of of this thesis is creation of the simple computing system. In the first part will be processed research, which involves comparison of RIA platforms. The result of research is the analysis of advantages and disadvantages of each platform. Due to orientation of the thesis, JavaFX platform will be described in detail in a separate chapter.

For creating the application, which is presented as a simple computing system, is necessary create modular kernel at first, then propose a user interface, which prescribes the basic look of the application. Subsequently is created function drag & drop. Modules of arithmetic operations and their input will be transmitted by using this function. Calculation of the result is possible after connecting inputs with the arithmetic operations. Creation of computing system is described in detail in the practical part of thesis.

Bachelor thesis is implemented in a development environment e(fx)clipse, and CSS Cascading Style Sheets are used for the styling of thesis.

KEYWORDS

JavaFX, RIA, Modular system, GUI

FUSEK, Zdeněk *Grafické uživatelské prostředí v JavaFX*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2013. 46 s. Vedoucí práce byl Ing. Jan Karásek

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Grafické uživatelské prostředí v JavaFX“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Janu Karáskovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

(podpis autora)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této bakalářské práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .

(podpis autora)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Předložená bakalářská práce je zaměřena na zpracování teoretických údajů z oblasti RIA [1], JavaFX, GUI [2] a zásuvných modulů, a následné využití těchto poznatků v praktické části, kde dojde k vytvoření aplikace prezentované jako jednoduchý výpočetní systém.

Dokument je rozdělen do dvou částí. Teoretická část je zaměřena na vysvětlení pojmu RIA, jeho historii a vyjmenování hlavních vývojových platforem, které budou následně podrobněji charakterizovány, a zhodnoceny jejich kladné a záporné stránky. Z důvodu zaměření práce, se další kapitola této části důkladně zabývá platformou JavaFX. Je zde zmíněna historie, technologické možnosti, klíčové vlastnosti této platformy, výhody pro vývojáře, a přednosti oproti jiným srovnatelným technologiím. Poté je vysvětlena problematika zásuvných modulů a grafického uživatelské rozhraní.

Na základě získaných teoretických poznatků byl vytvořen jednoduchý výpočetní systém, který je podrobně popsán v praktické části. Aby byl princip fungování aplikace lépe pochopen, je popis obohacen o obrázky aplikace, ukázky zdrojových kódů, životních cyklů a diagramů tříd. Nejprve je charakterizováno jádro vytvořené modulární aplikace a jeho komunikace se zásuvnými moduly. Posléze je stručně popsáno navržené uživatelské rozhraní, jehož jednotlivé komponenty jsou detailně popsány v další kapitole. Praktická část se také věnuje funkci drag & drop a stručně popisuje balíčky se třídami, z nichž se aplikace skládá. V poslední kapitole je popsána funkčnost celkové aplikace.

[1] bohaté internetové aplikace – Rich Internet Applications.
[2] grafické uživatelské rozhraní – Graphical User Interface.

1 TEORETICKÁ ČÁST

Tato část se zabývá teoretickými poznatky z oblasti bohatých internetových aplikací, zásuvných modulů a grafického uživatelského prostředí.

1.1 Bohaté internetové aplikace

Bohaté internetové aplikace (RIA) byly vždy o požitkovém vnímání uživatele. Jsou to webové aplikace, které nesou mnoho rysů desktopových aplikací a jsou dodávány uživateli pomocí zásuvných modulů přes webový prohlížeč, nezávisle přes sandboxy (bezpečnostní mechanizmy obvykle používané pro spuštění neověřených zdrojových kódů nebo programů) nebo také skrz virtuální počítače. Termín RIA skýtá nespočet různých definicí, jejichž počet roste spolu s rozvíjející se internetovou komunitou, ale všechny tyto definice spojuje jediné, snaha o zlepšení celkového zážitku uživatele v různých směrech. RIA přeměňuje zpracování, které je pro uživatelské rozhraní web klienta nezbytné, ale udržuje převážnou část dat (jako je zachování stavu programu, udržení jeho dat a podobně) v aplikačním serveru, tak mohou nabídnout lepší požitek a neustále tím posouvají hranice toho, co je od internetového vyhledávače uživateli očekáváno [1].

V dobách, kdy byl internet na počátku své existence, se ve vyhledávači objevoval statický, ničím nezajímavý prostý text. To se ale poslední dobou prudce změnilo, nyní hraje dynamický obsah ve webových aplikacích důležitou roli. RIA technologie způsobila obdobnou revoluci na výpočetní straně klienta. Dá se říci, že uživatele nyní práce více baví, protože je jednodušší a více přístupná. Někdo může vidět RIA jako utužování vztahů mezi uživatelskými rozhraními určené pro desktopy a web, a tím i usnadnění dodání jednotného uživatelského požitku skrze platformy, zařízení a prohlížeče. Pod pojmem „Bohatý“ v kontextu RIA si lze představit pohodlné, atraktivní a oslnivé zážitky uživatele. Tímto vším ale zastoupení RIA nekončí, lze je najít i v aplikacích pro mobilní zařízení. Na obr. 1.1 jsou zobrazeny hlavní oblasti, které RIA pokrývá [1].

1.1.1 Historie

Koncept byl uveden v březnu 2002. Tehdy bylo cílem nabídnout vylepšené uživatelské zážitky, které nejsou nijak závislé na technologii [1].

Zprostředkovat uživateli nevšední zážitek se v dnešní době snaží mnoho platforem. Mezi ty největší a prominentní patří bezesporu platforma Adobe Flash. Nicméně v posledních letech bylo provedeno několik výzkumů a vývojů podílejících se na růstu technologií, které web rozvíjejí, a zlepšení standardů podporující nejnovější

Obr. 1.1: Hlavní oblasti rozsahu RIA aplikací, vlastní zpracování dle zdroje [15].

webové prohlížeče. Tímto bylo podpořeno mnohem více technologií sloužící k vývoji webových aplikací při vstupu na trh. Patří mezi ně např. AJAX (Asynchronní JavaSkript a XML), Adobe Flex, Microsoft Silverlight, Mozilla Prism, JavaFX, atd. Pokud jsou však tyto technologie zkoumány z pohledu vývojové báze, pouze pár z nich, např. Adobe Flex, Silverlight a JavaFX se může stavět do pozice plnohodnotných vývojových platforem pro RIA [1].

1.1.2 RIA platformy

V dnešním světě existuje nespočetné množství vývojových platforem, ale pouze několik z nich jsou pro RIA smysluplné a představují pro ně hlavní hráče.

1. **Adobe Flex** je programovací jazyk vyvinutý Adobe technologií za účelem rozšířit uživatelské možnosti při budování bohatých internetových aplikací. Adobe Flex se skládá ze dvou programovacích jazyků - MXML a Action Script [23]. MXML je deklarativní jazyk založený na XML, umožňuje používání českého jazyka a používá se pro tvorbu uživatelského rozhraní. Action Script slouží k programování pro Adobe Flash Player [29].

   Nejprve je nutné vyvinout zdrojový kód, ten je možné vytvořit jak v Adobe Flash Professional, tak v Adobe Flash Builder. Adobe Flash Professional je prostředí pro vytváření animací a multimediálních obsahů. V tomto prostředí lze také navrhnout interaktivní obsah, který má stejnou podobu pro mobilní, televizní a počítačová zařízení [30]. Adobe Flash Builder je vývojářský nástroj pro mobilní, webové a počítačové aplikace. Umožňuje vytvářet pomocí profesionálních nástrojů výkonné aplikace [31]. Jakmile je tento zdrojový kód vytvořený, lze jej vidět buď pomocí Adobe Flash Playeru v internetovém prohlížeči, nebo může být zobrazený jako počítačová aplikace prostřednictvím

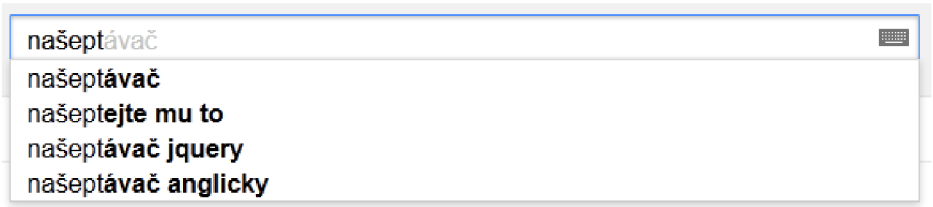
Adobe AIR [5].

(a) **Výhody:** Adobe Flex využívá stejné běhové prostředí jako Adobe Flash player, který je dostupný téměř na každém počítači. Další výhodou je možnost spuštění aplikace mimo prohlížeč v offline režimu nebo využití předpřipravených knihoven a komponent [33].

(b) **Nedostatky:** Jako první můžeme uvést nedostatek podpory pro mobilní zařízení [5]. Dalším nedostatkem je vysoká cena licenčních poplatků nebo to, že zdrojový kód vytvořený technologii Adobe Flex není přeložen do strojové podoby.

2. **Ajax (jQuery):** jQuery je knihovna JavaScriptu, která byla vyvinuta, aby zjednodušila metodu psaní JavaScript kódu. Pokud bychom vynechali platformu jQuery a chtěli implementovat zdrojový kód, při použití čistého JavaScriptu by bylo potřeba několik desítek řádků, při použití jQuery se z těchto desítek stane pouze jeden [3].

   Ajax znamená anglicky Asynchronous JavaScript and XML (asynchronní JavaScript a rozšiřitelný značkovací jazyk) a je funkcí jQuery [2]. Tato technologie umožňuje posílat data na server a taky je přijímat bez nutnosti opětovného načítání, jak tomu bývá klasicky. Ajax bývá využíván především pro realizaci tzv. našeptávačů. Při psaní do textového pole vyhledávače se text průběžně odesílá na server a zpět je přijímána nápověda zobrazující varianty slov, které by uživatel mohl zadat. Jako příklad lze uvést společnost Google, která začala Ajax využívat v dynamickém vyhledávání (obr.1.2) a Gmailu. Podobnou funkci jako Google Suggest(Google našeptávání) u nás nabízí např. Seznam [28].



Obr. 1.2: Google našeptávač, vlastní zpracování.

(a) **Výhody:** Mezi hlavní výhody patří rychlost běhu uživatelského rozhraní internetových a intranetových aplikací. Tato neobvyklá rychlost je zapříčiněna zejména tím, že po každé akci není potřeba znovu načítat obsah stránky. Z pohledu uživatele je tedy funkcionalita zaměnitelná s klasickými počítačovými aplikacemi. Další velkou výhodou je úspora přenosové kapacity, protože se neposílá pokaždé celý kód stránky. Jako pří-

klad můžeme uvést dynamickou stránku, při které se posílá pouze obsah části stránky, která se pravidelně mění. Uživatel tím získá dojem plynulé práce [32].

(b) **Nedostatky:** Právě zde se nejvíce projeví skutečnost, že nejde o novou technologii, ale jenom o nadstavbu existujících technologií, které s sebou nesou mnohá negativa. Například protokol HTTP není navrhnut na intenzivní komunikaci mezi serverem a klientem (při každém požadavku se musí navázat spojení) a jeho velkým omezením je, že nemůže v případě potřeby kontaktovat uživatele [4]. Kromě toho, Ajax aplikace představují pro vývojáře značné nepříjemnosti – bývají lehce „napadnutelné“, a tím se dostávají na internetové stránky, kde jsou volně ke stažení [5].

3. **Microsoft Silverlight** patří mezi nejmodernější technologie internetových prohlížečů. Tato platforma se používá se při tvorbě dynamického online obsahu, kde kombinuje text, bitmapovou grafiku, video a animace [6]. Aby se platformě Silverlight podařilo překročit meze standardních webových stránek, používá k tomu dobře známou techniku – odlehčený zásuvný modul (lightweight-plug-in) prohlížeče. Silverlight je vyvinutý společností Microsoft a mezi jeho přímé konkurenty patří AdobeFlash. Oba dva tito konkurenti jsou si podobní, co se týče vytváření interaktivního obsahu [7]. Microsoft Silverlight je podporou pro většinu současných webových prohlížečů (Internet Explorer, Mozilla Firefox, Safari, Opera, Google Chrome) na platformách Windows a Mac OS X. Je dostupný i na Linuxu a to pod názvem Moonlight [6].

