VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

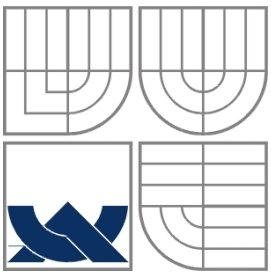
UŽIVATELSKÉ ROZHRANÍ PRO SYSTÉM DOLOVÁNÍ Z DAT

BAKALÁRSKA PRÁCA
BACHELOR´S THESIS

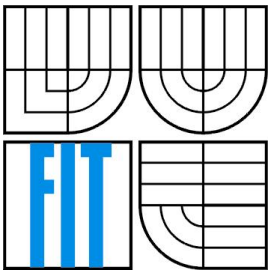
AUTOR PRÁCE
AUTHOR

TOMÁŠ MINÁR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

UŽIVATELSKÉ ROZHRANÍ PRO SYSTÉM DOLOVÁNÍ Z DAT

USER INTERFACE FOR DATAMINING SYSTEM

BAKALÁRSKA PRÁCA
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

TOMÁŠ MINÁR

VEDÚCI PRÁCE
SUPERVISOR

Ing. ZBYNĚK KŘIVKA, Ph.D.

BRNO 2009

Abstrakt

Práca sa zaoberá štúdiou grafických užívateľských rozhraní v pokročilých informačných systémoch s objektovo orientovaným dátovým modelom. Pre demonštračné účely je vytvorená aplikácia v jayzku Java SE pre systém hodnotenia hráčov hádzanárskeho tímu, ktorá kladie dôraz na rýchly a efektívny zber dát za pomoci GUI vytvoreného vo frameworku swing. Úložisko dát reprezentuje databáza db4objects, dolovanie dát zastupuje nástroj Weka.

Kľúčové slová

grafické užívateľské rozhranie, swing, objektovo orientovaný dátový model, dolovanie z dát

Abstract

This work deals with the problem of user interface in advanced informatics systems with object-oriented data model. For demonstration purposes I have created an application in Java SE language for ranking system of handball club players. This application insists on quick and effective data collecting using framework “swing”. Data store is represented by db4objects database and data mining is represented by learnig machine Weka.

Keywords

graphical user interface, swing, object oriented data model, data mining

Citácia

Tomáš Minár: Uživatelské rozhraní pro systém dolování z dat, bakalářská práce, Brno, FIT VUT v Brně, 2009

Uživatelské rozhraní pro systém dolování z dat

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Zbyněk Křivka, Ph. D..
Ďalšie informácie mi poskytol p. Kelemen Štefan.
Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

……
Tomáš Minár
18.05.2008

Poďakovanie

Chcel by som vysloviť poďakovanie vedúcemu práce Ing. Zbyňkovi Křivkovi, Ph.D. za prejavené úsilie pri vzniku tohoto zadania a odborné rady, ktoré mi pri tvorbe tejto práce poskytol. Ďalej by som chcel poďakovať p. Štefanovi Kelemenovi, ktorý pôsobí v trénerskom prostredí slovenskej hádzanej, za špecifikáciu a požiadavky kladené na systém hodnotenia hráčov a konzultácie na danú problematiku.

Obsah

1 Úvod

Práca v jednotlivých kapitolách priblíži princípy využitia návrhových vzorov (Design Patterns) pri tvorbe softvérových systémov vo svete informačných technológií. Rozoberie spôsoby perzistencie objektových modelov, priblíži problematiku objavovania znalostí v databáze, ktorou sa zaoberá dolovanie z dát (data mining), popíše správne metódy vývoja grafického užívateľského rozhrania. Cieľ práce je vytvoriť užívateľské rozhranie pre efektívny zber dát. V našom prípade poslúži ako model hodnotenie hráčov hádzanárskeho tímu v zápase. Následne špecifikovať požiadavky kladené na systém užívateľom,zanalyzovať ich a nakoniec vytvoriť systém pre hodnotenie hráčov hádzanárskeho tímu a otestovať ho v reálnych podmienkach.

2 Teoretický úvod

Rozvoj informačných technológií zaznamenáva v dnešnej dobe rozmach vo všetkých odvetviach. Vznikajú tak rôzne užívateľské rozhrania na zber rôznych údajov, ktoré produkujú nespočetné množstvo dát. Kapitola nás prevedie základnými pojmami a definíciami, ktoré nás uvedú do problematiky práce.

2.1 Princípy využitia návrhových vzorov

Návrhové vzory (design patterns), ďalej iba NV, predstavujú všeobecné riešenia problémov, ktoré môžu byť použité pri rôznych situáciách, v našom prípade hlavne pri tvorbe programu. NV nepochádzajú zo softvérového inžinierstva, ale z bežného života. Ide o takzvanú šablónu, ktorá má slúžiť ako návod na riešenie problémov určitého druhu. Preto správny výber a použitie NV v budúcnosti minimalizuje starosti spojené s údržbou a prípadným rozšírením programu, obvykle rozsiahleho systému. Dobrý návrh by mal ochrániť pred neprekonateľnými prekážkami, ktoré by si vyžiadali napr. zmenu celého systému.

V objektovo orientovanom svete NV ukazujú hlavne vzťahy medzi triedami a objektami, ale neurčujú konkrétnu implementáciu. Popisujú ako a kedy objekty vytvárať, aké štruktúry a vzťahy majú triedy obsahovať a ako majú medzi sebou spolupracovať. Použitie jedného návrhového vzoru nám nepokryje celé riešenie problému a preto často používame niekoľko NV súčasne. GoF, ako sa nazýva skupinka Gang of Four (Gamma, Helm, Johnson a Vlissides) vytvorili katalóg 23 vzorov v 3 kategóriách, ktorý popisuje tabuľka 2.1. V súčastnosti sa k nim pripojilo množstvo ďalších NV, ktoré majú často špecializované použitie.

	Creational Patterns *Vytváracie vzory*	**Structural Patterns** *Štrukturálne vzory*	**Behavioral Patterns** *Vzory chovania*
Trieda	Factory Method	Adapter	Interpreter Template
Objekt	Abstract Factory Builder Prototype Singleton	Bridge Composite Decorator Facade Proxy Flyweight	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

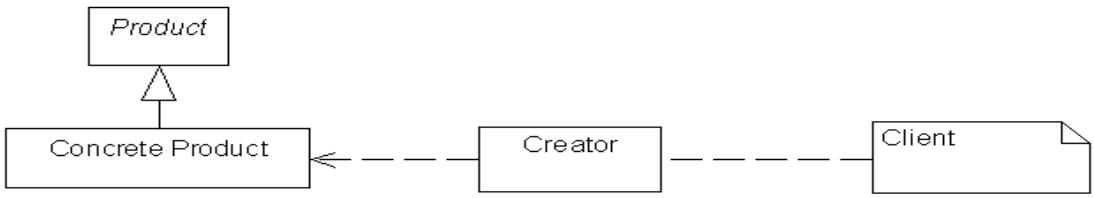
Tabuľka 2.1: Kategórie návrhových vzorov GoF

2.1.1 Creational Patterns

Vytváracie vzory riešia problémy spojené s vytváraním objektov v systéme. Ich úlohou je popísať postup pri výbere tried nového objektu a zaručenie správneho počtu týchto objektov. Väčšinou sa jedná o dynamické rozhodnutia vykonané za behu programu. [1]

2.1.1.1 Factory Method

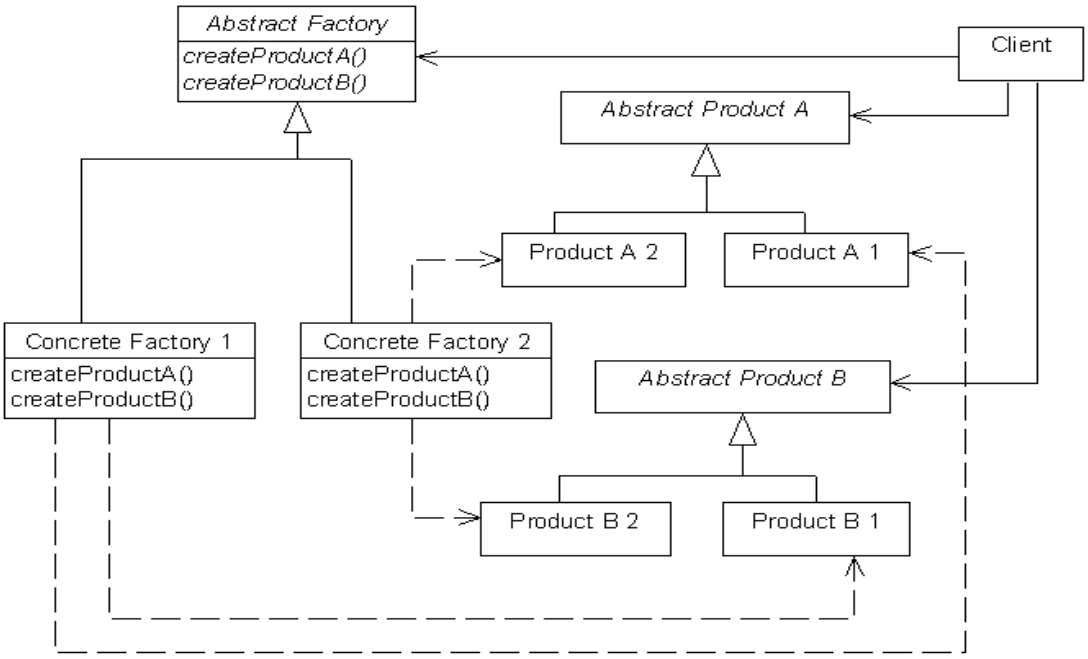
Factory Method Pattern sa stará o spôsob vytvorenia inštancií podriadených tried. Rozhoduje sa, či vytvoriť alebo použiť už vytvorený nepoužívaný objekt. Vytváranie objektov je preto uzavrené do samostatnej triedy. Schéma je na obrázku 2.1.1.1. Napr. v .NET technológii ako je MSDN.[1]



Obrázok 2.1.1.1 Factory Method Pattern

2.1.1.2 Abstract Factory

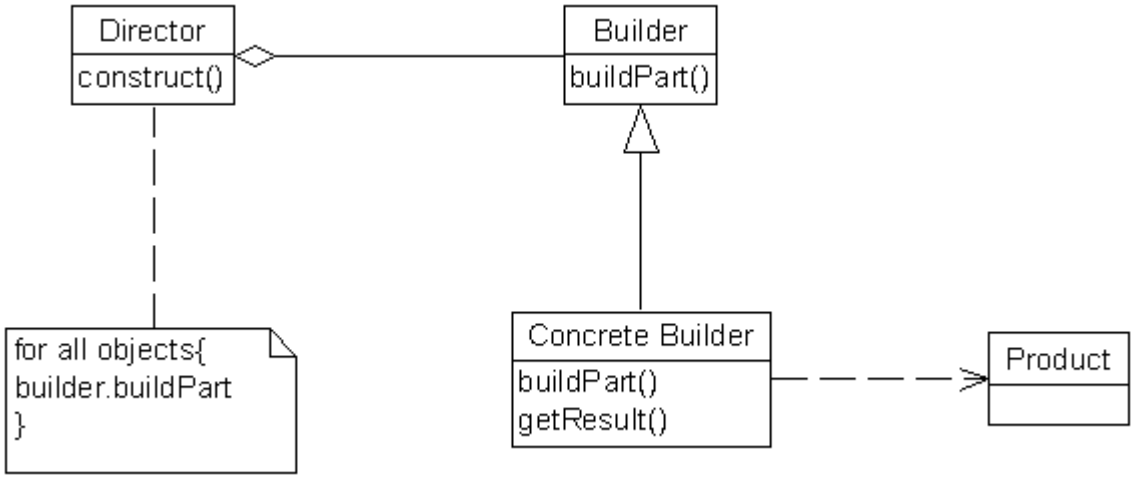
Abstract Factory Pattern rieši spôsob vytvorenia za behu programu inštancie triedy, ktorá ďalej tvorí inštancie súvisiacich tried. Popisuje obrázok 2.1.1.2. Typický predstaviteľ tohto vzoru Java AWT. [1]



Obrázok 2.1.1.2 Abstract Factory Pattern

2.1.1.3 Builder

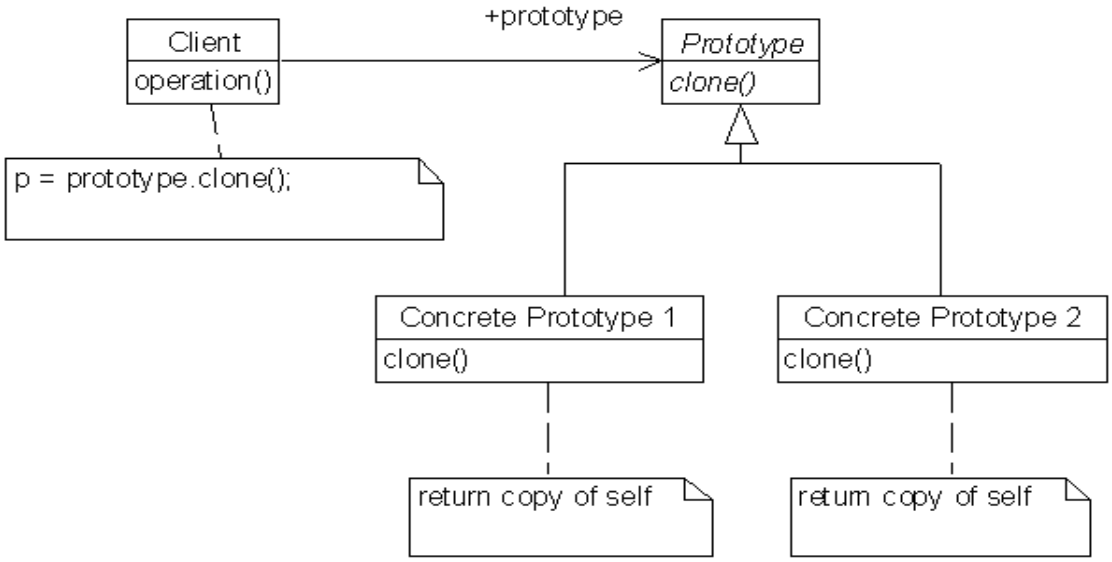
Oddeľuje vytváranie objektov od ich prezentácie. Jedná sa hlavne o zložité objekty, ktoré majú podobný spôsob konštrukcie. Primárne je určený k vytváraniu rozdielnych a nezávislých objektov. Znázornené na obrázku 2.1.1.3. Typickým použitím sú JavaBeans. Builder nepozná, či budú použité a ako ich poskladať. Stačí vedieť, akým spôsobom ich vytvoriť a pracovať s nimi. [1]



Obrázok 2.1.1.3 Builder Pattern

2.1.1.4 Prototype

Vzor pre vytváranie kópie existujúceho objektu namiesto novej triedy, ktorého vytvorenie by bolo časovo alebo zdrojovo náročné. Preto musí byť programovací jazyk schopný klonovať - vytvárať kópie už existujúceho objektu. Ukážka na obrázku 2.1.1.4. [1]



Obrázok 2.1.1.4 Prototype Pattern

2.1.1.5 Singleton

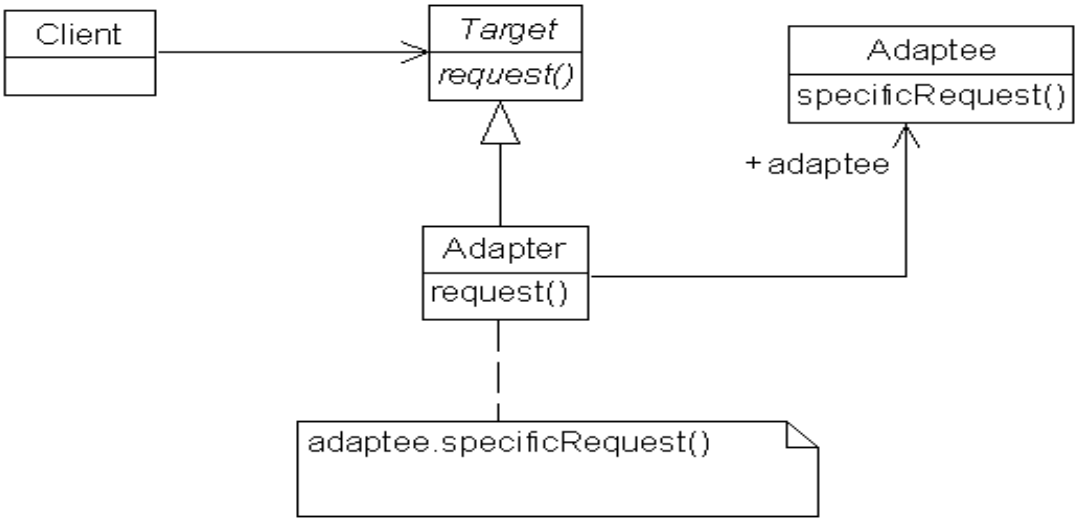
Singleton zabezpečuje existenciu iba jednej inštancie danej triedy a poskytuje globálny prístup k nej. Využíva sa často pri implementácii Abstract Factory, Builder, Prototype. Dobrý príklad pre použitie je riadenie prístupu do databázy, aby sme zamedzili plytvaniu zdrojov. [1]

2.1.2 Structural Patterns

Snažia sa sprehľadniť systém možnosťou usporiadania tried alebo komponent za pomoci štrukturalizácie kódu. To znamená, ako ich poskladať do väčších štruktúr s využitím dedičnosti rozhrania alebo implementácie. [1]

2.1.2.1 Adapter

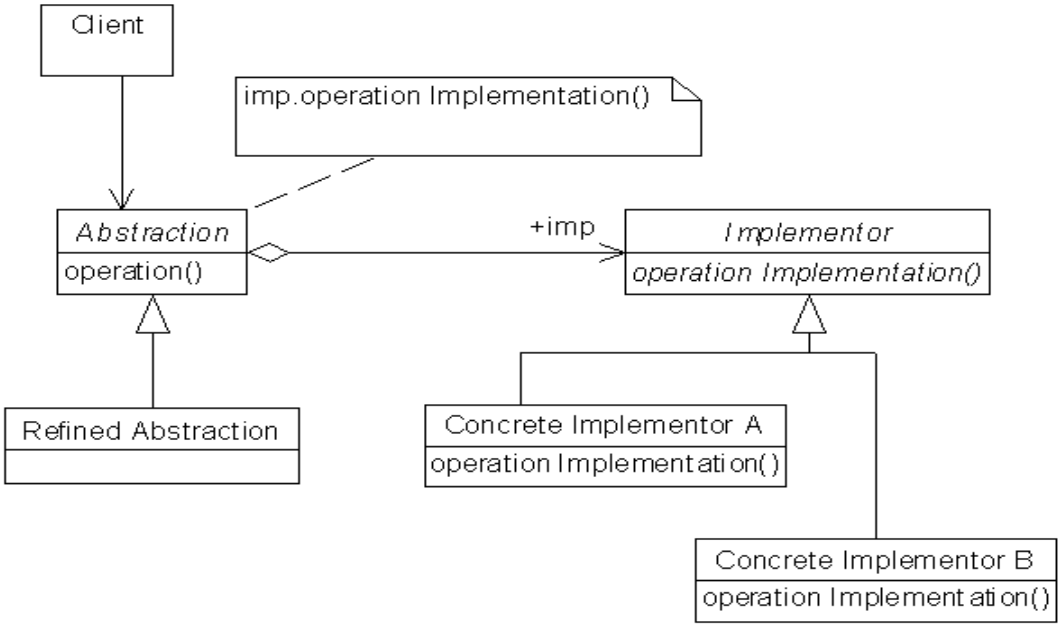
Názov Adapter už napovedá, že sa jedná o prispôsobenie určitej triedy, aby ju bolo možné využiť aj iným spôsobom. Zvyčajne sa rozhranie jednej triedy prevedie na rozhranie požadované druhou triedou. Na úrovni tried sa nová odvodí od starej a pridajú sa nové metódy. Na úrovni objektov referencia na Adaptee v adaptéri a delegovanie požiadavkov na metódy Adaptee objektu. [1]



Obrázok 2.1.2.1 Adapter Pattern

2.1.2.2 Bridge

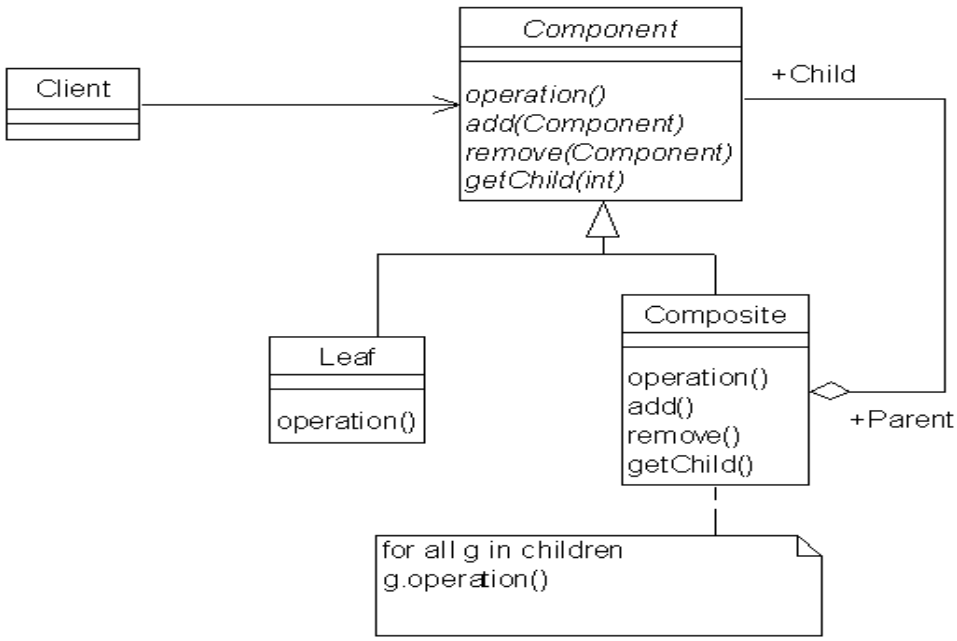
Definuje rozhranie triedy a tým oddelí jej vlastnú implementáciu. Hlavná výhoda tohto vzoru je možnosť menenia implementácie tried bez potreby menenia kódu na strane klienta. Popisuje obrázok 2.1.2.2. [1]



Obrázok 2.1.2.2 Bridge Pattern

2.1.2.3 Composite

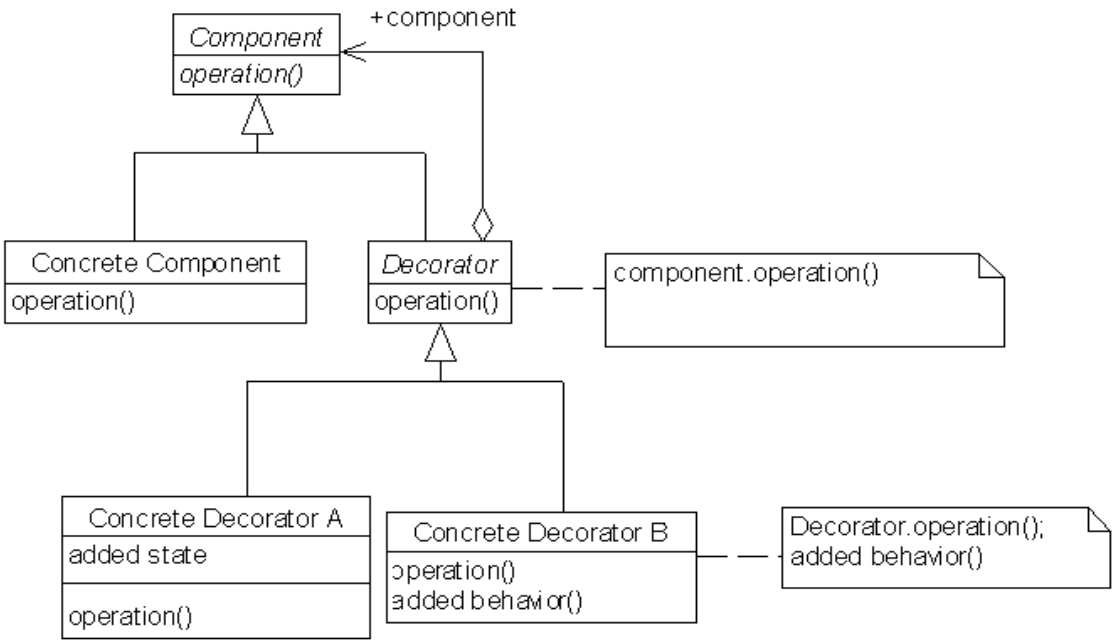
Composite Pattern hovorí, ako usporiadať jednoduché objekty a z nich zložené, aby prístup k nim bol jednotný. Tento návrhový vzor sa využíva napr. pri tvorbe GUI. [1]



Obrázok 2.1.2.3 Composite Pattern

2.1.2.4 Decorator

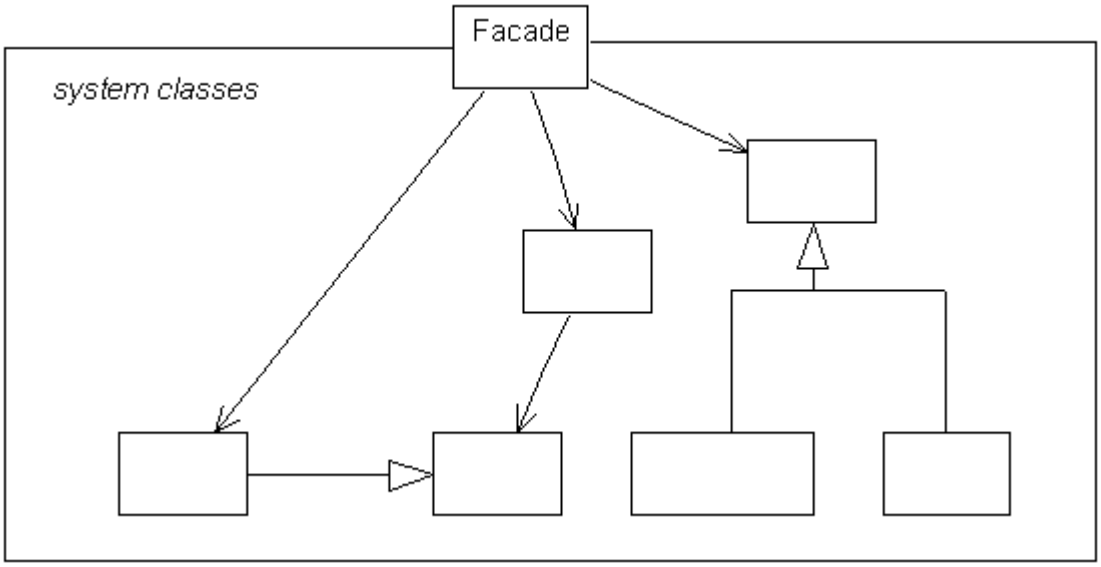
Poskytuje riešenie problému, ako meniť vlastnosti inštancie triedy a nemusieť pri tom vytvárať novú odvodenú triedu. Decorator sa snaží o zmenu chovania objektu. [1]



Obrázok 2.1.2.4 Decorator Pattern

2.1.2.5 Facade

Obvykle sa používa pri nutnosti zjednodušiť vstupný bod do systému. Rozsiahle systémy majú obvykle veľmi zložitú štruktúru a sú ťažko zvládnuteľné na pochopenie. Preto Facade nám poslúži na zjednodušenie komunikácie medzi klientom a systémom. [1]