(a) **Výhody:** Balíček má malou velikost (asi 4 MB), což znamená, že lze rychle získat z webu distributora a snadno nainstalovat. Samotná instalace je stejně bezproblémová jako u konkurenta Adobe Flash [34].

Další výhodou je 3D grafika, jež umožňuje implementovat 3D obsah. Uživatelé tak mohou s tímto obsahem manipulovat, přibližovat ho nebo oddalovat v prostoru [35].

Technologie Silverlight se také využívá jako platforma pro vývoj aplikací běžících na operačním systému Windows Phone 7, například sledování videa a poslouchání audia ve vysoké kvalitě nebo funkce Deep Zoom, ta uživatelům zprostředkovává lepší zážitek z prohlížení textů a fotografií [36].

Nejvíce působivou výhodou je zahrnutí zredukované, prakticky kompletní verze CLR s vyčerpávající sadou tříd jádra, tzv. garbagecollectorem, kompilátorem JIT (just-in-time), podporou generiky atd. Pro vývojáře, kteří vytvoří zdrojový kód pro CLR .NET je tato výhoda obzvlášť

příjemná, mohou totiž tento zdrojový kód po mírných úpravách použít i v aplikaci Silverlight [34].

(b) **Nedostatky:** Silverlight je poměrně nová technologie, rychle se vyvíjí, a to podle firemních vývojářů způsobuje nízkou propracovanost bloků. Ti se pak musí při své práci spoléhat na bohatě vybavenou knihovnu. Silverlight postrádá jakýkoli druh funkcí pro vázání dat a zahrnuje také poměrně málo hotových, rovnou použitelných ovládacích prvků. Silverlight neobsahuje ani některé základní věci, jako jsou např. tlačítka, které je nutné si vytvořit samostatně. Nicméně tlačítka patří mezi snadno vytvořitelné objekty, textová pole se svépomocí budují obtížně. Z tohoto důvodu mají o Silverlight zájem převážně ti vývojáři, kteří vytvářejí graficky náročné, kompletně přizpůsobené uživatelské rozhraní, a kterým nevadí, že nad ním stráví poměrně dlouhou dobu [7].

4. **JavaFX** je jednou z dalších platforem na trhu. Z důvodu zaměření práce bude platformě JavaFX věnováno více prostoru v následujících kapitolách.

1.2 Platforma JavaFX

Je bohatá klientská platforma pro vytváření cross-device aplikací, tj. aplikací běžících na více různých zařízeních. Byla navržena pro tvorbu bohatých internetových aplikací a měla konkurovat známějším platformám Adobe Flash nebo Microsoft Silverlight [37]. Základem je skriptovací jazyk, který zobrazuje rozhraní pro jednotlivé aplikace a také je v případě potřeby animuje. Slouží například webovým obsahům k renderování animací, což lze nazvat také jako tvorba reálného obrazu animace, a přehrávání filmů. Videa a filmy spuštěné na platformě JavaFX jsou přehrávány s VP6 kodekem od On2, který je elementem JavaFX běhového prostředí. Kvalita přehrávání je srovnatelná s DivX 6, pro uživatele tedy plně dostačující [8].

1.2.1 Historie

JavaFX byla známá jako projekt Christophera Olivera pod jménem F3, což v anglickém jazyce znamená FormFollowFunction. Christopher Oliver byl softwarový inženýr společnosti Sun Microsystems. V roce 2007 na konferenci JavaOne, byl projekt F3 oficiálně spuštěn jako platforma, která od té doby měla interpretově založený jazyk. V červenci o rok později zahájila firma Sun Microsystems první ukázkovou verzi JavaFX s vlastním kompilátorem (překladačem). První verze platformy, tedy JavaFX 1.0, byla vypuštěna v prosinci ještě téhož roku a byla vybavena mnoha dalšími vylepšeními a optimalizacemi. JavaFX 1.1 byla spuštěna v únoru

roku 2009. Tato novější verze byla primárně zaměřena na mobilní telefony. Tímto krokem se JavaFX stala plně funkční pro mobilní zařízení, jak ukázala při příležitosti Mobile WorldConference v únoru téhož roku. Společnost Sun Microsystems ale nezahálela a dále pokračovala ve vylepšování. Přidáváním více rysů, optimalizací a zlepšením výkonu vznikla v roce 2009 JavaFX verze 1.2. Tato verze byla ukázána při příležitosti konference JavaOne, která se konala tentýž rok [1].

V roce 2010 nastal pro společnost Sun Microsystems velký zlom, když byla koupena firmou Oracle. Od tohoto okamžiku pracovali hardwaroví a softwaroví inženýři společnými silami na vybudování plně integrovaného a optimalizovaného systému. Jejich cílem bylo dosáhnout úrovně výkonu, která byla v té době a v tomto průmyslu zcela nevídaná a bezkonkurenční. Kolektivně tak vytvořili JavaFX verze 1.3 [9].

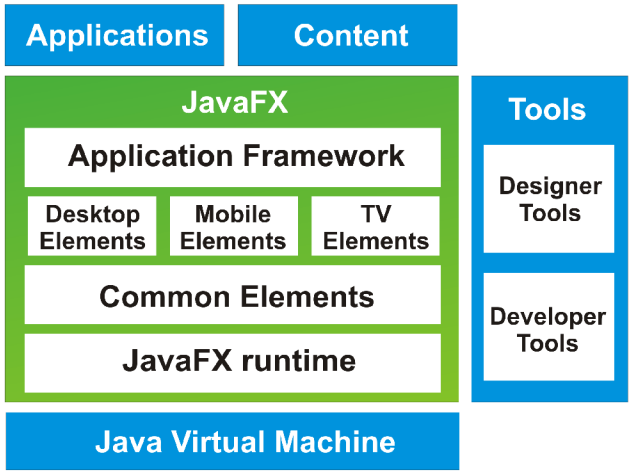
Postupem času vznikaly stále novější verze JavaFX. V roce 2011 byla vypuštěna verze 2.0 a v letošním roce 2012 se uživatelé mohou těšit z verze 2.1, jež zavádí podporu pro přehrávání medií ve formátu MPEG4. Dalším vylepšením je vykreslování sub-pixel rendering pro moderní LCD displeje, podpora JavaSriptu. Tato verze je zatím k dispozici pouze pro Windows a Mac OS X [21]. Úplně nejnovější verze 2.2 zajišťuje plnou podporu pro Linux s 64 bitovým nebo x86 bitovým systémem. Příjemnou novinkou verze 2.2 je Multi-touch podpora, která je využívána u dotykových zařízení a touchpodech osobních počítačů [20].

1.2.2 Technologické možnosti

JavaFX je velmi specifická technologie s jasně určenými cíli. Na obr.1.3 je ukázáno, z jakých částí se platforma skládá.

- **JavaFX Runtime** – Běhové prostředí obsahuje potřebné knihovny sloužící k funkci programů. JavaFX využívá běhového prostředí platformy Java. Pro zařízení obsahující platformu Java stačí instalovat pouze malý objem dat, aby bylo s technologií JavaFX možné pracovat. Pokud ovšem platforma na zařízení instalována není, je nutná její kompletní instalace, což bývá zdlouhavé a pro zákazníky odrazující [38].
- **Common Elements** – Zde je aplikační rozhraní (Application Interface, API), které je společné pro všechny platformy [18]. Obsahuje knihovny základních tvarů, animací, ovládacích prvků a multimédií, které lze využít pro mobilní i desktopovou platformu [38].
- **Desktop Elements** – V této částí se nachází API specifické pro desktopovou platformu [18]. Rozšiřuje knihovny základních tvarů například o knihovny efektů a reflexních funkcí [38].
- **Mobile Elements** – Tato část obsahuje API pro mobilní zařízení [18].

- **TV Elements** – Zde nalezneme API pro televizní platformu [18].
- **Application Framewor** – Application Framework lze nazvat jako základní stavební prvek, který slouží pro vývoj aplikací [18].
- **Designer Tools** – Tento balíček obsahuje zásuvné moduly pro grafické nástroje např. Adobe Photoshop. Programátor nepotřebuje znát JavaFX, jednoduše si navrhne grafické prostředí a exportuje ho ve formátu FXZ. Formát FXZ obsahuje také soubor FXD, který je psán v JavaFX syntaxi. Importováním FXZ získá grafické objekty, které může používat [18].
- **Developer Tools** – Z vývojových nástrojů, jež JavaFX nabízí, jsem vyzkoušel e (fx) eclipse a podle mého názoru poskytuje dostatečné pohodlí pro vývoj JavaFXaplikací.



Obr. 1.3: Technologie JavaFX, vlastní zpracování dle zdroje [14].

1.2.3 Implementace

Skriptovací jazyk JavaFX se stává doopravdy silným teprve s propojením na rozsáhlý programovací jazyk Java. Ve srovnání s tímto řešením je AIR s ActionScriptem značně pozadu. Silverlight 2 má ovšem v té době co nabídnout. K dispozici jsou zde programovací jazyky jako VB. NET a C#. Nástroje pro vývojáře JavaFX poskytuje Oracle bezplatně. Mezi tyto nástroje patří například e (fx) eclipse, což je vývojářské prostředí [8].

1.2.4 Spuštění platformy

V roce 2009 společnost Sun Microsystems poskytla JavaFX svůj vlastní programovací jazyk a tím nabídla vývojářům nástroje sloužící pro vytváření zdrojových kódu.

Pokud jde o samotné spuštění RIA aplikace v JavaFX, jediné, co uživatelé potřebují, je programovací knihovna, která spustí běhové prostředí. Java od společnosti Sun Microsystems nemá při distribuci obtíže, protože už v té době je k dispozici na 92% všech počítačů, tím pádem většina uživatelů má JavaFX ve svém zařízení nainstalovanou. V polovině října Sun Microsystems distribuoval Java Runtime Engine 6 Update 10, který obsahuje JavaFX. Pro Sun Microsystems byla toto velká výhra a prostředek k expanzi podílu na trhu. Od jiných firem jako například Microsoft a Adobe museli uživatelé tyto zdrojové kódy teprve dodatečně stahovat. Dalším plusem bylo, že platforma Java je v té době již k dispozici na mnoha mobilních zařízeních. Na druhé straně JavaFX silně pokulhává ve vývoji platformy a aplikací. Při zobrazení internetového obsahu mimo vyhledávač a pro multimédia musí společnost Sun Microsystems začít od nuly [8].

1.2.5 Klíčové vlastnosti

- Plná integrace s JDK 7. V době uvedení JavaFX SDK 2.2 a Java SE 7 aktualizace 6, je JavaFX SDK plně integrována s Javou SE 7 Runtime Environment (JRE) a DevelopmentKit (JDK) [10].
- API byly navrženy tak, aby byly přátelské k různým variantám JVM jazyků, stejně tak jako tomu je u Scala a JRuby. Scala je univerzální programovací jazyk navržený pro vyjádření běžných programových schémat stručným, elegantním, typově bezpečným způsobem. Hladce integruje vlastnosti objektově orientovaných a funkčních jazyků, což umožňuje Java programátorům být více produktivní [24]. JRuby je integrován s platformou Java a je interpretem dynamicky napsaného, objektově orientovaného programovacího jazyka Ruby [25]. Vzhledem k tomu, že JavaFX funkce jsou k dispozici prostřednictvím rozhraní Java API, mohou uživatelé i nadále používat své oblíbené Java vývojářské nástroje (např. IDE – integrované vývojové prostředí, kód refactoring – technika pro zlepšení struktury stávajícího kódu beze změny pozorovatelného chování [26] a debuggery – program umožňující zastavení běhu programu po každém jeho vykonaném řádku nebo na předem určené zarážce [27]) k rozvíjení JavaFX aplikací.
- FXML, deklarativní značkovací jazyk, který je založen na XML a slouží k definování uživatelského rozhraní v aplikaci JavaFX. Je to poměrně jednoduchý jazyk, který nevyžaduje rekompilovat zdrojový kód pokaždé, když je provedena změna v rozvržení [10].
- Podpora přehrávání webového multimediálního obsahu. To přináší stabilitu mediálního rámce, který je založen na multimediálním rámci GStreamer [10].
- Široká škála prvků vestavěných na uživatelském rozhraní, která zahrnuje grafy,

tabulky, nabídky a podokna. Příjemnou změnou pro uživatelskou komunitu je, že API umožňuje třetím stranám, aby těmito ovládacími prvky přispívaly [10].