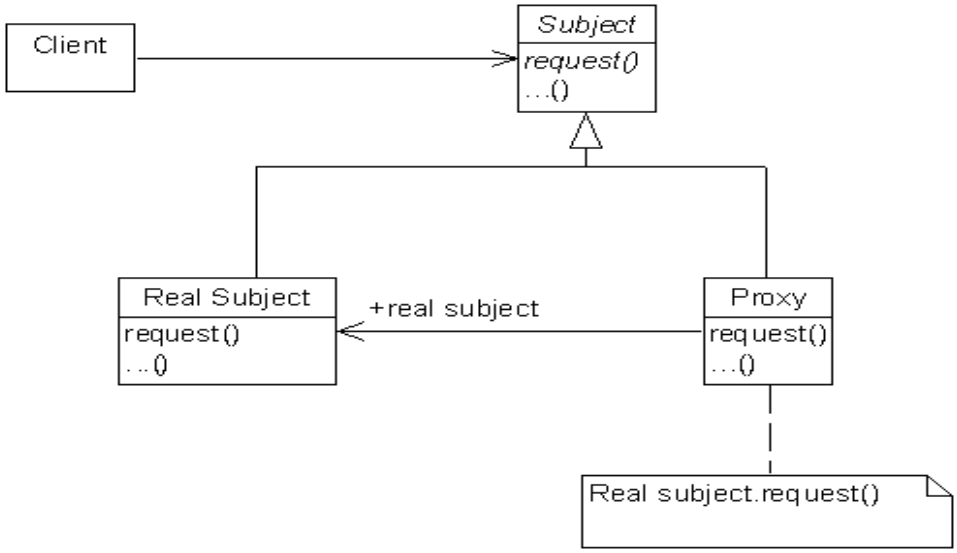


Obrázok 2.1.2.5 Facade Pattern

2.1.2.6 Proxy

Tento vzor je použiteľný, ak treba zaistiť kontrolu nad iným objektom. Zastupuje daný objekt, kontroluje a spravuje prístupy k tomuto objektu (Popisuje obrázok 2.1.2.7). Môžeme ho rozdeliť do 4 druhov. [1]

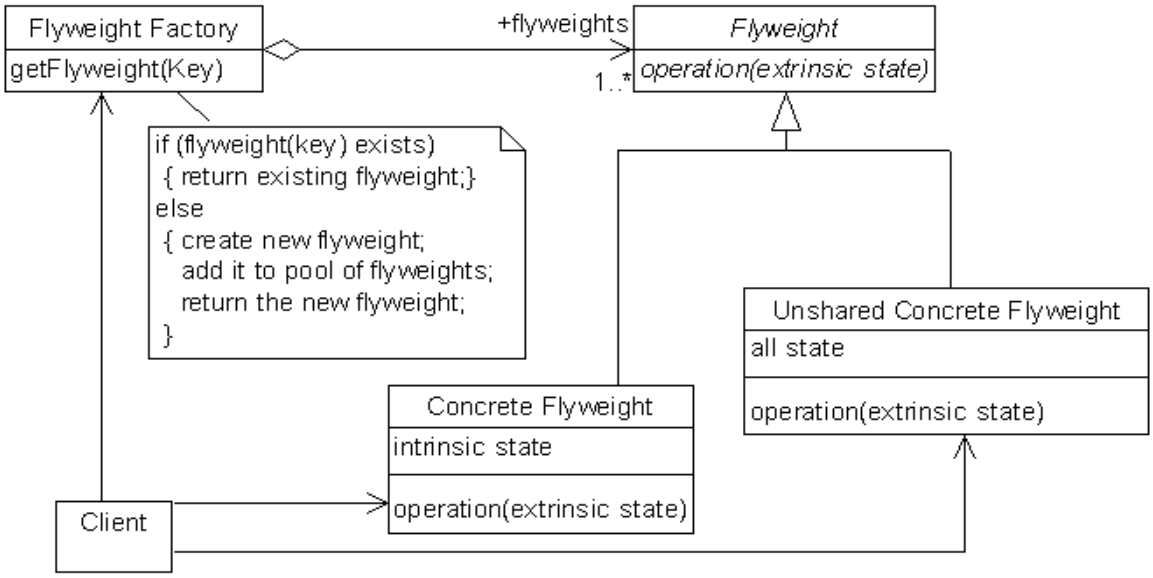
- *Virtual* vytvorenie reálneho objektu je náročné, vytvoriť ho až v prípade potreby
- *Remote* lokálna interpretácia vzdialeného objektu, napr. v inom adresovom priestore
- *Protective* kontrola prístupu k zapúzdreniu objektu (kontrola práv používania objektu)
- *Smart* poskytuje dodatočné operácie pred volaním objektu (vzdialený objekt, na ktorý neexistujú referencie, sa môže uvolniť z pamäte alebo ak je objekt uložený na perzistentnom médiu, zaistí jeho nahratie do pamäte a taktiež môže zaisťovať správu zámkov, aby objekt nemohol byť používaný viacerými klientmi zároveň)



Obrázok 2.1.2.6 Proxy Pattern

2.1.2.7 Flyweight

Jeho úlohou je zaistiť efektívnu správu veľkého množstva podobných objektov, čím ušetríme pamäťové nároky, napr. pri textovom editore. Preto je potreba objekty patriace do rovnakého druhu vytvoriť len raz a umožniť ich zdieľanie. [1]



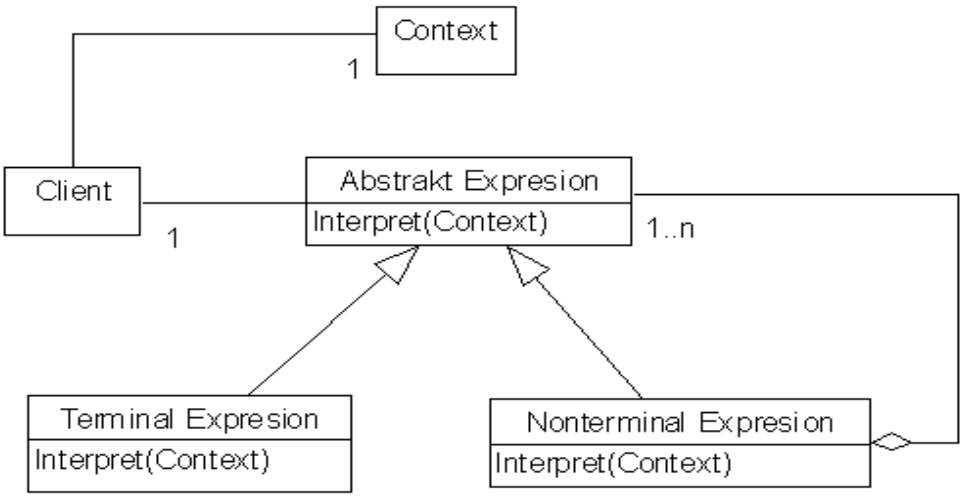
Obrázok 2.1.2.7 Flyweight Pattern

2.1.3 Behavioral Patterns

Zaoberajú sa chovaním systému. Ich základ tvoria triedy alebo objekty. Pri triedach sa využíva princíp dedičnosti, v druhom prípade je to rozdelenie funkčnosti a zodpovednosti medzi objektami, aby sme dosiahli požadovaný výsledok. [1]

2.1.3.1 Interpreter

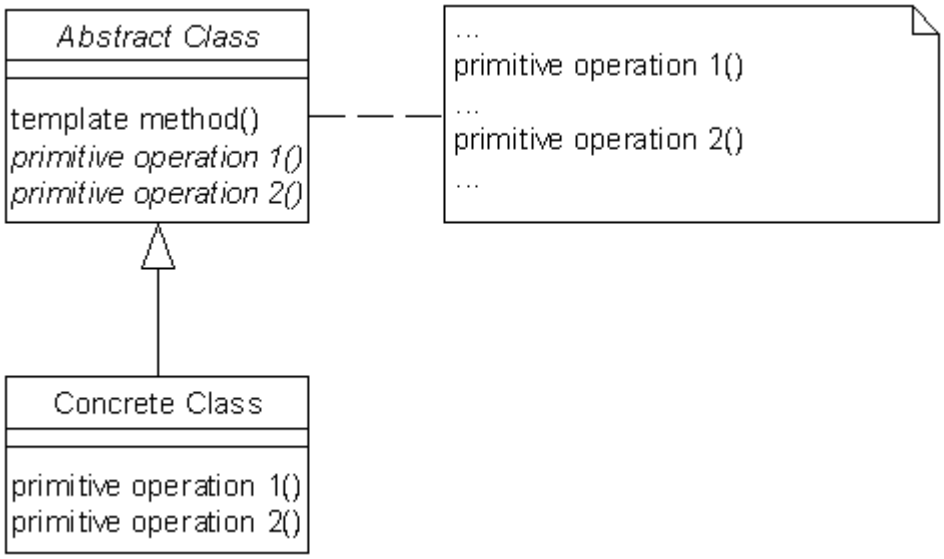
Návrhový vzor popisuje vytvorenie jazyka. Preto musí definovať gramatické pravidlá a určenie spôsobu, ako vzniknutý jazyk interpretovať. Použitie je napr. pri možnosti pohodlného prechádzania databázy užívateľom tým, že SQL jazyk nahradíme jazykom, ktorý má blízko reči. [1]



Obrázok 2.1.3.1 Interpreter Pattern

2.1.3.2 Template

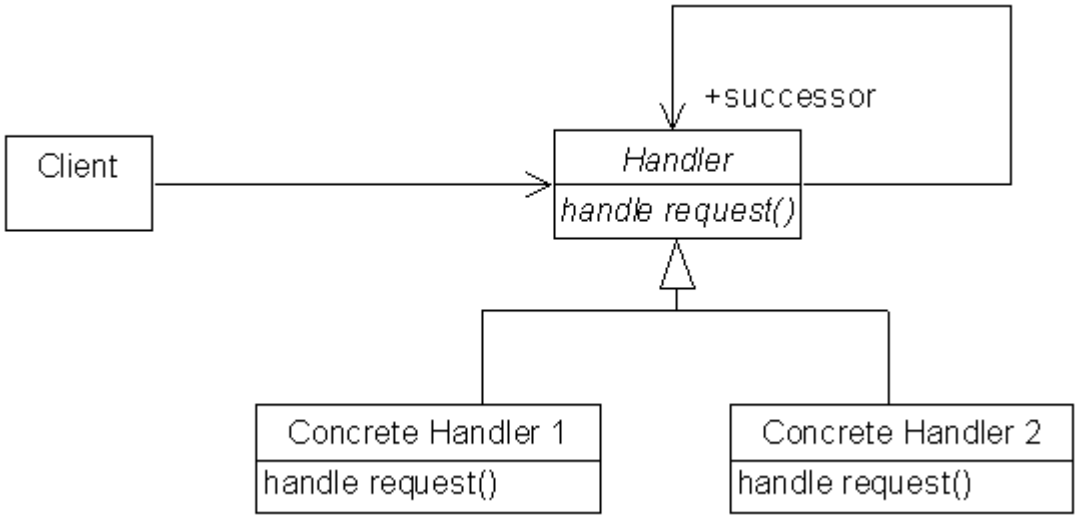
Rieši problém, ako zaistiť, aby sme mohli ovplyvniť kroky algoritmu, ktorý je implementovaný v triede predka pomocou zdedených tried. Napr. java.awt.Componjent. Kde template metóda `update()` využíva metódu `paint()`, ktorá je definovaná ako primitívna metóda v potomkoch. [1]



Obrázok 2.1.3.2 Template Pattern

2.1.3.3 Chain of Responsibility

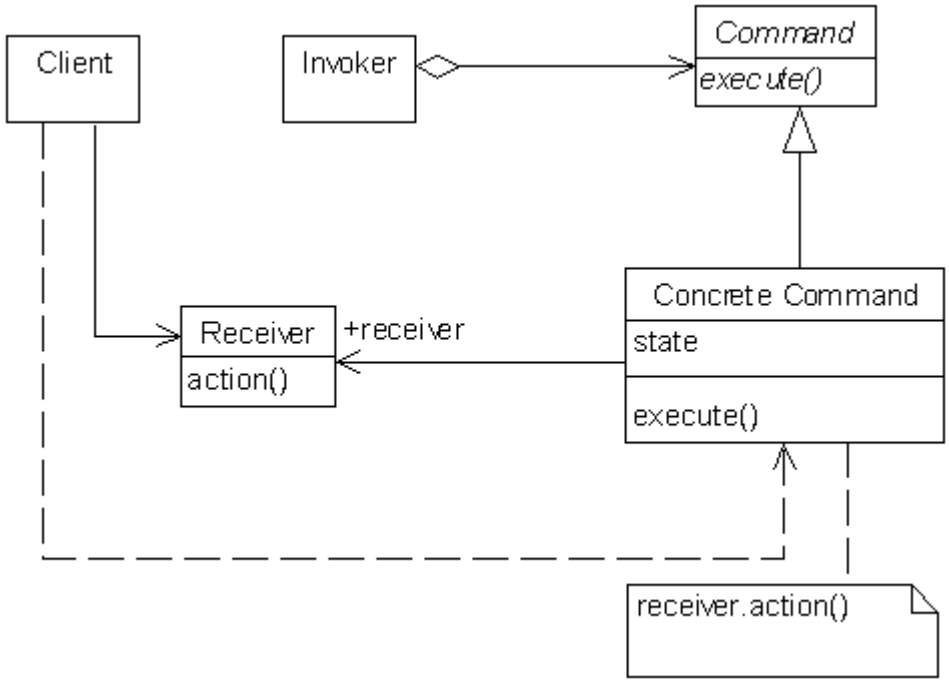
V preklade znamená reťaz(ec) zodpovednosti. Umožňuje zasielanie správ neznámym príjemcom, ktorý tvoria frontu a správa sa predáva, až pokiaľ ju niekto nespracuje. Umožňuje zrušenie väzby medzi odosielateľom a príjemcom. Najlepším príkladom tohto návrhového vzoru je dedičnosť v OO jazykoch. Ak je volaná funkcia na triede, ktorá ju neobsahuje, dochádza k hľadaniu na zdedených. [1]



Obrázok 2.1.3.3 Chain of Responsibility Pattern

2.1.3.4 Command

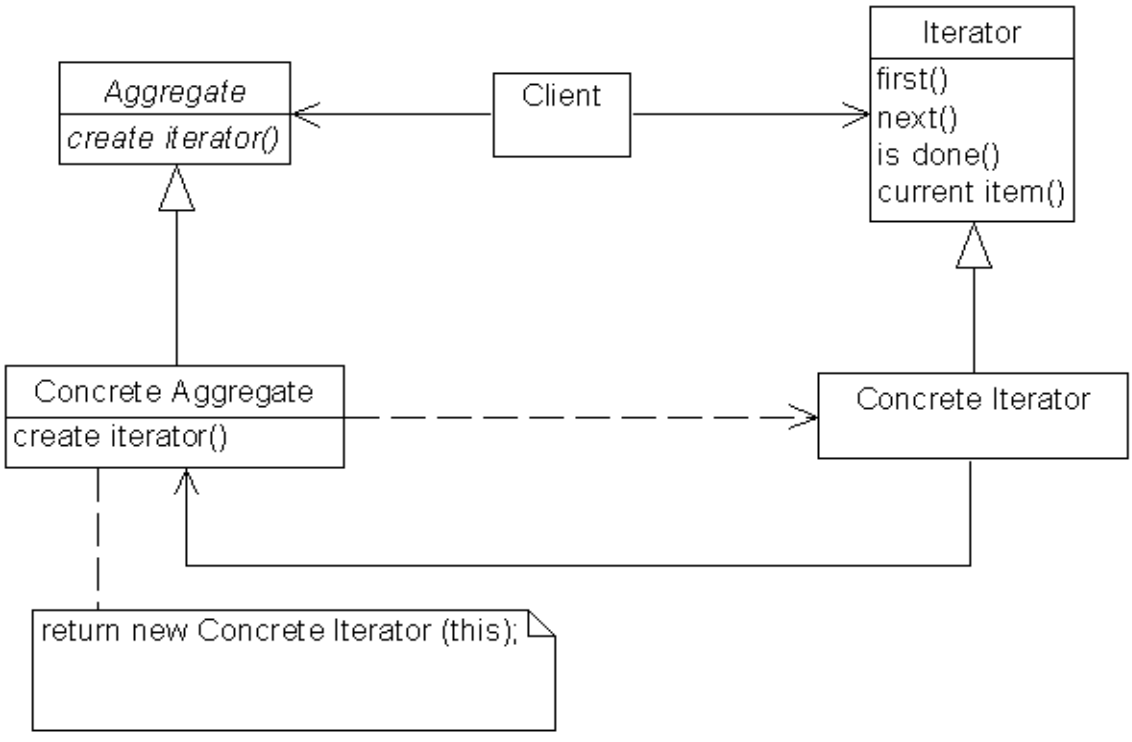
Má za úlohu odtieniť klienta od procesu spracovania požiadavku. Klient ho iba definuje a určí, kto bude spracovateľom a nezaujíma sa o čas a spôsob realizácie. Požiadavok môžeme chápať ako objekt splňujúci určité rozhranie. Použite napr. vo swingu UNDO / REDO operácie. [1]



Obrázok 2.1.3.4 Command Pattern

2.1.3.5 Iterator

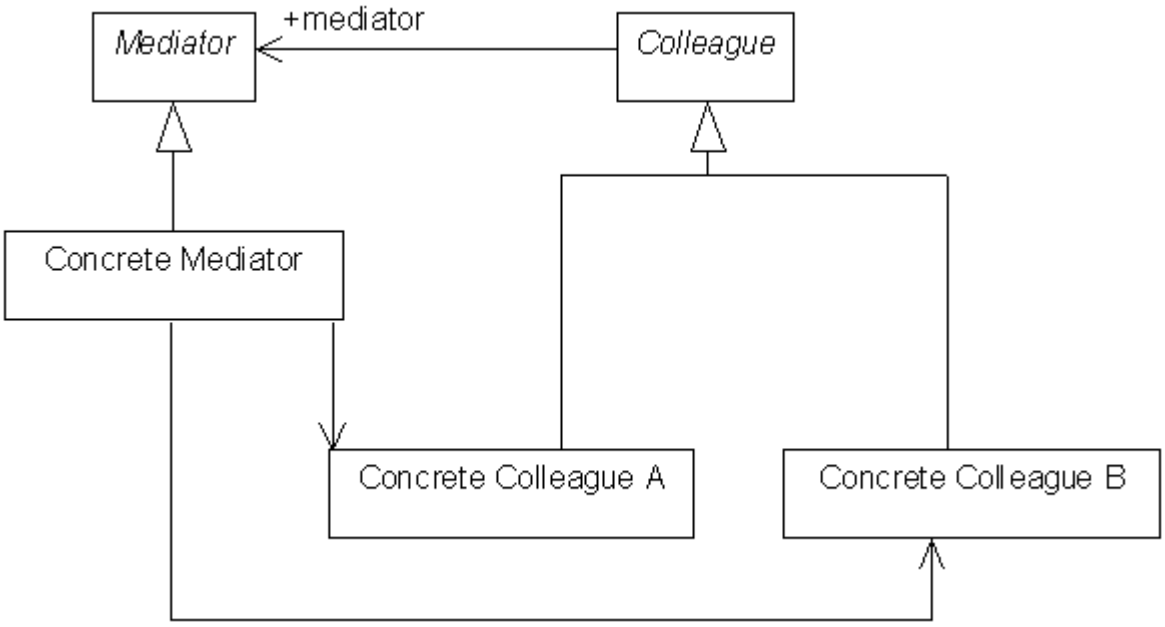
Najjednoduchší a najpoužívanejší NV. Definuje, ako sa pohybovať medzi prvkami, ktorých usporiadanie je sekvenčné. Implementáciu jednotlivých prvkov nemusíme poznať, pretože iterácie nezaisťuje priamo ich zoznam. Popisuje obrázok 2.1.3.5. Použitie java.util.Iterator. [1]



Obrázok 2.1.3.5 Iterator Pattern

2.1.3.6 Mediator

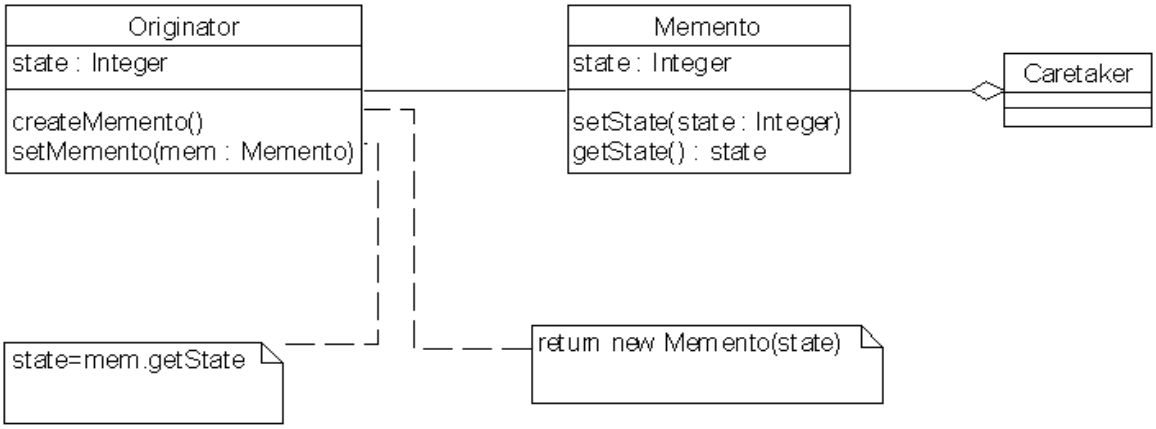
Zapúzdruje logiku komunikácie viacerých tried do jednej. Rieši komunikáciu dvoch komponént programu bez toho, aby museli presne poznať poskytované metódy. [1]



Obrázok 2.1.3.6 Mediator Pattern

2.1.3.7 Memento

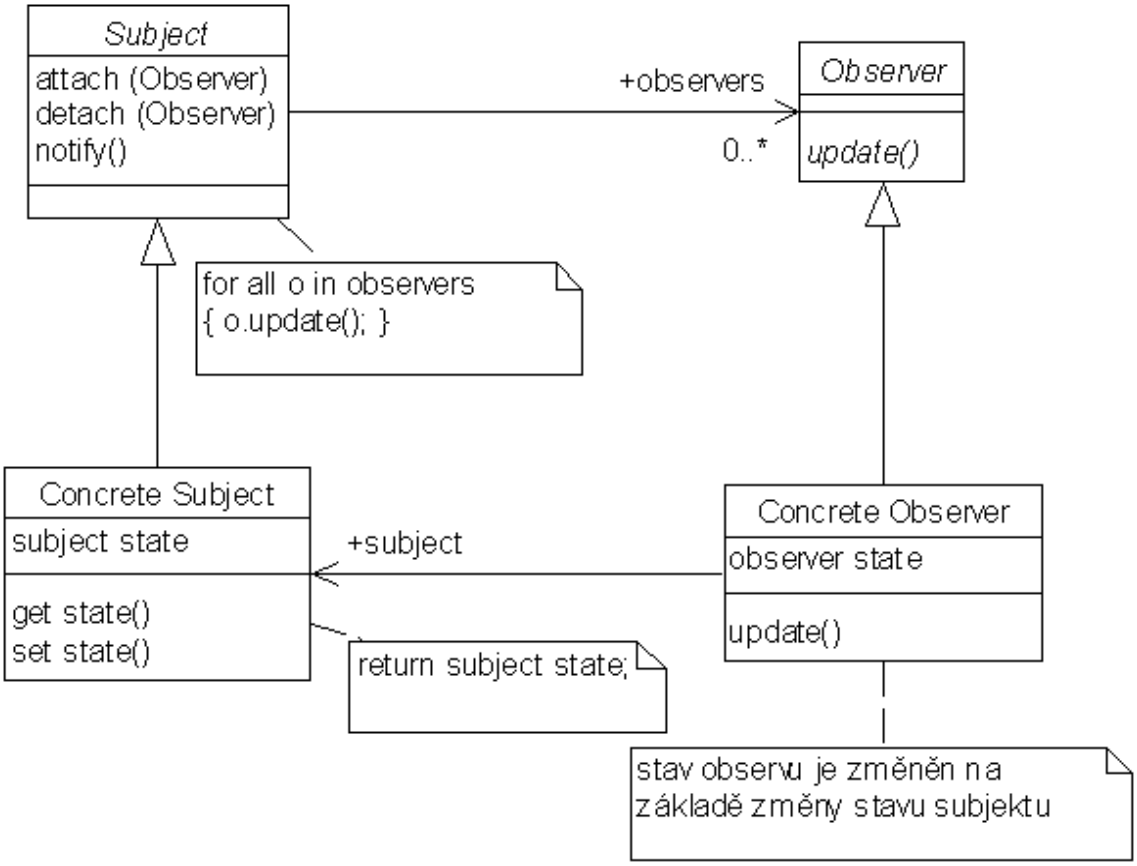
Rieši problém, ako uložiť stav objektu bez toho, aby sa narušilo jeho zapúzdrenie, aby sme uložený stav mohli znovu obnoviť. [1]



Obrázok 2.1.3.7 Memento Pattern

2.1.3.8 Observer

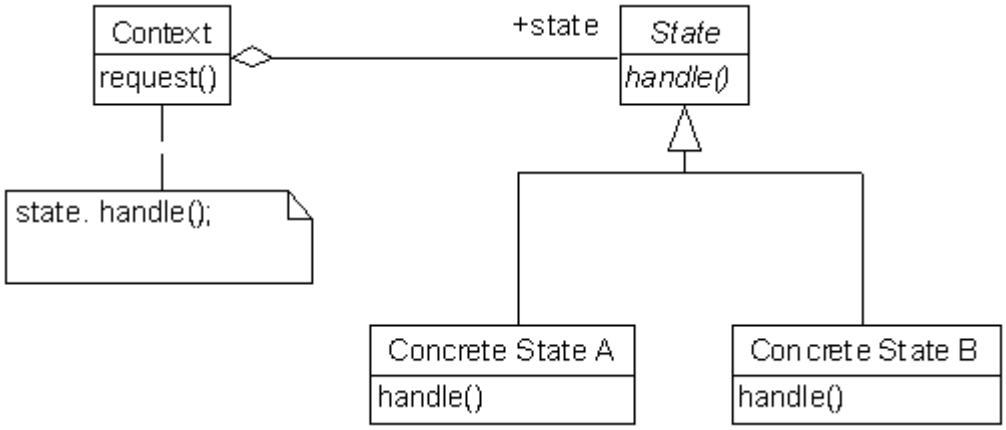
Princíp spočíva k umožneniu šírenia udalostí, ktoré nastali na jednom objekte k ostatným podriadeným. Príklad je napríklad listener v jave. [1]



Obrázok 2.1.3.8 Observer Pattern

2.1.3.9 State

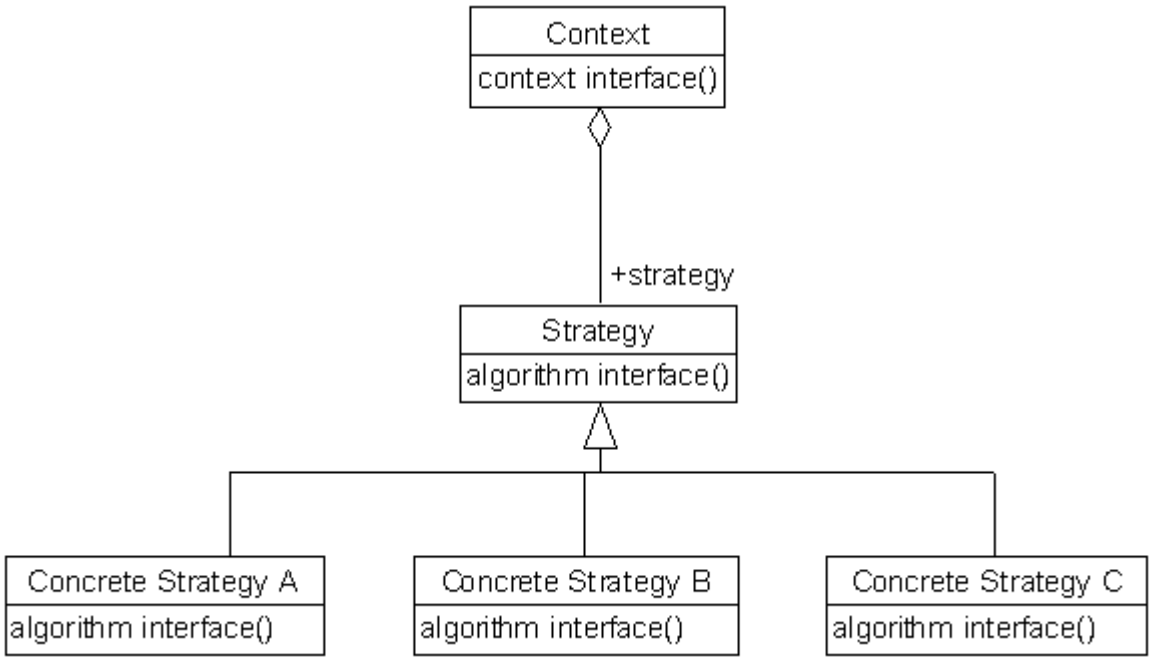
NV State rieši problém, ktorý nastáva pri potrebe zmeniť chovanie objektu, ak sa zmení jeho vnútorný stav. Po zmene sa objekt tvári ako inštancia triedy novej. [1]