- K dispozici na Windows, Mac OS X a Linux. Při vydání verze 2.2 je JavaFX k dispozici na všech hlavních platformách stolních počítačů, a zajišťuje tak konzistentní zkušenost s běhovým prostředím pro vývojáře a koncové uživatele. Oracle zajišťuje synchronizované vydávání a aktualizace na všech třech platformách a nabízí rozsáhlý program na podporu firem se systémem kritických aplikací [10].

1.2.6 Výhody pro vývojáře a společnosti

JavaFX poskytuje následující výhody pro vývojáře, které jsou součástí Java [10].

- Díky tomu, že je platforma JavaFX napsána v programovacím jazyce Java, mohou vývojáři využívat jeho stávající dovednosti a nástroje pro rozvoj JavaFX aplikací.
- Používáním homogenního souboru Java technologií pro oba servery, dochází ke značné redukci rizikových investic a snižuje se složitost podnikových řešení.
- Programovací jazyk Java je používán po celém světě, proto je opravdu snadné najít zkušené vývojáře, kteří mohou rychle zvyšovat svou produktivitu ve vytváření JavaFX aplikací.
- Díky výše zmiňovaným výhodám jsou náklady na vývoj prudce snižovány, a společnost tak může ušetřené finance investovat efektivněji.
- Poskytuje vývojářům prostředí pro vytvoření podnikových a obchodních aplikací, které je možné spustit na více platformách.

1.2.7 Výhody oproti jiným platformám

JavaFX je plnohodnotně vyvinutá platforma pro bohaté internetové aplikace a má mnohem více výhod než jiné rovnocenné technologie na trhu. Je zde mnoho klíčových faktorů, které od nich JavaFX výrazně odlišuje [1].

- RIA pro všechny obrazovky: JavaFX poskytuje jednotný vývoj a nasazení modelu pro vytváření působivých bohatých internetových aplikací napříč plochou, prohlížečem, mobilním telefonem a televizí.
- Bohatá klientská platforma: JavaFX pracuje jednoduše a intuitivně při integrování grafiky, videa, zvuku, animace a bohatého textu.
- Snadno použitelná: JavaFX Skript je jednodušší na naučení a implementaci jazyka a nabízí deklarativní syntaxi, která dělá programování snazší z hlediska vizuálního kontextu.

- Rychlost náběhu programu: JavaFX využívá všudypřítomnost, sílu, výkon a zabezpečení, které mu poskytuje JRE (Java Runtime Environment) – ve volném překladu je to prostředí v Javě, které je určeno pro „běh“ programů.
- Čas uvedení na trh: JavaFX nabízí dramaticky zkrácené výrobní cykly pro návrháře a vývojáře skrz její návrhářsko-vývojářské workflow. Také umožňuje začlenění multimediálních aktiv od populárních výrobců designerských nástrojů, jako je např. Adobe Ilustrator nebo Photoshop pomocí Java ProductionSuite.
- Připravený masový trh: JavaFX umožní distribuovat bohaté internetové aplikace hromadně, rychleji a jednodušeji a to skrz biliony přístrojů využívajících Java.
- Ochrana investic: Zde je možné znovu použít existující Java knihovnu v JavaFX, tím dochází k zachování investic, které byly v Java již vykonány.
- Funkčnost napříč vyhledávači: JavaFX poskytuje jednotný uživatelský požitek napříč všemi prohlížeči na rozdílných platformách.
- Osvědčený model zabezpečení: Umožňuje širší přístup k systému s osvědčeným Java modelem zabezpečení.
- Podnikové sjednocení: Integruje bohaté uživatelské rozhraní s komplexním podnikovým back-endem.

  Jako back-end se označuje ta část webové aplikace, která slouží k administraci webu a ke zpracování dat. V případě redakčního systému umožňuje vkládání a úpravy článků, zakládání účtů dalších administrátorů, či třeba manipulaci se strukturou celého webu. U internetového obchodu bude back-end sloužit především ke vkládání nového zboží a k úpravám jeho vlastností, například cen [11].

1.3 Zásuvný modul

Zásuvný modul lze považovat za určitý návrhový vzor, jež ukazuje jednoduché vytvoření modulární aplikace na základě tohoto vzoru. Dá se tedy říci, že se nejedná o samostatný program, nýbrž o program, který potřebuje aplikaci jako podporu a zároveň ji rozšiřuje. Výhoda těchto modulů spočívá v možnosti volby uživatele, který si instaluje jen moduly, jež chce využívat a nestahuje navíc ty nadbytečnostmi, které by počítač zbytečně zatěžovaly. Pro lepší pochopení zásuvných modulů lze použít obr.1.4, kde jsou zobrazeny hostující aplikace a zásuvný modul [40].

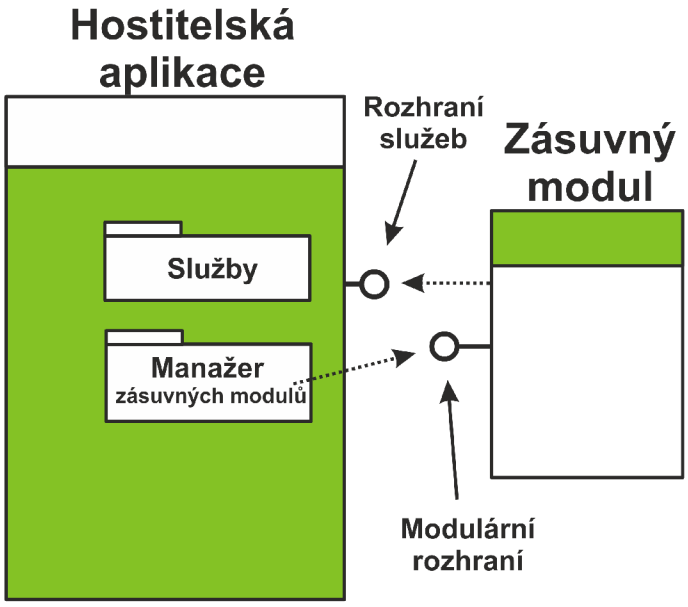
Zásuvný modul je sada softwarových komponent, které přidávají specifické vlastnosti větším softwarovým aplikacím. Aplikaci navrženou jako zásuvný modul lze rozšiřovat bez nutnosti změny či rekompilace. Vzhledem k tomu, že hostitelská aplikace je na zásuvných modulech nezávislá a nemůže tedy vědět, jak jsou implementovány,

definuje (předepisuje) rozhraní modulu, jež musí každý z těchto modulů respektovat. Nicméně vlastní implementace předepsaných metod je již ponechána na vývojáři konkrétního modulu. Plugin manager je používán pro načtení a spravování zásuvných modulů v hostitelské aplikaci. Všechny zásuvné moduly mohou být umístěny v jakémkoliv adresáři odkud plugin manager načítá všechny modulární rozhraní [40].

Plugin manager může načítat zásuvné moduly buď při startu hostitelské aplikace nebo dynamicky zatímco aplikace běží. Dokonce je možné zásuvný modul aktualizovat při jeho běhu a bez restartování základní aplikace [40].

V mnoha aplikacích zásuvné moduly potřebují některé funkce hostitelské aplikace, ta jim je poskytuje v rozhraní služeb nebo aplikačním rozhraní [40].

Další koncept, který je zvažován, mezitím co jsou modulární aplikace rozvíjeny, je aplikační doména. Ta poskytuje izolační hranice pro bezpečnost a spolehlivost a uvolňování sestav zásuvných modulů [40].



Obr. 1.4: Zásuvný modul, vlastní zpracování dle zdroje [22].

1.4 Grafické uživatelské prostředí

Grafické uživatelské rozhraní v JavaFX je vytvářeno pomocí SceneBuilder, který umožňuje rychle navrhnout uživatelské rozhraní. Umožňuje přetahovat různé komponenty do uživatelského rozhraní (Button, TextArea, Pane atd.), jednoduše měnit vlastnosti přidaných komponent a jejich styl. Výsledkem je pak FXML soubor, který je kombinován s kódem vývojáře.[12]

Obsah FXML kódu po rozvržení je automaticky generován v pozadí. FXML je založen na XML značkovacím jazyce pro definování uživatelského rozhraní v aplikaci JavaFX [12].

1.4.1 Značkovací jazyk definující uživatelské prostředí

FXML je jazyk založený na XML, který poskytuje strukturu pro budování uživatelského rozhraní a jazyk oddělený od aplikační logiky kódu. Toto oddělení prezentace a aplikace logiky je atraktivní pro webové vývojáře, protože mohou sestavit uživatelské rozhraní, které využívá Java komponenty bez osvojení kódu pro načtení a vyplnění údajů [19].

FXML nemá schéma, ale má základní předdefinovanou strukturu. Co lze vyjádřit v FXML, a jak se to uplatní při výstavbě scénového grafu, závisí na API objektů, které vytváří. Vzhledem k tomu, že FXML značkuje přímo do Javy, umožňuje použít dokumentaci API, aby bylo pochopitelné, jaké prvky a atributy jsou povoleny. Obecně platí, že většina JavaFX tříd mohou být použity jako prvky a většina Beanatributů mohou být použity jako vlastnosti [19].

Rozšiřitelný značkovací jazyk je z hlediska modelu „Zobrazit Controller (MVC)“ pohled, který obsahuje popis uživatelského rozhraní. Controller je třída, volitelně implementující Initializable třídu, která je deklarována jako regulátor pro FXML soubor. Model se skládá z domény objektů definovaných na straně Java, které se připojují k pohledu skrze regulátor [19].

K vytvoření libovolného uživatelského rozhraní je značkovací jazyk zvláště užitečný pro ta uživatelská rozhraní, která mají velké, složité scénové grafy, formuláře, zadávání dat nebo složitější animaci. Je také velmi vhodný pro definici statických rozložení, jako jsou formuláře, ovládací prvky a tabulky. Kromě toho se dá použít ke zkonstruování dynamického rozvržení pomocí přidělených skriptů [19].

2 PRAKTICKÁ ČÁST

Na základě nastudování teoretických poznatků byla vytvořena aplikace, která je v následujících kapitolách popsána. Aby bylo možné lépe vysvětlit princip fungování této aplikace, bude její popis proložen ukázkami zdrojových kódů, diagramů tříd a životních cyklů. Aplikace obsahuje celkem sedm balíčků, jež jsou vyjmenovány a popsány v kapitole Implementace.

Jako první bylo vytvořeno modulární jádro aplikace zpracovávající základní aritmetické operace, jež jsou schopny pracovat s hodnotami v množině reálných čísel.

V dalším kroku bylo navrženo grafické uživatelské prostředí obsahující horní menu, jež ukazuje možnosti kaskádových stylů, a dvě postranní menu. V levém bočním menu jsou ve stromové struktuře obsaženy zásuvné moduly v podobě jednoduchých aritmetických operací a vstupů pro tyto operace. Menu také obsahuje prvek, s jehož použitím lze ve stromové struktuře zásuvné moduly filtrovat. V pravém menu byla implementována třída, pracující s anotacemi a generující základní nápovědu pro lepší pochopení uživatelem. Horní menu je rozděleno do čtyř skupin, z nichž každá skupina obsahuje tlačítka pro ovládání aplikace.