Obrázok 2.1.3.9 State Pattern

2.1.3.10 Strategy

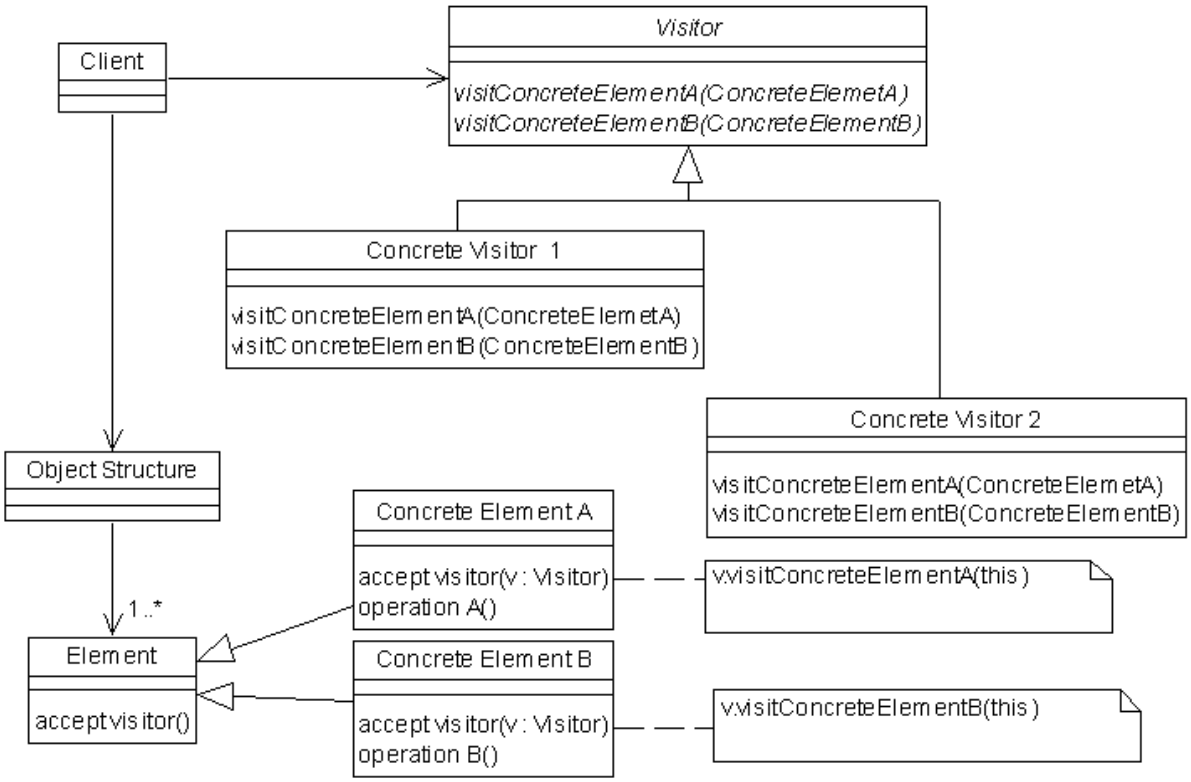
Strategy Pattern sa používa, ak pri riešení rovnakého problému existuje viacero podobných možností ich vyriešenia nasledujúcim spôsobom. Určíme množinu algoritmov (spôsoby riešenia), zapúzdrime ich do samostatného objektu a a umožníme ich zámenu. Výber je nezávislý od klienta. Napr. Kontajner za pomoci layout manageru.



Obrázok 2.1.3.10 Strategy Pattern

2.1.3.11 Visitor

Pri NV je potrebné vytvoriť metódu, ktorá bude pracovať s viacerými objektami, ktoré patria do objektovej usporiadanej štruktúry. Pri vytváraní novej metódy by nemalo dôjsť k zmenám v objektoch, s ktorými pracujeme. [1]



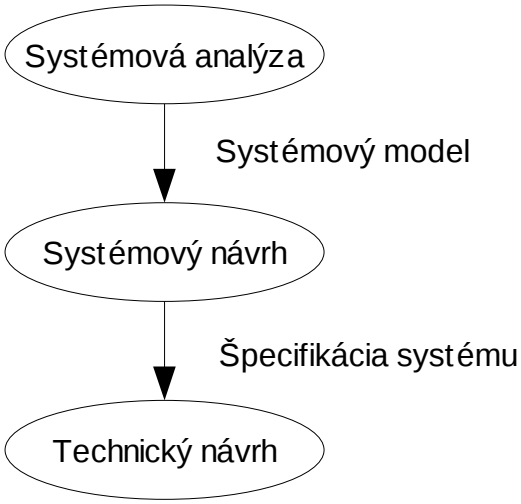
Obrázok 2.1.3.11 Visitor Pattern

2.2 Perzistencia objektových modelov

Základnou myšlienkou OO programovania je, aby metódy a dáta tvorili jeden celok. Zároveň nastáva problém, ako objekty udržať v použiteľnej forme aj po dobe jedného spustenia programu. Preto je zapotreby vytvoriť dátový model systému. Data model je abstraktný model, ktorý popisuje, ako sú dáta reprezentované a sprístupnené. Termín dátový model má dva základné významy. Teória dátového modelu a inštancia dátového modelu.

- **teória dátového modelu** (data model theory), ktorá formálne opisuje, ako majú byť dáta štrukturované a sprístupnené. Možme sem zaradiť databázové modely, ktoré v práci neskôr rozoberieme
- **inštancia dátového modelu** (data model instance) aplikuje teóriu dátového modelu pre vytvorenie skutočného modelu pre jednotlivú aplikáciu.

Návrh dátového modelu sa zvyčajne vykonáva v troch krokoch. Systémová analýza, systémový návrh a poslednou fázou je samotný technický návrh. Proces tvorby je zobrazený na obrázku 2.2.



Obrázok 2.2: Proces tvorby informačných systémov

Aby sa zachovala perzistentnosť, aby po uložení a znovunačítaní objety nestratili svoje vlastnosti a vzťahy, je potrebné zvoliť vhodný nástroj. Pre jazyk java je najdostupnejším a najrozšírenejším spôsobom uchovania dát objektovo relačné mapovanie (Object – relational mapping ORM), ktoré rozloží objekty do tabuliek relačnej databázy, pričom si zachovajú svoje vlastnosti – perzistentnosť. Ďalšou možnosťou je použiť priamo objektovo orientovanú databázu, ktoré v poslednej dobe zaznamenali veľký posun smerom vpred. [2]

2.2.1 Objektovo relačné mapovanie

Programovacia technika, ktorá spája objektovo orientovaný jazyk s relačnou databázou. Vzniká tak možnosť uloženia objektov do relačnej databázy za pomoci ORM nástroja, ktorý vystupuje ako samotné úložisko objektov. Musia byť definované relačné metadáta a to väčšinou buď pomocou mapovacích súborov alebo vytvorením anotácie. Mapovacie súbory sú vo formáte xml, a umiestňujú sa do rovnakého adresára, kde sa nachádza zdrojový kód triedy, ktorú chceme namapovať. Anotácie sa používajú pri jednoduchších triedach a umiestňujú sa priamo do zdrojového kódu.

2.2.1.1 Java Persistence API

API definuje balíček javax.persistence a snaží sa štandardizovať ORM, ale nevyžaduje EJB kontajner, čo umožňuje použiť ho v rámci Java SE (Standart Edticion) prostredia. V dnešnej dobe existuje viacero implementácií (Hibernate, Toplink).

2.2.1.2 Java Data Objects

JDO je štandard pre objektovú perzistenciu. JDO 1.0 bolo vyvíjané pod JCP (Java Comunity Process) ako JSR 12, neskôr JDO 2.0 pod JSR 243. JDO 2.1 je teraz pod krídlami Apache.

2.2.1.3 Hibernate

Voľne šíriteľný open source software pod GNU Lesser General Public liecenciou. Poskytuje veľmi dobré ORM mapovanie a v súčastnosti je jedným z najpoužívanejších nástrojov.

2.2.1.4 TopLink

ORM nástroj z dielne Oracle, je aj v open source verzii pod označením TopLink Essentials, ktorý poskytuje JPA funkcionalitu pre EJB 3.0.

2.2.2 Objektovo orientované databázové systémy

V objektovo orientovanej databáze je informácia reprezentovaná vo forme objektov, akú používa objektovo orientované programovanie. Niektoré objektové databázy sú navrhnuté a optimalizované pre konkrétny OO jazyk ako je Python, Java, C#, Visual Basic, .NET, C++, Objective-C, Smalltalk. ODBMS používajú skutočne rovnaký model ako objektovo orientované jazyky.

Objektové aplikácie majú celkom iný model ako relačné databázy a preto je zapotreby podporovať celú radu krokov pri ich spolupráci. Vývoj OODBMS si môžme vysvetliť ako alternatívu a rozšírenie k relačným DBMS. Obidva spôsoby majú svoje pre a proti. Relačné databázy je výhodné použiť pri veľkom objeme pomerne jednoduchých dát
a naopak, OODBMS dobre vystihujú a zobrazujú vzťahy medzi objektami a možnosti vyjadrenia zložitosti modelovanej reality v databáze. Pri objektových databázach nám taktiež odpadá proces mapovania tried potrebných pre definíciu tabuliek pri ukladaní objektov do relačnej databázy.

2.2.2.1 Db4objects

Zastrešená open source licenciou s obmedzenou GNU General Public License pre komerčné účely. Db4o je perzistentná čisto objektová databáza, ktorá dokáže veľmi jednoducho pracovať aj s tými najzložitejšími objektami. Na databázovej úrovni sa neriešia prístupové práva, ale prenechávajú sa priamo na aplikáciu. Objekty sa ukladajú v rovnakom formáte, ako boli navrhnuté v OO jazyku. [3]

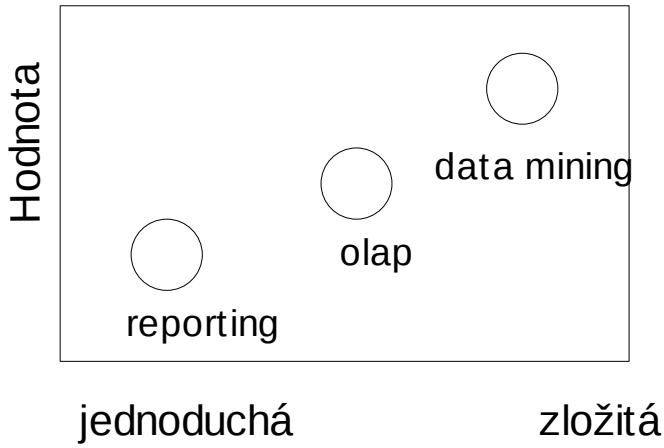
2.3 Dolovanie z dát

Dolovanie z dát (Data Mining), OLAP (On Line Analytical Processing), alebo taktiež hĺbková analýza dát predstavuje proces analýzy veľkého objemu dát, z ktorého má za úlohu vybrať

pre nás „cenné“ informácie za pomoci štatistiky, umelej inteligencie, matematiky a strojového učenia. Poznáme dva základné typy modelov: [4]

- **predikčné** sa snažia z historických dát predpovedať budúcnosť
- **segmentačné** triedia veľké objemy dát do zvládnuteľných homogénnych skupín

Na obrázku 2.1 je zobrazená použiteľnosť dolovania z dát na základe hodnoty.



Obrázok 2.3.1: Použiteľnosť

2.3.1 Úlohy dolovania

2.3.1.1 Klasifikácia

Objekt je charakterizovaný množinou, z ktorej je jedna kľúčová premenná. Za náš cieľ je nájsť taký model, ktorý opisuje kľúčovú premennú ako funkciu vstupných premenných – prediktorov. [5]

2.3.1.2 Regresia

Na prvý pohľad sa môže zdať podobná s klasifikačnou, ale je tu jeden rozdiel. Kľúčovú hodnotu reprezentuje číslo. Jednoduchý príklad pre regresný model môže byť predpoveď teploty v závislostí na rýchlosti vetra, tlaku a vlhkosti.[5]

2.3.1.3 Prognózovanie

Je tvorba klasickej prognózy pre vývoj určitého faktoru, napr. aký bude obrat podniku za mesiac, vývoj cien o týždeň. Vstupnými hodnotami je postupnosť čísel zaznamenaných na časovej osi, ktoré na sebe závisia. Snaží sa nastoliť vývoj trendov, cyklov určitých druhov za odfiltrovania nepotrebného šumu.[5]

2.3.1.4 Analýza sekvencií

Zaoberá sa diskrétnymi stavmi, akými je napríklad sekvencia kliknutí na web.[5]

2.3.1.5 Zhlukovanie

Vytvára prirodzené skupiny objektov, ktoré charakterizujú určité premenné. Objekty v rámci jednej skupiny sú si podobné, ale skupiny sú navzájom rôzne.[5]

2.3.1.6 Analýza odchýlok

Zaoberá sa vyhľadaním netradičných objektov, ktoré sa výrazne líšia od ostatných. V praxi sa využívajú hlavne pri odhaľovaní napr. finančných podvodov a na detekciu defektov vo výrobkoch. Jednoduchý príklad môžme uviesť pri bankovom prevode, ktorý sa výrazne líši premennými od ostatných. [5]

2.3.1.7 Asociácia

Ukážkový príklad asociácie je analýza nákupného koša - nájdenie skupín výrobkov, ktoré sa často predávajú spoločne. Výsledkom je odhalenie asociačných pravidiel, napr. ak zákazník kúpi výrobok A a B, tak s pravdepodobnosťou 60 % kúpi aj výrobok C. Pre obchodné reťazce sú to veľmi dôležité informácie, na základe ktorých vedia nastaviť podmienky tak, aby získali čo najväčší obrat. To môžu docieliť tak, že výrobky, ktoré sa s veľkou pravdepodobnosťou predávajú spolu sa umiestnia v pultoch pri sebe. [5]

2.3.2 Techniky dolovania

2.3.2.1 Riadené

Tiež supervised alebo directed majú za úlohu pomocou jednej kľúčovej premennej riadiť proces učenia, aby vzorka ostatných premenných predpovedala v cieľovej skupine hodnotu kľúčovej hodnoty. Najčastejšie používané algoritmy:

- neurónové siete
- mechanizmy podobných vektorov
- logistická regresia
- lineárna regresia
- diskriminačná analýza
- rozhodovacie stromy
- bayesovské modely
- techniky najbližších susedov
- genetické algoritmy

– analýza prežitia

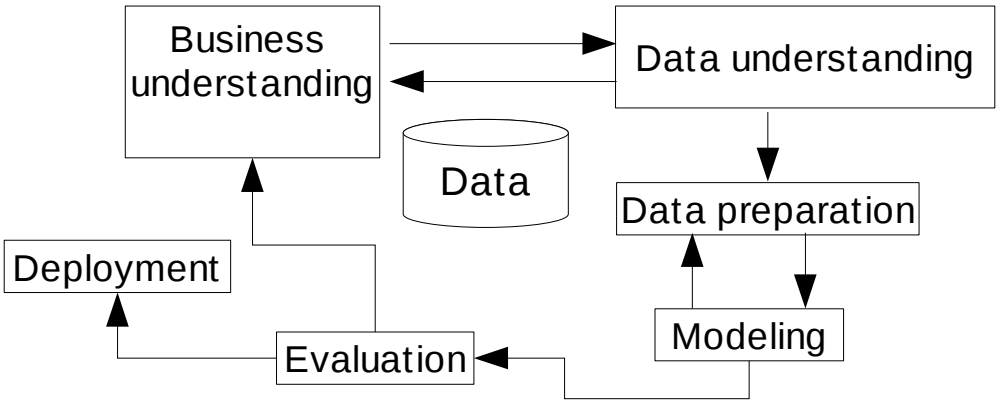
2.3.2.2 Neriadené

Všetky premenné modelu sú si navzájom rovnocenné, žiadna neriadi proces učenia a vyberá ich analitik podľa vlastného uváženia. Samotný algoritmus nám tak nepovie, ktoré premenné sú, resp. nie sú doležité. Cieľom tejto techniky je zisťovanie znalostí – knowledge discovery.