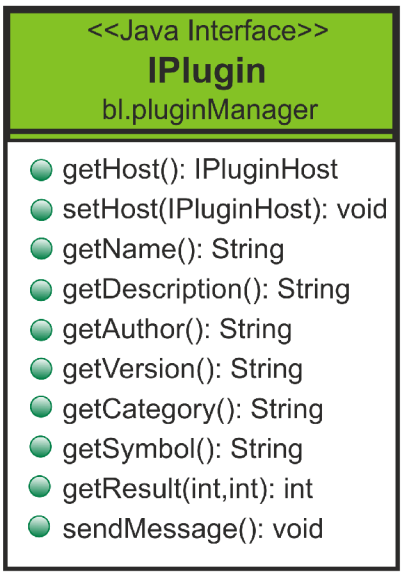
Následně byla na levé menu implementována funkce drag & drop, která umožňuje přesun objektu z jednoho místa na druhé. Funkce byla upravena, aby zprostředkovala přesun objektu z prvku typu `TreeView`, transformaci na vymezeném místě do objektu `shape` typu `Entity`, a vepsání vlastního popisu `Description` do tohoto objektu.

2.1 Modulární aplikace

Tato kapitola byla zpracována pomocí zdroje [40].

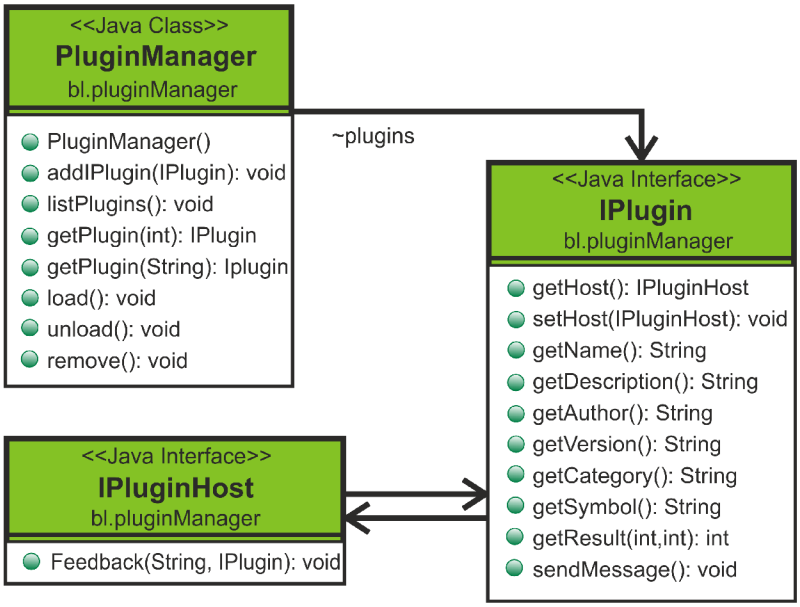
Jádrem modulární aplikace je třída `FormMain` obsažena v balíčku `ui.run`. Balíček `bl.pluginManager` obsahuje rozhraní `IPlugin`, na jehož základě budou vytvořeny třídy `Subtraction`, `Addition`, `Division`, `Multiplication`, `InputDouble` a `OutputDouble`, které budou přidány jako zásuvné moduly, tudíž nebude docházet ke změně hostitelské aplikace. Při spuštění aplikace se budou všechny operace načítat z implicitního adresáře `plugins`, který je umístěn v kořenové složce aplikace. Všechny vytvořené třídy obsahují aplikační rozhraní a modulární rozhraní, viz obr. 2.1, jež jsou implementovány v rozhraní `IPlugin`.

Důležitou součástí zásuvného modulu je zpětná vazba `Feedback`, přes kterou je vedená komunikace s hostitelskou aplikací. Na základě této komunikace hostitelská aplikace kontroluje zda uvnitř zásuvného modulu nenastala chyba.



Obr. 2.1: Definice modulárního rozhraní `IPlugin`, vlastní zpracování.

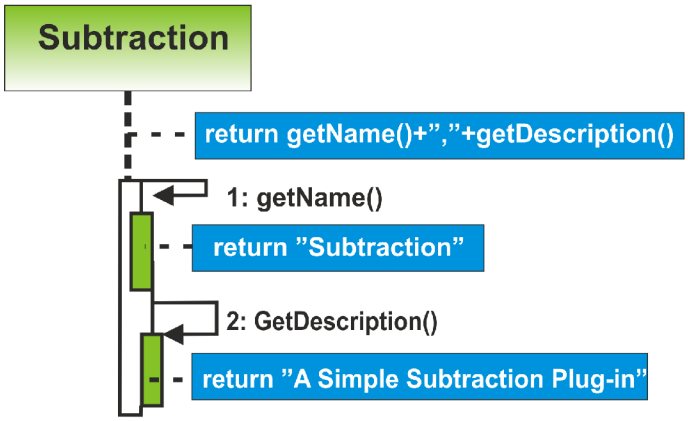
Hlavní třída modulární aplikace nebo – li `PluginManager` je použita k aktivaci a správě zásuvných modulů, a poskytuje k těmto modulům přístup. V ukázce obr. 2.2 je zobrazen vztah mezi třídou `PluginManager` a rozhraním `IPlugin`, a vzájemná komunikace mezi rozhraními `IPlugin` a `IPluginHost` v tomto programu.



Obr. 2.2: Diagram tříd modulárního rozhraní, vlastní zpracování.

V balíčku `bl.plugins` jsou vytvořeny zásuvné moduly, jež mají atributy `IPlugin` ve svém konstruktoru. Protože třída `IPlugin` poskytuje základní funkcionalitu pro zásuvné moduly, jsou implementovány jako obecný parametr. Všechny tyto zásuvné

moduly (`Adition`, `Division`, ...) tedy obsahují konstruktory `getName`, `getDescription`, `getAuthor`, `getVersion`, `getCategory`, `getSymbol`, apod., a jsou exportovány do JAR souborů. Veškeré načtené soubory (zásuvné moduly) s koncovkou `.jar` jsou naplňovány do prvku typu `TreeItem`, kterým je vyplněn prvek `TreeView`. Na obr. 2.3 lze vidět názornou ukázku životního cyklu třídy `Subtraction`.



Obr. 2.3: Životní cyklus třídy `Subtraction`, vlastní zpracování.

2.2 Návrh uživatelského rozhraní

Celá aplikace je rozvržena v komponentě `BorderPane` a skládá se z vrchního menu `ribbonControl`, dvou bočních menu `leftPane`, `rightPane` a pracovní plochy nebo-li `centerPane`.

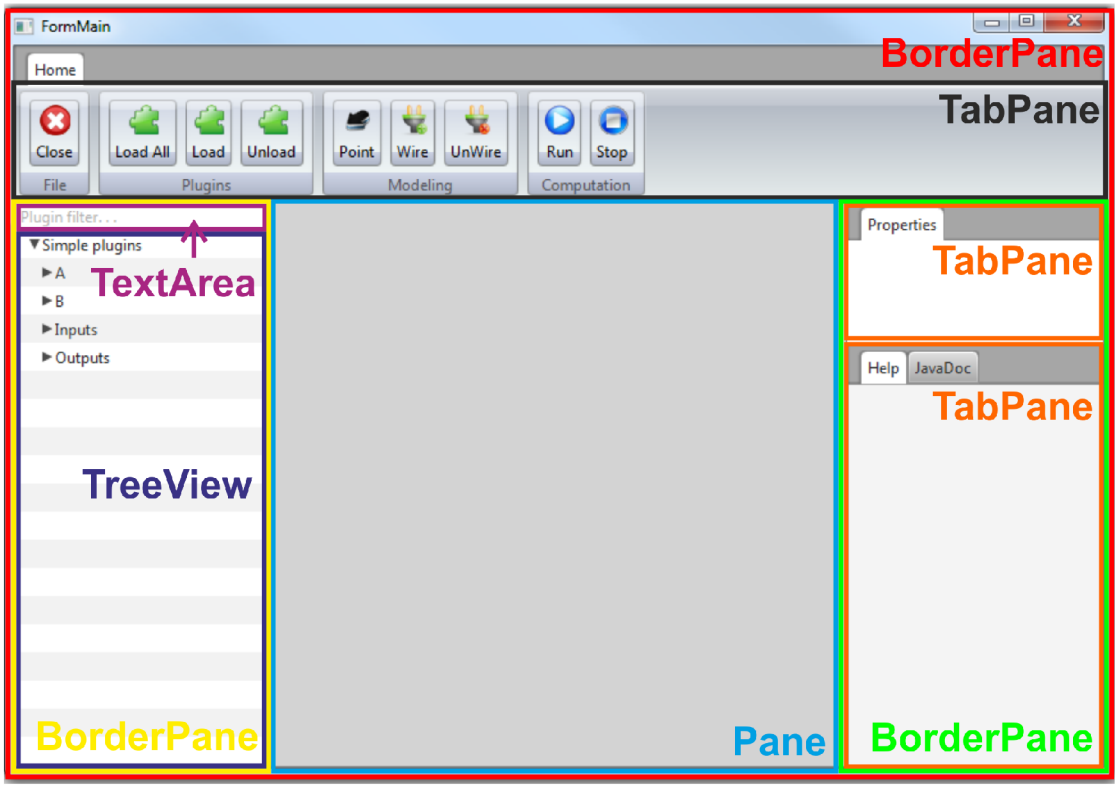
Ve vrchním menu se nachází jednoduchá tlačítka pro ukončení aplikace, načítání zásuvných modulů, mazání zásuvných modulů, propojování prvků, apod. Je zde ukázána bohatost grafických možností platformy JavaFX vytvořených pomocí kaskádových stylů. Jednotlivé funkce tlačítek budou rozepsány v kapitole Rozbor uživatelského rozhraní. Na obr. 2.4 je vrchní menu zvýrazněno černou barvou.

V levém bočním menu se nachází zásuvné moduly, které jsou rozvrženy v komponentě typu `TreeView`, jež má stromovou strukturu. Toto menu je rozděleno do kategorií `A`, `B`, `Inputs` a `Outputs`, jež po rozvinutí obsahují zásuvné moduly aritmetických operací, vstupy a výstupy, které budou využívány aritmetickými operacemi. Použitím funkce drag & drop je možné moduly přesouvat na pracovní plochu. Prvek z kategorie `A` nebo `B` přetažený na pracovní plochu aktivuje pravé dolní menu s nápovědou. Pokud se však bude jednat o vstupy (`Inputs`), aktivuje se nejen nápověda, ale i pravé horní menu `Properties` obsahující prvek `TextField`, který slouží pro zadávání numerických hodnot pro výpočet. Nad stromovou strukturou se nachází

prvek typu `TextArea`, který umožňuje filtrování načtených zásuvných modulů. Levé boční menu je na obr. 2.4 ohraničeno žlutou konturou.

Ve spodní části pravého bočního menu se nachází jednoduchá nápověda v prvku typu `TabPane`, jež obsahuje dvě komponenty `Tab`. První z nich `Help` by měl uživateli poskytnout základní popis zvoleného modulu, druhý `JavaDoc` zobrazuje například základní funkce, parametry či návratové hodnoty zásuvného modulu. Pravé boční menu je na obr. 2.4 ohraničeno zelenou barvou.

Pracovní plocha rozvržena v komponentě `Pane`, je využívána k přesunu položek z prvku typu `TreeView` pomocí funkce drag & drop. Co se týče zásuvných modulů umístěných na pracovní ploše, fungují na tomtéž principu a mohou být tedy přesouvány v rámci pracovní plochy pomocí funkce drag & drop. Pracovní plocha je na obr. 2.4 zvýrazněna světle modrou barvou.



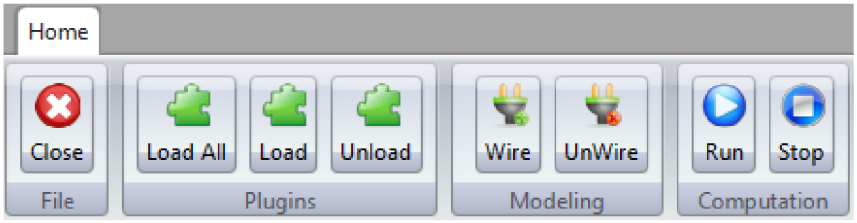
Obr. 2.4: Rozvržení uživatelské rozhraní, vlastní zpracování.

2.3 Rozbor uživatelského rozhraní

Tato kapitola se zabývá podrobným popisem vzhledu a funkčnosti jednotlivých menu. Bude doplněna o ukázky vzhledu aplikace, zdrojových kódů a vývojových diagramů.