Algoritmy:

– hierarchivké zhlukovanie
– metóda k – priemerov
– neurčité / fuzzy zhlukovanie
– kohoneneove siete (samo - organizúce mapy)
– asociačné pravidlá
– analýza sekvencií
– faktorová analýza

Jedna z najrozšírenejších metód pri tvorbe dataminigových projektov je priemyselný CRISP-DM (CRoss-Industry Standard Process for Data Mining).



Obrázok č. 2.3.2.2.1: CRISP-DM

2.3.3 Nástroj dolovania

2.3.3.1 Weka 3

Dolovanie z dát za pomoci Open Source Machine Learning Software Weka 3. Tento nástroj je vyvinutý na univerzite Waikato na Novom Zélande. Napriek tomu, že je voľne dostupný, svojou kvalitou prekonáva aj niektoré komerčné systémy.

2.4 Metódy vývoja GUI

User Interface (UI) slúži na komunikáciu človeka so strojom. Nastáva obojsmerná výmena informácií za pomoci vstupných a výstupných periférnych zariadení, ďalej len PZ. Na strane stroja sa generujú výstupy, na základe ktorých reagujú zmysly človeka. Tok informácií opačným smerom (človek >>> PZ) nastáva za pomoci výstupných zariadení, ktoré reagujú na akcie človeka. Základné vnemy na prenos smerom k človeku (PZ >>> človek) sú [6]:

- **obraz** zmyslový orgán zrak, jedno z najrozšírenejších UI
- **zvuk** sluch, prenos menšieho objemu informácií vo zvukovej forme
- **hmat** menej používaný, trend v IT určuje jeho rozvoj do budúcnosti, avšak dnes sa využíva hlavne pre nevidiacich užívateľov
- **čuch** takmer nepoužiteľné pre stav chémie a techniky neumožňujúcej umelo rozpoznávať chute a vône

Pre smer k PZ (človek >>> PZ) sú:

- **pohyb** medzi najbežnejšie prostriedky predávania informácií týmto smerom patrí nepochybne klávesnica, ktorá býva doplnená zvyčajne myšou pre pohodlnejšie ovládanie polohy kurzora, ako aj operácií stláčaním tlačidiel
- **zvuk** ovládanie strojov ľudskou rečou je predmetom skúmania a je daná vízia, že tento spôsob komunikácie môže nahradiť pohybové PZ akým je klávesnica. Stále je však problém s porozumení reči na strane strojov.
- **obraz** rozpoznávanie snímaných pohybov strojmi

V nasledujúcich častiach budeme hovoriť hlavne o grafickom užívateľskom rozhraní - Graphical user interface (GUI) a jeho správnom návrhu a realizácii. Voľba správneho grafického

užívateľského rozhrania je jedným z najdôležitejších faktorov, ktoré nám zaručia nielen úspech systému, ale aj spokojnosť ich budúcich používateľov. Napríklad aj voľba farieb môže negatívne vplývať na užívateľa a tým tak znížiť produktivitu jeho práce. Tento jav môže nastať napríklad voľbou sivých a fádnych odtieňov, ktoré pôsobia depresívne a melancholicky. Ďalej voľba správnych či nesprávnych komponentov nám môže zaručiť úspech daného rozhrania, alebo vyvolať nežiadúce stavy zúrivosti na strane užívateľa. Je dôležité, aby každý programátorský tím mal aj grafického dizajnéra, ktorý bude navrhovať vzhľad a ucelenosť vyvíjaných softvérov. Bude musieť byť svojím spôsobom aj psychológ, aby spoznal cieľovú skupinu užívateľov, pre ktorých bude systém určený. Je rozdiel tvoriť GUI pre sekretárku, skúseného užívateľa alebo vývojára. Dôležité sú požiadavky, ktoré sa budú od systému očakávať.

2.4.1 Ovládacie prvky

Ovládacie prvky musia byť jednoznačné. V najlepšom prípade by malo byť užívateľovi hneď zrejmé, akú funkcionalitu daný prvok znázorňuje. Pri tlačítkach môžeme použiť ikony, ktoré by mali byť výstižné a v rámci aplikácie jednotné. Je vhodné použiť aj *„tool tips"* pre upresnenie funkcie tlačítka.

2.4.2 Menu

Väčšina programov obsahuje menu. Správne zásady, ako má menu vyzerať sú nasledujúce. Umiestňuje sa zvyčajne do hornej časti hlavného okna. Najčastejšie používané položky patria na jeho ľavú stranu a smerom doprava by sa mali dostať jeho menej používané časti. Ale nemusí to byť vždy pravidlo. Klávesové skratky nie sú tiež na škodu a vo forme akcelerátorov urýchlia spúšťanie často používaných častí. Pri menu by sme sa mali dodržiavať maximálny počet zanorení, ktorý neprekročí hodnotu 2. [7] Posledným prvkom menu býva zvyčajne nápoveda.

2.4.3 Panel nástrojov

Môžme ho nazvať aj panel rýchlej voľby a umiestňujeme doň najčastejšie používané časti programu. Panel nástrojov sa umiestňuje najčastejšie pod menu do horizontálnej polohy. Ale nič nepokazíme ak zaexperimentujeme a umiestnime ho vertikálne.

2.4.4 Okná – formuláre

Na zobrazenie vstupu alebo výstupu sa zvyčajne používajú okná. Či už sú to plávajúce okná alebo záložky, ich funkcionalita je rovnaká. Pri plávajúcich oknách treba zabezpečiť zámok a nadradenosť, ktorý na jednej strane obmedzuje užívateľa ale na druhej strane nestratí prehľad nad

vykonávanou akciou. Maximálny počet súčasne používaných okien by nemal prekročiť hranicu 3, pretože v nich užívateľ pravdepodobne stratí orientáciu. [8] Pri záložkách je správa okien jednoduchšia, ale nie v každom prípade sa dá táto komponenta použiť ako základný stavebný prvok aplikácie. Pri zložitých formulároch je dobré jednotlivé časti rozlíšiť aj vizuálne. Musia pôsobiť usporiadane, súvisiace časti najlepšie ohraničiť a priradiť im názov, aby boli prehľadnejšie.

2.4.5 Dialógové okná

Ďalšou neoddeliteľnou súčasťou správneho GUI sú dialógové okná, ktoré majú za úlohu informovať užívateľa. Väčšinou sa jedná o informovanie nás o vykonanom deje. Taktiež ich môžeme použiť dialóg otázka typu Áno/ Nie a zmenšiť tak riziko náhodného spustenia dôležitej udalosti.

2.4.6 Ukazovateľ priebehu

Pri časovo náročných operáciách je zapotreby dať užívateľovi informáciu o stave vykonávanej akcii jednak v grafickej podobe a pre lepšiu zrozumiteľnosť pridať aj textovú podobu vo forme percent alebo pomeru. Bez tejto funkcionality by užívateľ strácal kontrolu nad programom, ak by operácia trvala dlhšie ako pár sekúnd.

2.4.7 Nápoveda

Je jednou z najdôležitejších častí GUI. Veľmi ťažko definovať model, ako by mala nápoveda vyzerať. Mala by byť pre užívateľa ľahko zrozumiteľná a prehľadná. Tento stav ale ťažko dosiahneme, keďže užívateľ je ľudská bytosť s jedinečným vnímaním a pohľadom na vec, ale môžeme sa aspoň pri jej návrhu a tvorbe priblížiť.

2.4.8 Kvalita GUI

Je zložená s mnohých aspektov. Musí fungovať ako celok a zároveň poskytnúť komfort užívateľovi pri práci. Zároveň musí dbať na bezpečnosť a mala by byť pripravená na rôzne výpadky, ktoré by mali sprevádzať čo najmenšie škody. Testovanie prebieha ručne (klasické testovanie zo strany užívateľa), za pomoci testovacích skriptov a v neposlednej rade vznikli rôzne programy určené priamo k testovaniu GUI.

3 Analýza systému

Kapitola sa zaoberá požiadavkami kladených na systém a návrhom niekoľko možných spôsobov riešenia. Dôraz bude kladený na kvalitu grafického užívateľského rozhrania, ktoré zabezpečí užívateľovi príjemný a efektívny zber dát a na druhej strane vyberieme vhodné metódy a technológie, ktoré tento cieľ pomôžu zrealizovať. Každý dobrý systém musí prejsť procesom dôkladnej analýzy, aby boli následne zvolené vhodné prostriedky pri jeho realizácii.

3.1 Požiadavky na systém

Táto časť predchádzala častým konzultáciám s trénerom, ktorý dôkladne špecifikoval a upresňoval požiadavky na systém a uviedol ma tak do problematiky systému hodnotenia hráčov hádzanárskeho tímu.

3.1.1 Systém hodnotenia hráčov – špecifikácia systému

Pri sledovaní videozáznamu zápasu sa zapisuje počet kladných a záporných činností každého hráča, ako aj jeho odohraný čas. Tie sú potom zdrojom informácií pre následné hodnotenie - výpočtu herného výkonu (HV) jednotlivých hráčov. Činnosti sú rozdelené podľa kategórií na streľbu, technické chyby a jednotlivé plusové a mínusové body. Streľba sa zaznamenáva zvlášť u hráčov, kde sa berie do úvahy úspešná / neúspešná, pri poste brankárky to je chytená / nechytená. Úspešná a chytená sa zaraďuje medzi plusové činnosti, neúspešná a nechytená medzi mínusové. Každá jedna činnosť má svoju bodovú váhu.

Tabuľka 3.1.1.1: Streľba

Činnosť	úspešná	neúspešná	chytená	nechytená
spojka	3,5	2,0	2,0	3,5
krídlo	2,5	2,5	2,5	2,5
pivot	2,5	3,0	3,0	2,0
trhák	2,0	3,5	4,0	1,0
preskok	2,5	3,0	3,5	1,5
7m hod	2,0	3,5	4,0	1,0
iný priestor	4,0	1,0	1,0	4,0

Technické chyby ako je prešľap, prerážanie a kroky sa považujú za mínusové činnosti.

Činnosť	
prešľap	2,5
prerážanie	2,5
kroky	2,5

Tabuľka 3.1.1.2: Technické chyby

Ďalej sú to osobné body každého hráča/hráčky, ktoré sú zobrazené v tabuľke 3.1.1.3.

asistencia	2,5	-	-	-
zisk lopty	2,5	strata lopty, zlá prihrávka		2,5
zisk 7m hodu	3,0	zavinenie 7m hodu		3,5
+ v obrane	2,5	- v obrane		2,5
získané vylúčenie	2,0	vylúčenie		2,0
získané vykázanie	5,0	vykázanie		5,0
získaná diskvalifikácia	5,0	diskvalifikácia		5,0

Tabuľka 3.1.1.3: Plusové a mínusové body

Číselné hodnoty v tabuľkách predstavujú bodové hodnoty jednotlivých úkonov (Váha činnosti). Herný výkon následne vypočítame podľa vzorca:

$$HV = \frac{(S + \sum C + \sum (PC * H) - \sum (MC * H) + T)}{OH}$$

HV	herný výkon hráča
S	percentuálna úspešnosť streľby
C	počet jednotlivých činností
PC	počet jednotlivých kladných činností
MC	počet jednotlivých záporných činností
H	bodová hodnota jednotlivej činnosti
T	odohraný čas
OH	osobné hodnotenie trénera pre hráča

Zo strany trénera bol daný požiadavok na OH, ktoré vyjadruje známku rovnakým spôsobom ako na školách, v stupnici 1 až 5., aby systém dokázal určiť osobné hodnotenie, ktoré by sa stalo nezávislým na jeho subjektívnom názore. Analýzou a testovaním na vzorku dát som dospel k nasledovnému riešeniu. Vytvoril som si váhu V1 = $\sum$ (PC*H), ktorá predstavuje sumu všetkých plusových činností vynásobených o ich príslušné bodové hodnoty a obdobne váhu V2 = $\sum$ (MC*H), ktorá predstavuje sumu všetkých mínusových bodov vynásobených o príslušné bodové hodnoty. Ďalší postup pre jednoduchšie porozumenie znázorním jednoduchým algoritmom.

ak je (V1 – V2) >= 0 potom

priemernaVahaNaJedenPlusovyBod = (V1 – V2) * V1/PC * 100

ak priemernaVahaNaJedenPlusovyBod >= 90 potom OH = 1

ak priemernaVahaNaJedenPlusovyBod >= 75 potom OH = 1.5

ak priemernaVahaNaJedenPlusovyBod > =55 potom OH = 2

ak priemernaVahaNaJedenPlusovyBod >= 30 potom OH = 2.5

inak OH = 3

inak (prevyšujú váha V2)

priemernaVahaNaJedenMinusovyBod = (V2 – V1) * V2/MC * 100

ak priemernaVahaNaJedenMinusovyBod >= 75 potom OH = 5

ak priemernaVahaNaJedenMinusovyBod >= 50 potom OH = 4.5

ak priemernaVahaNaJedenMinusovyBod > =25 potom OH = 4

inak OH = 3.5

Tým sa vyhovelo požiadavku trénera, čo sa určite odzrkadlí na skrátení výsledného času potrebného pre zadávanie dát.