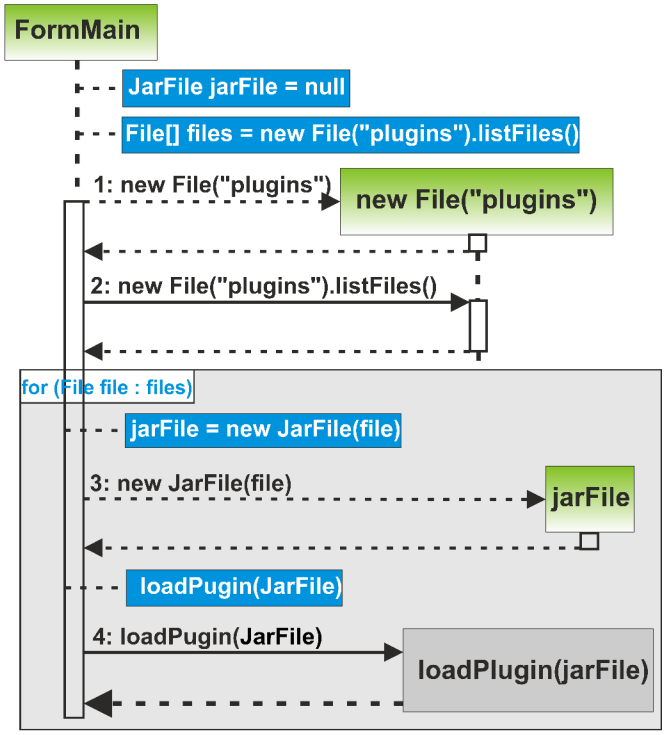
2.3.1 Vrchní menu

Vrchní menu na obr. 2.5 se skládá ze čtyř částí. První skupina `File` obsahuje pouze jedno tlačítko `Close`, které umožňuje zavřít celou aplikaci.



Obr. 2.5: Ukázka vrchního menu, vlastní zpracování.

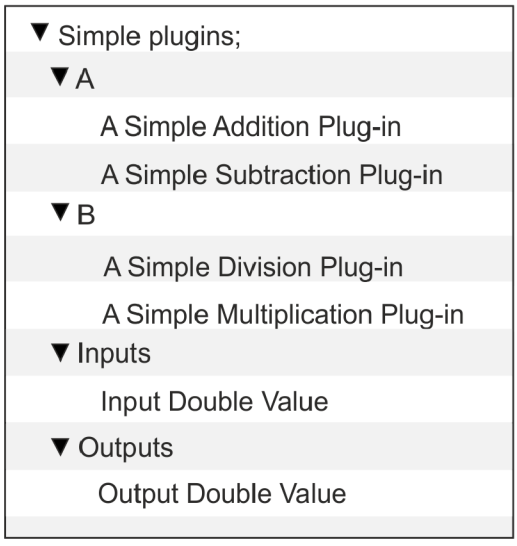
Druhá skupina `Plugins` slouží pro správu zásuvných modulů. Prvním tlačítkem `Load All` (načíst všechny) budou načteny všechny soubory obsahující koncovku `.jar`. Cesta k základním modulům jednoduchých aritmetických operací je nadefinována implicitně do složky `plugins`. Na obr. 2.6 je znázorněn životní cyklus metody `loadAllPlugins()`, ve které je provedena implementace pole s názvem `files`, do něhož se zapíší všechny zásuvné moduly obsažené v adresáři `plugins`. V následujícím kroku cyklus `for` provede výčet všech zásuvných modulů obsažených v poli `files`, a postupně přiřazuje cesty s názvy zásuvných modulu do `JarFile`.



Obr. 2.6: Načtení zásuvných modulů ze složky `plugins`, vlastní zpracování.

Druhé tlačítko `Load` umožňuje načítat jednotlivé zásuvné moduly. Po stisknutí tohoto tlačítka se v implementované funkci vytvoří nový `FileChooser`, následně je mu nadefinován filtr pro koncovku `.jar` a implicitní cesta do adresáře `plugins`. Poté je zobrazen otevírací dialog v adresáři `plugins`. Po vybrání zásuvného modulu bude cesta vybraného modulu zaznamenána do řetězce a bude vytvořen nový `file` s touto cestou.

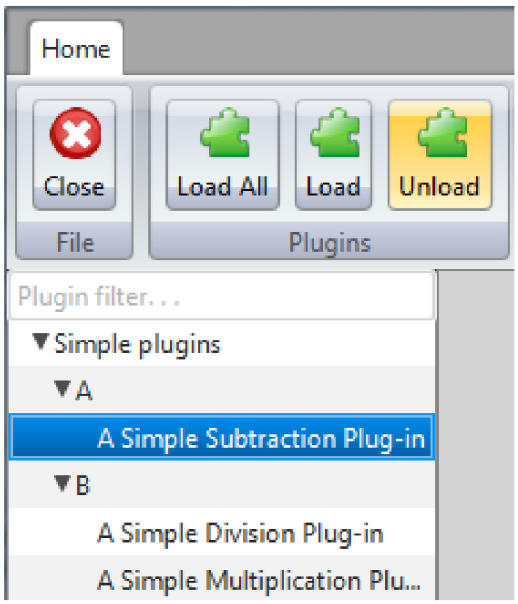
U obou předchozích tlačítek `Load All` a `Load` bylo zapotřebí implementovat metodu `loadPlugin()`. Tato metoda zapíše popis zásuvného modulu do komponenty `originalData` typu `ObservableList`, a do komponenty typu `TreeView` se načte prvek `TreeItem <String>` s popisem zásuvného modulu. V dalším kroku bude pomocí cyklu `for` rozhodnuto, do jaké ze čtyř nabízených kategorií bude tento zásuvný modul zařazen. Zásuvné moduly jsou rozděleny do kategorií jednoduchých aritmetických operací, které jsou dále rozděleny do dvou skupin: A (`Addition, Subtraction`), B (`Division, Multiplication`), a kategorií `Outputs` (výstupy) a `Inputs` (vstupy), kde je možné definovat hodnoty, se kterými bude počítáno v aritmetických operacích viz obr. 2.7.



Obr. 2.7: Rozdělení zásuvných modulů do kategorii, vlastní zpracování.

Posledním tlačítkem ve skupině `Plugins` je `Unload`, kterým je možné odebírat jednotlivé zásuvné moduly viz obr. 2.8. Po kliknutí na toto tlačítko se provede událost `setOnMouseClicked()`, kde `SelectedNode` typu `TreeItem<String>` je proměnná, pomocí které se v komponentě `TreeView` zvolí položka pro odebrání z tohoto prvku. Pomoci proměnné `index` typu `integer` bude zjištěno jakou hodnotu má index námi vybrané položky. V první podmínce se odstraní tento prvek z komponenty

`originalData`, aby v levém menu, popsaném v další kapitole správně fungovalo filtrování. Následně zjistíme pomocí proměné `ParentNode` typu `TreeItem<String>`, ve které kategorii se zvolený prvek pro odstranění nachází, a v druhé podmínce jej odstraníme.



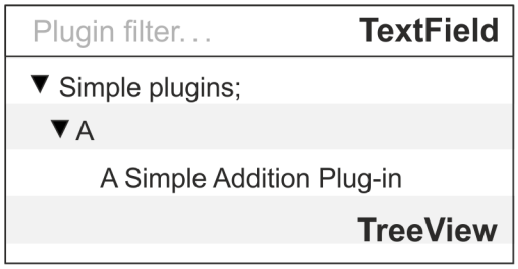
Obr. 2.8: Odebrání zásuvného modulu, vlastní zpracování.

Třetí skupina `Modeling` je určena pro propojování vstupních prvků s aritmetickou operací. Skupina se skládá ze dvou tlačítek, z nichž aktivní je pouze tlačítko Wire. UnWire tlačítko je zde připraveno pro možnost rozpojení prvků. Po stisknutí tlačítka Wire se do konzole vypíše, zda je propojení prvků aktivní nebo neaktivní. Za předpokladu, že je tedy aktivní, se do proměnné `connector` nastaví hodnota `FirstElement`. Aktivované tlačítko Wire čeká na označení první položky umístěné na pracovní ploše. Jakmile je toto označení provedeno, v proměnné `connector` se nastaví hodnota `SecondElement`, jež znovu čeká na označení druhého objektu. Po výběru druhého objektu, se hodnota `connectoru` nastaví na `NoElement`. Poté dojde k propojení obou položek linií. Při označení první položky se do pomocné proměnné `op1` zaznamená, že jde o operátor. V druhé vybrané položce se proměnná `op1` nastaví jako vstupní hodnota pro aritmetickou operaci, první objekt je tedy vždy vstupem pro aritmetickou operaci.

Po propojení dvou vstupů s aritmetickou operací linií je možné přejít k využití tlačítka Run, které se nachází v poslední skupině vrchního menu s názvem `Computation`. Toto tlačítko slouží ke spuštění matematických výpočtů a vypsání výsledku do konzole.

2.3.2 Levé boční menu

Levé boční menu na obr. 2.9 se skládá ze dvou komponent `TextField` a `TreeView`.



Obr. 2.9: Ukázka levého bočního menu, vlastní zpracování.

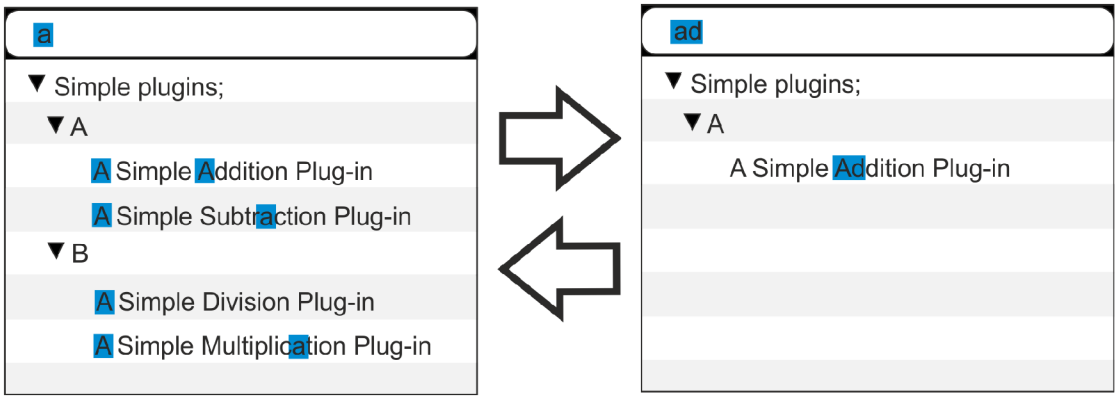
První zmíněnou komponentou se provádí hledání uvnitř druhé komponenty. Pro lepší orientaci je na obr. 2.10 celá tato operace názorně ukázána. V počátku vývoje aplikace se místo komponenty `TreeView` používala komponenta `ListView`. Ačkoliv implementace filtrovaní byla v tomto prvku méně složitá, musel být tento prvek nahrazen, protože neodpovídal zvoleným požadavkům (nepodporoval stromovou strukturu). Filtrace je prováděna zadáváním hodnot do prvku `treeFilter` na základě zdrojového kódu 2.1, v němž je na komponentu `TextField (treeFilter)` přidán `ChangeListener`. Ten naslouchá změnám provedeným ve filtrovacím prvku `treeFilter`. Použití filtrace ve stromové struktuře nebylo dosud v žádné odborné literatuře publikováno, proto bylo nutné vytvořit vlastní filtraci. Popis funkčnosti filtrace následuje za ukázkou zdrojového kódu 2.1.