3.2 Grafické užívateľské rozhranie

Budúca aplikácia bude vytvorená vo vývojovom prostredí NetBenas IDE, ktoré zabezpečí dostatočnú základňu pre pohodlnú prácu. Jeho súčasťou je nástroj SWING, poskytujúci veľkú podporu pri tvorbe GUI novej aplikácie.

Pre budúci systém bude ako základný stavebný prvok použítá swingová komponenta JTabbedPane, ktorá nám zabezpečí poriadok a prehľad pri vytváraní a správe okien a panelov. Za cieľ som si stanovil vytvoriť rozhranie na princípe *„typu firefox“*, kde sa spustený panel zobrazí do nového tabu v hlavnom formulári. Bude treba navrhnúť jednotlivé súčasti tak, aby užívateľ mohol jednoducho spravovať všetky potrebné úkony a aby ako celok pôsobili jednotným usporiadaným dojmom pre užívateľa. Taktiež vhodná kombinácia klávesových skratiek môže efektívne napomôcť k celkovej obsluhe programu. Za pomoci myši a klávesnice sa budem tak snažiť vytvoriť vyhovujúce prostredie pre správu hodnotenia hráčov. Bude zapotreby mať k dispozícii ich paletu, s jednoduchým a pohodlným výberom a následné zadávanie jednotlivých činností do systému, ako aj ich zobrazovanie. K tomuto účelu budem často využívať základnú komponentu pre zobrazenie údajov akou je tabuľka, v prípade swingu - JTable. Na oživenie systému mám v úmysle vytvoriť sadu ikon, ktoré nám nahradia labely tlačidiel a pri správnej voľbe sú niekoľkonásobne efektívnejšie. Jednou z nich určite budú ikony plus a mínus, ktoré nám nahradia pridávanie a rušenie jednotlivých objektov.

Pri ich správnom umiestnení to ocení hlavne používateľ, pretože budú výraznejšie ako text a tým budú jednoduchšie rozpoznateľné pre identifikáciu potrebného úkonu, ktorý tlačítko reprezentuje.

4 Návrh implementácie

V tejto časti si podrobne rozoberieme modelovanie požiadavkov na objektový návrh aplikácie, ktorý dôkladne zanalyzujeme a navrhneme tak model nastávajúcej aplikácie. Ten si rozdelíme do príslušných balíkov podľa spoločných vlastností tried.

4.1 Vývojové prostredie

Aplikácia bude vyvíjaná v prostredí NetBeans IDE 6.5 za pomoci objektovo orientovaného programovacieho jazyka java. Presnejšie Java SE. Výhodou tohoto prostredia je jeho zabudovaný grafický nástroj SWING pre tvorbu užívateľských rozhraní, ako aj možnosť využitia rôznych dostupných pluginov pre uľahčenie práce, ktoré toto prostredie ponúka. NetBeans je open source projekt vytváraný pod licenciou CDDL – Common Development and Distribution License. Od roku 2000 patrí pod spoločnosť Sun Microsystems (en). Osobne hodnotím toto IDE ako jedno z najlepších v súčastnosti.

4.2 Úložisko dát

Úložisko dát bude reprezentovať objektovo orientovanú databáza db4objects, ktorá disponuje mnohými kladnými vlastnosťami a pomerne dobre popísaným štandardom pre tvorbu budúcich dotazov, ktorých náročnosť sa zvyšuje od zložitosti štruktúr použitého objektového modelu. K projektu stačí pripojiť knižnicu db4o-6.4.48.10991-java5.jar, ktorá poskytuje všetky potrebné rozhrania pre prístup k našim budúcim dátam.

4.3 Návrh Tried

Vychádza zo špecifikácii užívateľa na daný systém. Postupnými krokmi a upresneniami sme sa prepracovávali k výslednému návrhu. Ten spočíva z možnosti správy hráčov, klubov a zápasov a hodnotení.

4.3.1 Person

Balíček zastrešujúci osobu a všetky jej potrebné súčasti. Hlavná trieda `Objekt` je predok všetkých tried, ktorých objekty sa budú ukladať do databázy. Trieda `Person` základných atribútov obsahuje aj kolekciu verzií osoby, ktoré bude reprezentovať trieda `PersonVersion`. Z triedy `Person` dedia všetky jej vlastnosti triedy `Trainer` a `Player`. `Player` obsahuje kolekciu verzií hráča `PlayerVersion`, ktorá zaznamenáva úplne iný typ informácií, ako trieda `PersonVersion`. Všetko si podrobne rozoberieme, keď sa dostaneme k samotným triedam. Ďalej obsahuje jednoduché triedy reprezentujúce kontaktné body ako sú `Telephone`, `Adress` a `Email`. Tie sú naviazané na osobu. Ďalej obsahujú triedy s príponou `Factory`, ktoré slúžia na ukladanie príslušných objektov daných tried do databázy. Triedy zakončené príponou `FactoryCreator` obsahuje v sebe konštruktor tried `FAC,` ktorá predstavuje `Factory` príslušnej triedy.

4.3.1.1 Objekt

Predok všetkých tried, obsahuje v sebe atribút objId, na základe ktorého bude možné určiť jeho jednoznačnú identifikáciu a bude sa generovať pri uložení do databázy.

4.3.1.2 Person

Predok tried Player a Trainer bude obsahovať pripojené kontaktné body ako adresu, telefón a email. Taktiež bude obsahovať informáciu o pohlaví, ktoré bude definované v číselníku CPlayer, či dátume narodenia, ktoré sú v rámci celého životného cyklu osoby nemenné, ale pre užívateľa je ponechaná možnosť zmeny k prípadnej potrebe opravy. Dôležitým atribútom je zoznam verzií osoby..

4.3.1.3 PersonVersion

Verzia osoby bližšie špecifikuje osobu v jednotlivom časovom období, na ktorú sa odkazuje pomocou jej id. Každá verzia sa vyznačuje svojou platnosťou od - do, ktorá identifikuje, aké atribúty má osoba pri danej platnosti. Ak budeme zobrazovať hráčov v zápase, tak príslušnú verziu nám určí práve dátum zápasu. Ide o atribúty meno, priezvisko a stav osoby (single – married).

4.3.1.4 Player

Player, odvodená od triedy Person dedením, preberá všetky jej vlastnosti. Má jediný atribút domáci klub, ktorý je jediný nemenný v rámci hráča, ale užívateľovi je opäť táto možnosť ponechaná. Ostatné údaje bude obsahovať v kolekcii s verziami hráča, ktoré ho budú bližšie špecifikovať v jednotlivých časových obdobiach.

4.3.1.5 PlayerFactory

Továreň na ukladanie objektu PlayerVersion do databázy.

4.3.1.6 PlayerFactoryCreator

Obsahuje v sebe konštruktor FAC, ktorý nám sprístupní metódy na továrni PlayerFactory.

4.3.1.7 PlayerVersion

Verzia hráča nesie informácie o stavoch hráča. Obsahuje odkazujúce id na hráča, či je reprezentant, post hráča na ihrisku. Posty sú definované v triede číselníku CPlayer (ľavé a pravé krídlo, ľavá, pravá a stredná spojka, pivot a brankár). Posledným jeho atribútom je pole kategórií, v ktorých môže hrať. Maximálne môže zastávať tri kategórie, ale zvyčajne to býva 1 – 2. Je to jeden z faktorov, na základe ktorého sa pridávajú hráči do zápasu. Ako aj podľa toho, či dátum zápasu je v rozsahu platnosť od – do verzie.

4.3.1.8 PlayerVersionFactory

Továreň na ukladanie objektu PlayerVersion do databázy.

4.3.1.9 PlayerVersionFactoryCreator

Obsahuje v sebe konštruktor FAC, ktorý nám sprístupní metódy na továrni PlayerVersionFactory.

4.3.1.10 Trainer

Trieda dedí všetky atribúty od osoby, sám však nemá žiadne, pretože to špecifikácia systému nevyžadovala, ale takto bude možné jeho vlastnosti ľahko rozšíriť.

4.3.1.11 TrainerFactory

Továreň na ukladanie objektu Trainer do databázy.

4.3.1.12 TrainerFactoryCreator

Obsahuje v sebe konštruktor FAC, ktorý nám sprístupní metódy na továrni TrainerFactory.

4.3.1.13 Adress

Trieda adresy v sebe ukrýva ulicu, číslo domu, PSČ a mesto.

4.3.1.14 Email

Email obsahuje atribút email 1 a email 2.

4.3.1.15 Telephone

Telephone obsahuje atribút telefón 1 a telefón 2.

4.3.2 Match

Hlavná trieda je označená obdobne menom Match. Reprezentuje zápas, ktorý v sebe ukrýva list hodnotení zápasu pre jednotlivých hráčov. Hodnotenie reprezentuje trieda Rating, ktorá sa odkazuje na zápas a hráča z balíčka person. Ďalej tu nájdeme triedu Club, ktorá v sebe zahŕňa zoznam verzií klubu, zastrešenou triedou ClubVersion.

4.3.2.1 Club

Reprezentuje klub tímu. Obsahuje príznak, či je klub domáci, mesto klubu a kolekciu verzií klubu, na základe ktorých je zobrazovaný príslušný názov klubu podľa časového horizontu v konkrétnom zápasu.

4.3.2.2 ClubFactory

Továreň na ukladanie objektu Club do databázy.

4.3.2.3 ClubFactoryCreator

Obsahuje v sebe konštruktor FAC, ktorý nám sprístupní metódy na továrni ClubFactory.

4.3.2.4 ClubVersion

Verzia klubu, ktorá nesie atribúty id odkazujúceho klubu, názov klubu a platnosť od – do. V rámci systému môže byť len jeden domáci klub. Ak nastavíme nový za domáci, na predchádzajúcom sa príznak automaticky zneplatní. Názov klubu bude zobrazený z verzie, ktorá je platná k danému dátumu.

4.3.2.5 ClubVersionFactory

Továreň na ukladanie objektu ClubVersion do databázy.

4.3.2.6 ClubVersionFactoryCreator

Obsahuje v sebe konštruktor FAC, ktorý nám sprístupní metódy na továrni ClubVersionFactory.

4.3.2.7 Match

Obsahuje základné informácie o zápase a to sezóna v podobe rok1 / rok2, dátum zápasu, kategória odkazovaná na číselník CPlayer. Bude možné vyberať z mladších a starších žiakov / žiačok, mladších a starších dorastencov / dorasteniek a nakoniec muži a ženy. Pretože som použil skratky, ktoré sú rovnaké pre mužské aj ženské pohlavie, bude obsahovať aj informáciu o type zápasu (muži, ženy). Každý zápas bude mať aj svoje číslo, dvoch rozhodcov, delegáta, domáci a hosťujúci klub, príznak, či sa hralo doma, výsledok zápasu a nakoniec najdôležitejší atribút - hodnotenie, ktoré bude v sebe ukrývať kolekciu hodnotení každého hráča zápasu. Do zápasu sa hráči budú vyberať podľa pohlavia, dátumu platnej verzie hráča a kategórie, v ktorej majú oprávnenie hrať.

4.3.2.8 MatchFactory

Továreň na ukladanie objektu Match do databázy.

4.3.2.9 MatchFactoryCreator

Obsahuje v sebe konštruktor FAC, ktorý nám sprístupní metódy na továrni MatchFactory.

4.3.2.10 Rating

Trieda určujúca herný výkon HV hráča v zápase. Obsahuje id odkazovaného zápasu a hráča, odohraný čas, úspešnosť streľby a mapu s počtami jednotlivých kladných a záporných činností, ktorých kľúče budú v číselníku CRating. K bodovým hodnotám jednotlivých činností sa dostaneme pomocou rozšírenej triedy CRatingForm, ktorá v sebe obsahuje atribút s vytvorenou triedou CRating, v ktorej sa pri vytvorení inicializuje mapa a tak sa k hodnote dostaneme podľa identifikátoru jednotlivej činnosti.

4.3.2.11 RatingFactory

Továreň na ukladanie objektu Rating do databázy.

4.3.2.12 RatingFactoryCreator

Obsahuje v sebe konštruktor FAC, ktorý nám sprístupní metódy na továrni RatingFactory.

4.3.3 Codelist

Trieda v sebe ukrýva jednotlivé číselníky, ktoré obsahujú všetky potrebné identifikátory a názvy pre jednotlivé triedy z ostatných balíčkov. Pri niektorých triedach je vytvorená takzvaná forma danej triedy, ktorá má vo svojom názve pripojené slovo Form. Je to z toho dôvodu, aby sa nemuselo vytvárať stále nové triedy, ktoré v sebe obsahujú naplnenie štruktúr konštantnými symbolmi. Takto cez triedu obsahujúcu v názve „Form“ sa k nim pohodlne dostane.

Triedy balíčka codelist (číselníky) budú obsahovať všetky dôležité identifikátory a ich hodnoty v podobe premenných a máp, ktoré nám uľahčia ich časté používanie. Pomocou identifikátorov ako kľúčov sa bude pristupovať k ich hodnotám.

4.3.4 GUI

Balíček v sebe zastrešuje hlavnú triedu celého systému JFrameMain. Ďalej obsahuje jednotlivé panely pre správu osôb, klubov zápasov a hodnotení, ktoré ako celok zabezpečujú komunikáciu užívateľa a systému pomocou tohoto grafického užívateľského rozhrania. Jednotlivé panely som navrhol podľa jedného vzoru, aby výsledná aplikácia vzbudzovala dobrý dojem. Efektívny zber dát bude reprezentovať panel pre vytvorenie a zmenu hodnotenia zápasu jednotlivých hráčov.

4.3.4.1 JFrameMain

Je prvou triedou z balíka gui, ktorá je zároveň aj hlavnou triedou aplikácie, zobrazujúca jednotlivé panely do záložiek. Bude obsahovať menu a panel s nástrojmi. Zabezpečí chod a riadenie celej aplikácie.