Kód 2.1: Naslouchání na změny ve filtrovacím prvku.

```
/* FormMain */
...
treeFilter.textProperty().addListener(new ChangeListener<String>() {
    @Override
    public void changed(ObservableValue<? extends String>
    observable, String oldValue, String newValue) {
      updateTreeViewData();
    }
});
...
```

Metoda `updateTreeViewData()`, viz kód 2.2, provádí výčet všech filtrovaných dat obsažených v prvku `originalData` typu `ObservableList`. Tato metoda obsahuje dílčí metodu `matchesFilter()`, viz kód 2.3, která získá text z prvku

`treeFilter` a porovnává jej s daty v prvku `originalData`. Následně se do prvku `filteredData` typu `ObservableList` naklonují pouze ty prvky, které odpovídají uživatelem hledanému `substringu`. Poté dojde k vymazání všech prvků `TreeItem` z levého menu, je proveden výčet prvků z `filteredData` a zobrazení hledaných prvků v komponentě `TreeView`. Výsledek filtrování je zobrazen na obr. 2.10.



Obr. 2.10: Ukázka filtrování dat, vlastní zpracování.

Kód 2.2: Ukázka zdrojového kódu pro filtraci dat.

```
/* FormMain */
...
 filteredData.clear();
        for (String p : originalData) {
            if (matchesFilter(p)) {
                 filteredData.add(p);
                 System.out.println(p);
            }
         }

        treeItems.getChildren().clear();
        boolean categoryFound = false;
        for (String str : filteredData) {
            IPlugin plugin = pm.getByDescription(str);
            for (TreeItem<String> node : treeItems.getChildren()) {
                if (node.getValue().contentEquals(
                plugin.getCategory())){
                    node.getChildren().add(new TreeItem(str));
                    categoryFound = true;
                }
            }
```

```
            if (!categoryFound) {
                TreeItem node = new TreeItem(plugin.getCategory());
                treeItems.getChildren().add(node);
                node.getChildren().add(new TreeItem(str));
            }
        }
 }
...
```

Kód 2.3: Zdrojový kód metody `matchesFilter()`

```
/* FormMain */
...
  String filterString = treeFilter.getText();
  ...
     if (p.toString().toLowerCase().indexOf(
     lowerCaseFilterString) != -1) {
        return true;
  }
...
```

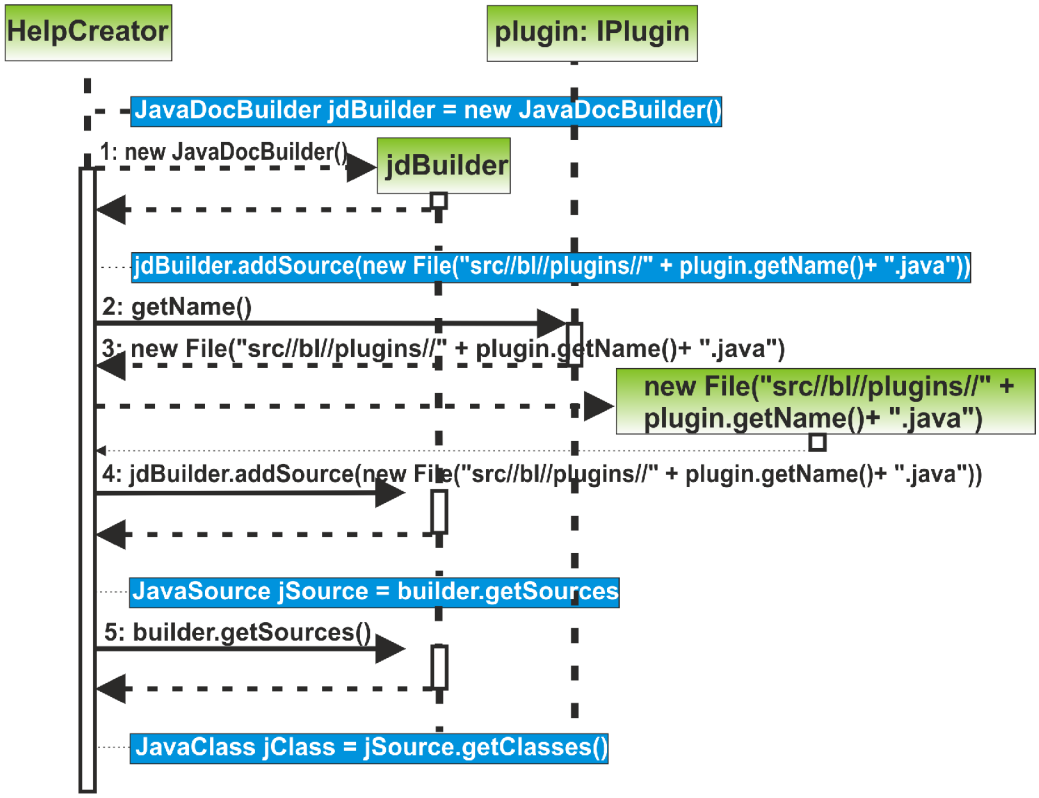
2.3.3 Pravé boční menu

Pravé boční menu je rozděleno na dvě části - vrchní část nebo-li `Properties` a spodní část obsahující nápovědu. Po přetažení operátoru `Input Double Value` pomocí funkce drag & drop z kategorie `Inputs` na pracovní plochu se pravé horní menu `Properties` aktivuje. V tomto menu je prvek `TextField` označený jako `txfParam`, jež nese numerickou hodnotu jednoho vstupu pro aritmetickou operaci. Každý tento vstup má předdefinovanou hodnotu 10. Na základě parametrů `NEGATIVE_INFINITY`, `POSITIVE_INFINITY` je uživatel oprávněn tuto hodnotu měnit na jakékoliv kladné či záporné číslo. Přetahování prvků `Input Double Value` a aritmetických operací na pracovní plochu je omezeno kapacitou při přidělování `Id`, jež nabývá hodnoty do 1000. Každá aritmetická operace dokáže však pracovat pouze se dvěma vstupy.

V pravém bočním menu se také nachází nápověda sloužící pro lepší pochopení aplikace uživatelem. Implementace nápovědy je provedena v balíčku `ui.run`, který obsahuje třídu `HelpCreator`, jež je volána z hlavní třídy `FormMain`. Třída `HelpCreator` získává nápovědu ze zásuvných modulů na základě anotací.

Pro načtení třídy aritmetické operace a vygenerování nápovědy je nutné nejprve zajistit přístup k třídě, kterou uživatel zvolil. Pomocí životního cyklu na obr. 2.11 je ukázán přístup k třídě a načtení obsahu zvolené třídy. Jako první je nutné vytvořit instanci `jdBuilder` typu `JavaDocBuilder`, do této proměnné je přiřazen zdroj

s implicitní cestou do adresáře `plugins`. Název třídy, pro kterou bude vygenerována nápověda, se odvíjí od příkazu `plugin.getName()`, na jehož základě bude zjištěno, který zásuvný modul je označen. V této chvíli je již znám název zásuvného modulu, a je potřeba mu přiřadit koncovku `.java`, čímž dostaneme přesnou cestu a název třídy. Například pro třídu `Subtraction` je cesta `bl/plugins/Subtraction.java`. Jakmile je známa trasa a název požadované třídy, je možné načíst obsah celé třídy do pomocné proměnné `jSource`. Další pomocná proměnná s názvem `jClass` kontroluje, zda třída neobsahuje další podtřídy. Co se týče této aplikace, žádný ze zásuvných modulů neobsahuje podtřídu.



Obr. 2.11: Životní cyklus přístupu k třídě a načtení jejího obsahu, vlastní zpracování.

Nápověda obsahuje dva prvky typu `Tab`, a to `Help` a `JavaDoc`. První z nich `Help` slouží uživateli jako vodítko k použití aplikace. Z vybrané třídy jsou do řetězce `comment` přiřazeny informace o aritmetické operaci, které byly nejprve zaznamenávány do prvku `TextArea`, jemuž byla zakázána editace. Při pozdějších úpravách se ukázalo, že použití tohoto prvku není vhodné, protože umožňuje vkládat pouze řetězce, jež není možné dále stylovat. Proto byla zvolena komponenta `VBox`, do kterého lze vkládat prvky typu `Text` a lze jim pomocí příkazu `setFont` nadefinovat různé styly jako například `BOLD`, `ITALIC`, změna velikosti atd. Generovaná nápověda se tedy zaznamenává do prvků typu `Text` např. `textComment`. Do prvku `boxHelp`

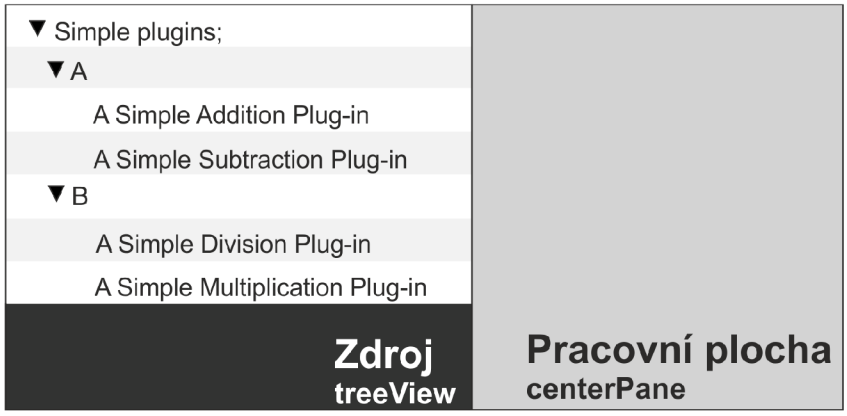
typu `VBox` jsou zděděny prvky typu `Text`, ve kterých jsou zaznamenány informace k použití aplikace.

Do druhého prvku `JavaDoc` jsou generovány konstruktory s jejich parametry a návratovými hodnotami. Tato nápověda je opět generována do prvků typu `Text`.

2.4 Drag & drop

Tato kapitola byla zpracována pomocí zdroje [41].

V praxi je funkce drag & drop poměrně dost využívaná, umožňuje přenos objektů z jednoho místa na druhé. Funguje na principu uchopení objektu ze zdroje a následným přetažením do cíle. Při přetahování vzniká zpětná vazba, která označuje, zda je objekt přijímán či nikoliv. Zdrojem, viz obr 2.12, byla určena komponenta `TreeView`, deklarována jako `treeView`. Cílem byl zvolen prvek `Pane`, jehož deklarace je `centerPane`.



Obr. 2.12: Ukázka zdroje a cíle funkce drag & drop, vlastní zpracování.

Základní třída, která byla použita k implementaci drag & drop, je `javafx.scene.input.DragEvent`. Drag & drop funkce bývá volána přes událost `startDragAndDrop` uvnitř obslužné funkce `treeCell.setOnDragDetected`. Metoda `startDragAndDrop` obsahuje řadu možností pro přesun objektu, jakými jsou `TransferMode.MOVE`, `TransferMode.COPY` atd. Metoda `MOVE` se v tomto případě jevila jako nevhodná, protože místo toho, aby se přemisťovaný objekt nakopíroval a přenášela se pouze jeho kopie, se při přesouvání na pracovní plochu prvek z menu vyjmul. Jako vhodnější funkce se jevila metoda `COPY`, která kopírování objektu při přesouvání umožňuje viz kód 2.4

Kód 2.4: Zdroj funkce drag & drop.

```
/* FormMain */
...
treeCell.setOnDragDetected(new EventHandler<MouseEvent>() {
  @Override
  public void handle(MouseEvent event) {
    Dragboard db = treeView.startDragAndDrop(TransferMode.COPY);
        ...
        event.consume();
     }
});
...
```

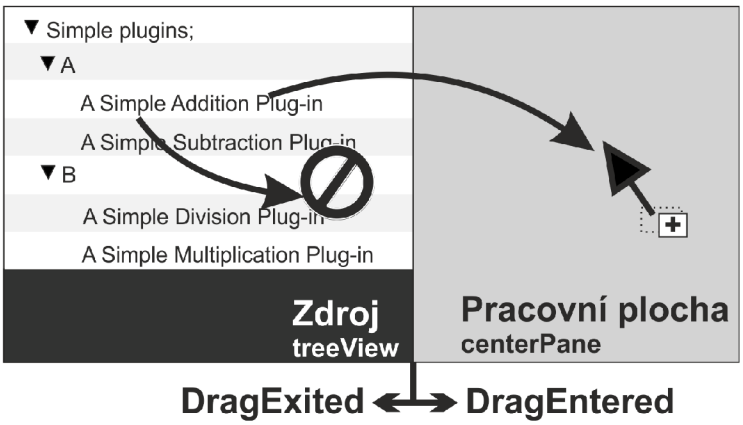
Manipulace s objektem pomocí události `DragOver` (kód 2.5) je uskutečněna, jakmile je objekt přesunut přes okraj pracovní plochy (je provedena událost `DragOver`). Důležitou událostí je `EventHandler`, uvnitř které je metoda `event.acceptTransferModes`, jež ohlašuje, zda je přijímán mód kopírování `TransferMode.COPY` či nikoliv. Pokud se tedy kurzor nachází nad pracovní plochou a dojde k upuštění objektu, objekt se zde nakopíruje.

Kód 2.5: Ukázka zdrojového kódu `DragOver`.

```
/* FormMain */
...
 public void setupGestureTarget(final Pane target) {
  target.setOnDragOver(newEventHandler<DragEvent>() {
    @Override
    public void handle(DragEventevent) {
     if (event.getGestureSource() != target&&
     event.getDragboard().hasString()) {
     event.acceptTransferModes(TransferMode.COPY);
     }
     event.consume();
    }
  });
...
```

V momentě, kdy je přenášeným objektem pohybováno nad pracovní plochou (nad cílem), je vzhled kurzoru odlišný, než když se vyskytuje mimo pracovní plochu. Pokud se kurzor nachází nad cílem a přenášený objekt je upuštěn, je provedena událost `DragEntered`, viz obr. 2.13, a pracovní plocha přijme upuštěný objekt.

Jestliže je objekt upuštěn mimo pracovní plochu, provede se událost `DragExited`, což znázorňuje obr. 2.13.



Obr. 2.13: Ukázka událostí `DragExited` a `DragEntered`, vlastní zpracování.

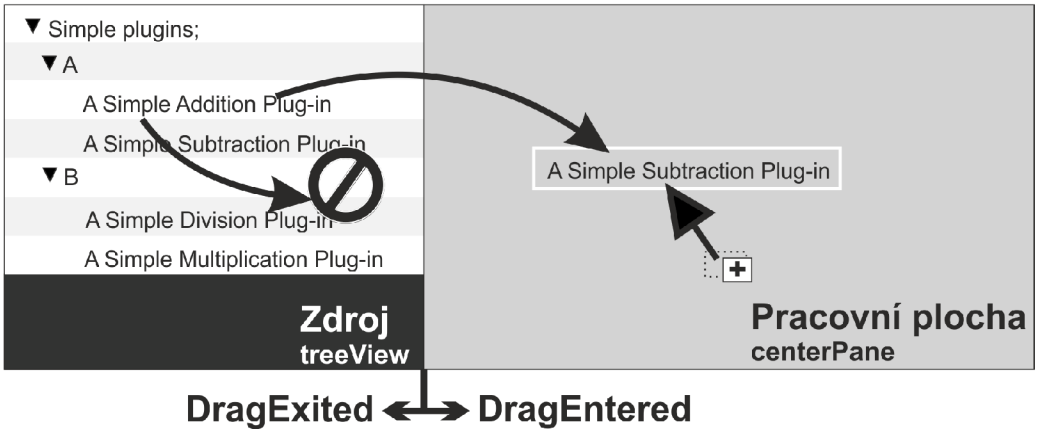
Událost na cíl nebo - li `DragDropped`, viz ukázka zdrojového kódu 2.6, je stav, kdy je objekt přenesen do cíle (na pracovní plochu) a upuštěn. Zde je nutné ve funkci `DragEvent` volat metodu `setDropCompleted(success)`. Tímto příkazem je zdroji oznámeno, zda byl řetězec zdárně přenesen a použit.

V události `DragDropped` je vytvořen operátor vybraného zásuvného modulu, kterému se přiřadí `Id` vygenerované ze třídy `IdGenerator`, poté je operátor spolu s `Id` zapsán do listu operátorů. Následně se vytvoří objekt `shape` typu `Entity`, kterému je přiřazeno vygenerované `Id` operátoru. Na tento objekt je implementována událost `setOnMouseClicked`, na jejímž základě je vytvořena nápověda. Tato událost dále obsahuje podmínku, která naslouchá aktivitě tlačítka Wire, pokud je tedy tlačítko aktivní, je možné provést propojení prvků. Přepínač při propojování mění hodnoty proměnné `connector` na `NoElement`, `FirstElement` nebo `SecondElement`. Podrobný popis je uveden v kapitole Vrchní menu.

Kód 2.6: Definice události na cíl.

```
/* FormMain */
...
target.setOnDragDropped(new EventHandler <DragEvent>() {
  @Override
  public void handle(DragEvent event) {
    Dragboard db = event.getDragboard();
    boolean success = false;
    if (db.hasString()) {
      ...
    }
    event.setDropCompleted(success);
    event.consume();
  }
...
```

Na obr. 2.14 je ukázka funkce drag & drop. Je zde zobrazeno boční menu obsahující zásuvné moduly ve stromové struktuře a pracovní plocha, na kterou se pomocí funkce drag & drop přenesou prvky aritmetických funkcí v podobě ohraničeného `shape` typu `Entity`, do kterého se vypíše popis pro daný modul. Kurzor myši při přetahování mění svůj vzhled a tím uživateli napovídá, zda se nachází nad cílem přenášeného objektu či nikoliv.



Obr. 2.14: Drag& drop objektu, vlastní zpracování.

Na ohraničený prvek `shape` je dále implementována událost drag & drop, která je vytvořena podobně jako všechny předchozí ukázky kapitoly Drag & drop, proto zde nebude uveden příklad. Jediná odlišnost spočívá ve změně zdroje a cíle funkce, jež nám umožňuje přemisťovat přesouvaný prvek pouze po pracovní ploše `centerPane`.

2.5 Implementace

Tato aplikace se skládá ze sedmi balíčků, které budou rozepsány v tab. 2.1. Každý z těchto balíčků obsahuje několik tříd. Jejich název a stručný popis bude uveden v tab. 2.2.

Tab. 2.1: Přehled balíčků aplikace, vlastní zpracování.

NÁZEV BALÍČKU	POPIS
`bl`	Obsahuje třídy rozšiřující zásuvné moduly, na základě kterých bude prováděn výpočet aritmetických operací.

Pokračování na další stránce

NÁZEV BALÍČKU	POPIS
`bl.plugins`	Obsahuje třídy, které provádí jednoduché matematické operace jako je sčítání, odčítání, násobení a dělení.
`bl.pluginManager`	V tomto balíčku je definována modulární aplikace.
`ui.base`	V tomto balíčku je obsažena třída a `CSS` soubor pro předpis základního vzhledu formuláře.
`ui.conf`	Balíček obsahuje třídu pro konfiguraci grafického okna.
`ui.run`	Tento balíček obsahuje třídu `FormMain`, která tvoří základ aplikace, a další třídy pro vytváření nápovědy nebo propojování prvků.

Tab. 2.2: Přehled jednotlivých tříd, vlastní zpracování.

TŘÍDA	POPIS
`Operator`	Tato abstraktní třída rozšiřuje zásuvné moduly o parametry např. (`getId`, `setId`, ...), které jsou potřebné k výpočtu aritmetických operací.
`ParamaterType`	Abstraktní třída obsahující konstruktor pro získání nebo vložení matematických hodnot ze vstupních prvků.
`ParamaterTypeDouble`	Třída rozšiřující třídu `ParamaterType`, ve které jsou definovány maximální a minimální hodnoty, se kterými je možné počítat výsledek.
`IPlugIn`	Tato třída je atributem pro zásuvné moduly a poskytuje jim základní funkcionalitu.
`IPlugin.Host`	Implementuje metodu zpětné vazby.
`PluginManager`	Tato třída slouží pro správu zásuvných modulů.

`Adition`
`Division`
`Multiplication`
`Subtraction`
`InputDouble`
`OutputDouble`
	Všechny tyto třídy jsou zásuvné moduly a obsahují jednoduché konstruktory `getName`, `getSymbol`, atd.

`FormBase`	Třída pro předpis základního vzhledu formuláře.

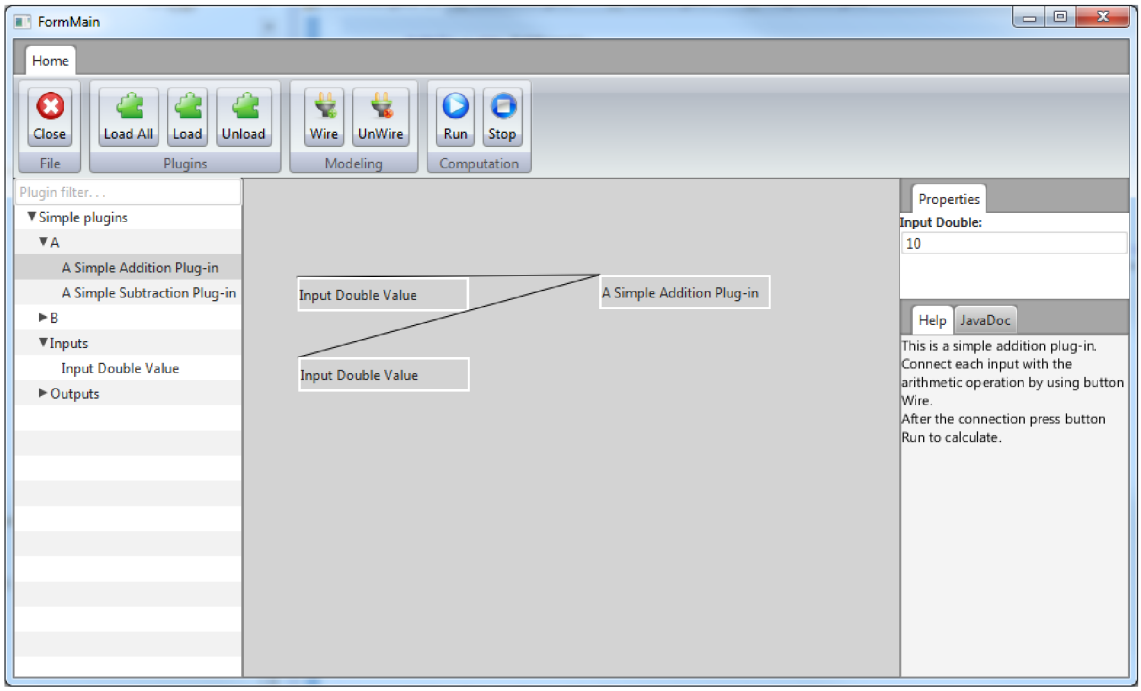
Pokračování na další stránce

TŘÍDA	POPIS
`ApplicationSetting`	Třída předepisující velikost aplikačního okna.
`ConnectionEnum`	Třída `enum`, ve které jsou definovány instance na propojení prvků.
`Entity`	V teto třídě se prvku `shape` typu `Entity` přiřadí identifikátor, základní vzhled a funkce drag & drop.
`FormMain`	Hlavní třída, jež řídí celý projekt.
`HelpCreator`	Třída pro generování nápovědy.
`IdGenerator`	Zde se generují `Id` každému objektu umístěnému na pracovní ploše.

2.6 Výsledná aplikace

Na obr. 2.15 je zobrazeno fungování celkové aplikace. Po spuštění aplikace je v pravém dolním rohu k dispozici nápověda, v níž je uvedeno, jak při výpočtu postupovat. Nejprve je tedy nutné pomocí funkce drag & drop přesunout dva vstupy nebo-li `Inputs Double Value` z levého menu na pracovní plochu, až poté se na pracovní plochu přesouvá vybraná aritmetická operace. Aritmetické operace i vstupy jsou ukryty ve stromové struktuře v kategoriích `A`, `B` a `Inputs`, je tedy nutné jednotlivé větve nejprve rozvinout. Nápovědu obsahují i kolonky `Inputs Double Value` a aritmetické operace vyskytující se na pracovní ploše, pro aktivaci je stačí stisknout. Po stisknutí kolonky `Inputs Double Value`, je zobrazena nápověda k zadávání číselných hodnot pro výpočet. Uživatel je naveden do menu `Properties`, kde se pole pro zadání hodnot nachází. Hodnota pole je přednastavena na číslo 10.

Na pracovní ploše se tedy nachází dvě buňky `Inputs Double Value` se zadanými numerickými hodnotami a buňka s aritmetickou operací. Dalším krokem je použití tlačítka Wire, jímž uživatel musí každý ze vstupů propojit s aritmetickou operací. Úspěšné propojení signalizuje linka spojující vstupy s aritmetickou operací. Nyní může uživatel přistoupit k použití tlačítka Run, po jeho stisknutí je proveden výpočet.



Obr. 2.15: Ukázka funkčnosti aplikace, vlastní zpracování.

3 ZÁVĚR

Cílem předložené bakalářské práce bylo na základě získaných teoretických poznatků vytvořit aplikaci, která slouží jako jednoduchý výpočetní systém, jež po zadání numerických hodnot a vybrání aritmetické operace spočítá výsledek.

Aby byl splněn cíl práce, bylo nejprve nutné navrhnout modulární jádro aplikace, jehož funkčnost je možné libovolně rozšiřovat použitím zásuvných modulů. Modulární jádro bylo implementováno v programovacím jazyce Java. Dále bylo navrženo a implementováno grafické uživatelské prostředí, jež se skládá z několika menu, tak aby byla aplikace přehledná, a snadno a intuitivně ovladatelná. Jednotlivá menu, jejich části, funkce a možnosti jsou v praktické části práce podrobně popsány. Grafické uživatelské prostředí bylo implementováno v programovacím jazyce JavaFX. Neméně důležité bylo vytvoření funkce drag & drop, sloužící k přesouvání komponent z bočního menu na pracovní plochu. Pro výpočet je nejprve nutné jednotlivé vstupy propojit s aritmetickou operací, proto bylo navrženo tlačítko Wire, jež toto propojení umožňuje. Následuje využití tlačítka Run, které na základě propojení objektů navrátí vypočítaný výsledek.

LITERATURA

[1] LAWRENCE Premkumar, PRAVEEN Mohan. *Beginning JavaFX* . New York: Apress, 2010. ISBN 978-143-0271-994.

[2] Ajax. In: *Vše okolo jQuery* [online]. 2010. [cit. 2012-10-30]. Dostupné z URL: <http://jquery-navod.cz/kategorie-ajax/9-ajax>.

[3] OKTÁBEC, Michal. AJAX a jQuery. In: *Jak na jQuery* [online]. 2012, 28.01 [cit. 2012-10-30]. Dostupné z URL: <http://jaknajquery.cz/ajax-a-jquery/>.

[4] LACKO, Ľuboslav. Výhody a nevýhody technológie AJAX. In: *Infoware* [online]. 2009, 07.12. [cit. 2012-10-30]. Dostupné z URL: <http://www.itnews.sk/tituly/infoware/2009-12-07/c130666–vyhody-a-nevyhody-technologie-ajax>.

[5] TUCKER, David. Enterprise RIA Development: Comparing Flex/-Flash, Silverlight, HTML5/CSS3, jQuery/JavaScript. In: *Code project* [online]. 2011, 19.05 [cit. 2012-10-30]. Dostupné z URL: <http://www.codeproject.com/Articles/199022/Enterprise-RIA-Development-Comparing-Flex-Flash-Si>.

[6] Microsoft Silverlight. In: *Microsoft/web* [online]. 2012 [cit. 2012-10-31]. Dostupné z URL: <http://www.microsoft.com/cze/web/silverlight/>.

[7] MACDONALD, Matthew a Mario SZPUSZTA. ASP.NET 3.5 a C# 2008: *tvorba dynamických stránek profesionálně*. Brno: Zoner Press, 2008. 1584 s. ISBN 978-80-7413-008-3.

[8] MANDU, Markus. JavaFX: Nová dimenze pro web. In: *Chip online* [online]. 2009 [cit. 2012-10-31]. Dostupné z URL: <http://earchiv.chip.cz/cs/earchiv/vydani/r-2009/javafx.html>.

[9] Hardware and Software. Engineered to Work Together. In: *Oracle* [online]. 2012 [cit. 2012-10-31]. Dostupné z URL: <http://www.oracle.com/us/sun/index.html>.

[10] JavaFX Overview. In: *Oracle* [online]. [cit. 2012-10-31]. Dostupné z URL: <http://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm>.

[11] Backend. In:*Adaptic* [online]. [cit. 2012-10-31]. Dostupné z URL: <http://www.adaptic.cz/znalosti/slovnicek/backend/>.

[12] JavaFX Scene Builder User Guide. In: *Oracle* [online]. [cit. 2012-10-31]. Dostupné z URL: <http://docs.oracle.com/javafx/scenebuilder/1/user_guide/jsbpub-user_guide.htm>.

[13] HANDY, Alex. Should JavaFX go Open Source. In: *DS Times* [online]. 2010, 23.07 [cit. 2012-11-01]. Dostupné z URL: <http://www.sdtimes.com/blog/post/2010/07/23/Should-JavaFX-go-Open-Source.aspx>.

[14] HOPKINS, Bruce. Three Reasons Why Your Next Java ME Mobile Application Should Include JavaFX Mobile. In: *Oracle* [online]. 2009 [cit. 2012-11-09]. Dostupné z URL: <http://www.oracle.com/technetwork/articles/javame/javafxmobile-javame-135476.html >.

[15] SLAVNIĆ, Saša. Technology Diagram. In: *JavaFX RIA Cookbook* [on-line]. 2010, 29.05 [cit. 2012-11-09]. Dostupné z URL: <http://javafx.retrocode.com/2010/05/technology-diagram.html>.

[16] SLAVNIĆ, Saša. Data Types And Operators. In: *JavaFX RIA Cookbook* [online]. 2010, 04.06 [cit. 2012-11-10]. Dostupné z URL: <http://javafx.retrocode.com/2010/06/data-types-and-operators.html>.

[17] Lesson 6: Operators. In: *Oracle* [online]. 2011 [cit. 2012-11-10]. Dostupné z URL: <http://docs.oracle.com/javafx/1.3/tutorials/core/operators/>.

[18] GAZÁREK, Tomáš. *Internetové aplikace v JavaFX* [online]. Brno, 2010 [cit. 2012-11-11]. Dostupné z URL: <http://is.muni.cz/th/143325/fi_m/diplomka.pdf>.Diplomová práce. Masarykova univerzita.

[19] Why Use FXML. In: *Oracle* [online]. 2012 [cit. 2012-11-11]. Dostupné z URL: <http://docs.oracle.com/javafx/2/fxml_get_started/why_use_fxml.htm#CHDCHIBE>.

[20] Oracle Releases New Java Updates - Java SE 7 Update 6, JavaFX 2.2 and JavaFX Scene Builder 1.0. In: *Oracle* [online]. 2012, 14.08 [cit. 2012-11-11]. Dostupné z URL: <http://www.oracle.com/us/corporate/press/1735645>.

[21] Oracle Releases Java SE 7 Update 4 and JavaFX 2.1. In: *Oracle* [online]. 2012, 26.04 [cit. 2012-11-11]. Dostupné z URL: <http://www.oracle.com/us/corporate/press/1735645>.

[22] SHLOEMI. Managed AddIn Framework, Add-ins and Extensible. In: *Automate your world* [online]. 2012 [cit. 2012-11-11]. Dostupné z URL: <http://www.shloemi.com/2012/05/managed-addin-framework-add-ins-and-extensible/>.

[23] What is Adobe Flex. In:*JavaBeat* [online]. 2008 [cit. 2012-11-23]. Dostupné z URL: <http://www.javabeat.net/2008/09/what-is-adobe-flex/>.

[24] The Scala Programming Language. In: *Scala* [online]. 2008, 09.08.2010 [cit. 2012-11-24]. Dostupné z URL: <http://www.scala-lang.org/node/25>.

[25] Introduction to JRuby and Rails on Oracle GlassFish Server. In: *Oracle* [online]. © 2010 [cit. 2012-11-24]. Dostupné z URL: <http://docs.oracle.com/cd/E19798-01/821-1760/intro/index.html>.

[26] Refactoring. In: *NetBeans* [online]. © 2012 [cit. 2012-11-24]. Dostupné z URL: <http://wiki.netbeans.org/Refactoring>.

[27] MARTINEK, David. Ladění pomocí debuggeru. In: *Ladění a testování programů* [online]. © 2006 [cit. 2012-11-24]. Dostupné z URL: <http://www.fit.vutbr.cz/ martinek/clang/debug.html>.

[28] AJAX – kde jsou hranice. In: *Snizekweb* [online]. 2005 [cit. 2012-11-24]. Dostupné z URL: <http://www.snizekweb.cz/clanky/ajax-kde-jsou-hranice/>.

[29] BERNARD, Borek. Adobe Flex - co je a co není. In: *Interval.cz* [online]. 2008 [cit. 2012-11-24]. Dostupné z URL: <http://interval.cz/clanky/adobe-flex-co-je-a-co-neni/>.

[30] Adobe Flash Professional CS6. In: *Adobe* [online]. © 2012 [cit. 2012-11-24]. Dostupné z URL: <http://192.150.16.67/cz/products/flash.html>.

[31] Adobe Flash Builder 4.6 Premium. In: *Adobe* [online]. © 2012 [cit. 2012-11-24]. Dostupné z URL: <http://www.adobe.com/products/flash-builder.html>.

[32] LACKO, Ľuboslav. Výhody a nevýhody technológie AJAX. In: *Itnews* [online]. 2009 [cit. 2012-11-24]. Dostupné z URL: <http://www.itnews.sk/tituly/infoware/2009-12-07/c130666–vyhody-a-nevyhody-technologie-ajax>.

[33] HULÁN, Radek. Adobe Flex / Air versus Microsoft SilverLight: Základní výhody Adobe Flex. In: *MyEgo* [online]. 2009 [cit. 2012-11-24]. Dostupné z URL: <http://myego.cz/item/adobe-flex-air-versus-microsoft-silverlight>.

[34] Silverlight - co je Silverlight. In: *Interval.cz* [online]. 2008 [cit. 2012-11-24]. Dostupné z URL: <http://interval.cz/clanky/silverlight-co-je-silverlight/>.

[35] Top vlastnosti. In: *Microsoft/web* [online]. © 2012 [cit. 2012-11-25]. Dostupné z URL: <http://www.microsoft.com/cze/web/silverlight/top-vlastnosti.aspx>.

[36] Windows Phone. In: *Microsoft/web* [online]. © 2012 [cit. 2012-11-25]. Dostupné z URL: <http://www.microsoft.com/cze/web/silverlight/windows-phone-7.aspx>.

[37] Getting Started With JavaFX Technology. In: *Oracle* [online]. © 2011, 2009 [cit. 2012-11-25]. Dostupné z URL: <http://docs.oracle.com/javafx/1.2/gettingstarted/javafx/index.html>.

[38] NOVÁK, Jiří. *Analýza RIA metod a technik* [online]. Brno, 2009 [cit. 2012-11-25]. Dostupné z URL: <http://is.muni.cz/th/98711/fi_m/DP_RIA_Novak.txt.>. Diplomová práce. Masarykova univerzita.

[39] Java FX 2 Ribbon Menu. In: *Introjava* [online]. 2012 [cit. 2012-12-02]. Dostupné z URL: <http://introjava.wordpress.com/author/introjava/>.

[40] KALKAN, Halil Ibrahim. A Simple Plug-In Library For .NET. In: *Code Project* [online]. 2011 [cit. 2012-12-06]. Dostupné z URL: <http://www.codeproject.com/Articles/182970/A-Simple-Plug-In-Library-For-NET>.

[41] FEDORTSOVA, Irina. Drag-and-Drop Feature in JavaFX Applications. In: *Oracle* [online]. © 2008 - 2012 [cit. 2012-12-06]. Dostupné z URL:<http://docs.oracle.com/javafx/2/drag_drop/jfxpub-drag_drop.htm>.