4.3.4.2 PersonCreatePanel

Spustiteľný panel v dvoch módoch – vytvoriť alebo zmeniť / pridať hráča alebo trénera. Bude prehľadne rozložený do podzáložiek podľa pridávaných parametrov. Zabezpečí prehľadné pridávanie osôb (hráčov / trénerov) a ich verzií.

4.3.4.3 PersonManagePanel

Bude mať za úlohu správu osôb – hráčov a trénerov. Bude možné vyhľadať a následne zvoliť možnosť spustiť `PersonCreatePanel` a pridať tak alebo zmeniť ich jednotlivé vlastnosti.

4.3.4.4 ClubCreatePanel

Umožní pridávať alebo meniť jednotlivé kluby a ich verzie, ktoré budú pod správou `ClubManagePanel`, ktorý zabezpečí správu, vyhľadanie a následné spustenie pridanť / zmenť klub a jeho verziu.

4.3.4.5 MatchCreatePanel

Bude základnou zložkou pre pridanie a zmenu zápasov. Zvolí sa sezóna, dátum, kategória, súťaž a typ (muži – ženy) a ostatné parametre ako číslo zápasu, rozhodcovia, delegát a výsledok zápasu. Domáci

klub bude automaticky doplnený, súpera bude možné vybrať z tabuľky z pomedzi platných klubov k danému dátumu zápasu. Ak sa zmení dátum, automaticky sa zmení aj ponuka klubov.

4.3.4.6 MatchManagePanel

Zabezpečí správu a vyhľadávanie zápasov a hodnotení, pridávanie a menenie ich vlastností podľa potreby užívateľa. Navyše bude rozpoznaný status zápasu vytvorenia hodnotenia pre jeho jednoduchšiu správu.

4.3.4.7 RatingCreatePanel

Je najdôležitejší panel aplikácie – vytvorenie a zmena hodnotenia zápasu. Snažil som sa vytvoriť čo najefektívnejšie zadávanie údajov pre vytvorenie jednotlivých hodnotení hráčov. Bližšie si ho popíšeme v implementácii, kde odôvodním výber správnych komponentov k dosiahnutiu stanoveného cieľa pre užívateľské rozhranie. K aktuálnemu dátumu zápasu bude načítaná paleta platných hráčov k dátumu zápasu, ktorých môžme hodnotiť na základe ich výberu. Hodnotiť môžme viacerých hráčov naraz a automaticky budú zobrazené ich informácie o všetkých činnostiach s aktuálnou hodnotou herného výkonu HV.

4.3.4.8 StatsMatchManagePanel

Panel slúžiaci na zobrazenie štatistiky, ktoré vyhovujú vyhľadávacej podmienke. Umožní tlačiť štatistiku do formátu pdf alebo spustiť detail zápasu za pomoci panela `DetailStatsMatchPanel`.

4.3.4.9 StatsPlayerManagePanel

Panel podľa zadanej podmienky vyhľadá hráčov a zobrazí ich hodnotenie v zápase. Na Detial Zápasu sa môžeme pozrieť cez panel `DetailStatsMatchPanel.`

4.3.4.10 DetailStatsMatchPanel

Slúži na zobrazenie detailu zápasu.

4.3.4.11 TableRenderer

Trieda sa stará o grafickú stránku tabuľky, nastavuje pozadie a farbu písma v jednotlivých bunkách.

4.3.5 Systém

Balíček systém obsahuje triedy spojené s prácou s databázou a tlačou do pdf súboru. Ďalej tu nájdeme triedy s príponou Factory a FactoryBuilder.

4.3.5.1 DBFactory

Továreň pre prácu s databázou. Obsahuje nastavenie databáze a metódy spojené s načítaním a ukladaním objektov.

4.3.5.2 DBFactoryCreator

Obsahuje v sebe konštruktor FAC, ktorý nám sprístupní metódy na továrni DBFactory.

4.3.5.3 Pdf

Trieda generujúca výstupný pdf súbor.

4.3.5.4 GenerateARFF

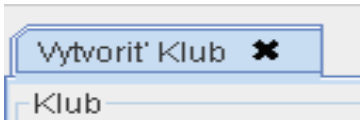
Transformuje dáta z databázy do formátu ARFF, aby bol použiteľný s aplikáciu Weka.

5 Implementácia

Kapitola bude venovaná predovšetkých implementácii užívateľského rozhrania, jeho vlastnostiam, zvoleným komponetom a ich vzájomnej komunikácii. Pre komplexné predstavenie hlavných balíčkov person a match si tiredy zobrazíme v class diagrame.

5.1 Realizácia

Hlavné okno je vytvorené pomocou `JFrame`, do ktorej je pridané menu `JMenuBar` pre vytvorenie komplexného a zároveň jednoduchého menu aplikácie, nasleduje `JToolBar`. Pridané klávesové skratky užívateľovi po ich zafixovaní značne urýchlia prácu pri používaní programu. Hlavným zobrazovacím prvkom je komponenta `JTabbedPane`, ktorá je rozšírená o zatváracie tlačítko – klasický krížik (obrázok 5.1.1).



Obrázok 5.1.1: Zatvoriť tab

Súčasťou `JFrameMain` sú metódy ako `initTabComponent(JPanel p)`, ktorá nám pridá tab s predaným panelom a rozšíri ho o spomínané tlačítko. Ďalšou metódou je `closeActiveTab ()`, ktorej názov nám napovedá, že bude zatvárať aktívne záložky z odkazujúcich sa panelov.

Pri všetkých paneloch sú použité tlačítka (`JButton`), z ktorých som niektoré rozšíril o ikony podľa ich funkčnosti. Ikony som tvoril tak, aby bolo ich použitie jednoznačné a pre užívateľa rýchlo identifikovateľné a aby dotvorili príjemný vzhľad aplikácie, ktorý je v dnešnej dobe dôležitý aj z estetického hľadiska. Pre zobrazovanie a zadávanie informácií som použil štandardné komponenty ako sú `JTextField`, `JFormattedTextField`, `JRadioButton`, `JChcekBox` a často využívanú tabuľku `JTable`. Každý panel má v sebe metódy na načítanie / uloženie príslušného modelu z / do panela, ako aj do databázy.

Teraz prejdeme na najdôležitejšiu časť a tou je panel hodnotenia zápasu `RatingCreatePanel`. Dostaneme s a k nemu cez zápas, správa a po vyhľadaní príslušného zápasu kliknutím na ikonu „+“. Panel je tvorený hlavičkou, ktorá identifikuje zápas. Ten je v tejto časti nemenný. Pri spustení panelu sa nám podľa dátumu zápasu načíta zoznam hráčov z databázy do tabuľky `jTablePlayers`. Ďalej sa načítajú do ďalších troch tabuliek príslušné akcie podľa postu hráčov. Pomocou tlačítka s ikonou plus alebo mínus (podľa prepnutia) nastavíme idnikátor, či chceme pracovať s kladnými alebo zápornými činnosťami hráča. Po oboch stranách tohoto tlačítka (hore / dole) nastavujeme, či budeme body pridávať alebo uberať. Potom po označení príslušných hráčov môžme klikať na akcie, ktoré sú v tabuľkách strely, technické a osobné body, ktoré slúžia ako tlačítka a tak pridávajú jednotlivým hráčom body podľa výberu príslušnej činnosti a taktiež sú pridávané do tabuľky hodnotenia, kde sa zobrazuje aktuálna percentuálna úspešnosť streľby ako aj HV každého označeného hráča. Aby som zvolil čo najväčšiu efektívnosť zadávania, pri každom kliknutí pridanie činnosti sa focus nastaví späť na tabuľku s paletou hráčov, a tak môžme efektívne kombinovať klávesnicu s myšou, čím dosiahneme časové zlepšenie zadávania dát. Odohraný čas sa nastavuje podobne až na to, že musí byť zvolený práve jeden hráč. Pri podržaní klávesy alt sa čas zväčší naraz o 5 minút, čim zamedzíme bezduchému klikaniu. A už stačí len uložiť a hodnotenie sa úspešne zapíše do databázy.

Tento panel prešiel tromi krokmi vývoja. Pridaní klávesových skratiek možnosti multiselectu pri výbere hráčov, ktorým možno naraz pridávať a uberať body stúpla efektivita zadávania dát a práca so systémom sa zjednodušila. Výsledné GUI ešte doplnilo k tejto funkčnosti krajší a ucelenejší grafický vzhľad pri zadávaní jednotlivých úkonov do systému, aby boli lepšie rozpoznateľné. Tým sa stala aplikácia v tomto smere veľmi prehľadnou a efektívnou.

Ďalšie dôležité panely sú `StatsMatchManagePanel`, ktorý spravuje štatistiku zápasov. Z tohto panela je možné za pomoci triedy `Pdf` vytvárať pdf súbory za pomoci knižnice iText pre tvorbu pdf dokumentov. Trieda `GenerateARFF` generuje ARFF súbor a následne sa spúšťa

aplikácia Weka. Tie sa nachádzajú v balíčku system spolu s triedami `DBFactory` a `DBFactoryCreator`. Tie majú implementované metódy spojené s ukladaním do DB. `StatsPlayerManagePanel` rozširuje funkcionalitu o možnosť správy hodnotenia hráčov. A nakoniec panel `DetailStatsMatchPanel` zobrazuje náhľad na hodnotenie zápasu.

6 Testovanie v reálnych podmienkach

Testovanie prebehlo na dvoch zápasoch. Priemerná doba zadávania dát bez použitej aplikácie býva približne 2 hodiny. Najskôr testoval verziu programu 1.0 nový používateľ, ktorému bol program predstavený a nemal s ním skoro žiadne skúsenosti. Dospeli sme k výsledku 1 hodina 40 minút. Približne rovnaký výsledok som očakával a s výsledkom testu som bol spokojný. Následne som pristúpil k zadávaniu údajov pristúpil pokročilý používateľ a výsledok sa dostavil už s badateľným rozdielom. Čas v tomto prípade bol 1h 25 minút. Pri verzii 1.1 boli časy kratšie, neskúsený užívateľ sa dostal pod hranicu 1hodina 33 minút, skúsený na čas 1 hodina 20 minút. Verzia 1.3 bola však najúspešnejšia. Neskúsenému užívateľovi to trvalo 1h 27 minút, skúsenému 1h 13 minút. Tým sa kladné očakávania naplnili a na druhej strane aplikácia splnila svoj účel.

7 Záver

V závere by som chcel zhodnotiť výsledky tejto práce, ako aj nastoliť možnosť ďalšieho vývoja aplikácie. Navrhnem možnosti neskoršieho rozšírenia, ako aj jej využitia v praxi.

Práca s aplikáciou je jednoduchá, rýchla a účelná. Zvolením objektovej databázy db4objects som sa dostal do konfliktu s data minigom, pretože nie je pre túto technológiu priamo podporovaný. Na toto zistenie som prišiel až príliš neskoro. Pomocou nástroja Weka a exportom pozorovaných dát do súboru formátu ARFF som sa snažil tento problém z časti eliminovať. Z toho dôvodu som sa sústredil na kvalitu a efektivitu zadávania dát, ako aj celkový vzhľad aplikácie. Navrhol som ho pre presnú potrebu užívateľa, ktorej zodpovedá finálna verzia 1.3.

K štatistikám zápasov a hráčov by som pridal grafy, ktoré by v niektorých prípadoch zrozumiteľnejšie zobrazovali výsledky na výstupe. Ďalej by to bolo doplnenie dialógových okien, ktoré by komplexnejšie informovali a ošetrovali niektoré udalosti programu a indikovali tak neoprávnené kroky užívateľa. Aplikáciu by bolo možné prerobiť na univerzálny hodnotiaci systém herného výkonu všetkých kolektívnych hier, čím by sme získali univerzálny nástroj, ktorý by bol veľkým pomocníkom trénerov pri ich práci.

Literatúra

[1] WWW stránky. Dvořák M. Návrhové vzory (design patterns)
http://objekty.vse.cz/Objekty/Vzory

[2] WWW stránky. Data model
http://en.wikipedia.org/wiki/Data_model

[3] WWW stránky. Db4objects
http://www.db4o.com/about/productinformation/resources/db4o-7.4-tutorial-java.pdf

[4] WWW stránky. HĺbkovÁ analýza dát.
http://sk.wikipedia.org/wiki/Hĺbková_analýza_dát

[5] WWW stránky. Data Mining
http://data-mining.sk/

[6] WWW stránky. Zemčík, P. Tvorba užívatelských rozhraní, Studijní opora
https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/ITU-IT/texts/lTU-Podpora.pdf

[7] WWW stránky. Bentley, John E.: 14 Steps to a Good GUI
http://www8.sas.com/scholars/05/PREVIOUS/1999/pdf/083.pdf

[8] WWW stránky. Hobart, James: Principles of good GUI design
http://axp16.iie.org.mx/Monitor/v01n03/ar_ihc2.htm

Zoznam príloh

Príloha 1. CD

Príloha 2. Manuál

Príloha 3. Pdf výstup jedného zápasu

7.1 Manuál

7.1.1 Správa osôb

Klávesová skratka `Ctrl+O`. V správe osôb máme možnosť vyhľadať osoby a následne po vybraní osoby v tabuľke kliknutím na zelenú šípku sa nám spustí editácia osoby, kde môžeme vytvoriť novú verziu alebo zmeniť pôvodnú.

7.1.2 Vytvorenie osoby

Máme dve možnosti. Buď vytvoríme hráča (`Ctrl+Alt+P`) alebo trénera (`Ctrl+Alt+T`). Pri vytvorení hráča máme panel s troma záložkami. Základné údaje, hráč a kontaktné údaje.

7.1.2.1 Základné údaje

Na tejto záložke vyplňujeme meno, priezvisko, dátum narodenia, pohlavie, stav a dátumy od-do, ku ktorým je verzia platná.

7.1.2.2 Hráč

Pri záložke hráč nastavujeme jeho post, kategóriu, ktorú zvolíme tlačidlom „ + “ do zoznamu. Hráč môže hrať najviac v troch kategóriách. Ďalej vyplňujeme materský klub, či je reprezentant, jeho výšku, váhu a dátumy od-do, ku ktorým bude daná verzia hráča platná. (Tréner túto záložku neobsahuje).

7.1.2.3 Kontaktné údaje

Vyplňujeme ulicu, číslo domu, PSČ, mesto, telefón 1, telefón 2, email 1, email 2.

7.1.3 Správa klubov

Klávesová skratka `Ctrl+K`. Umožňuje vyhľadať klub podľa mesta, vybraný klub z tabuľky môžeme spustiť tlačidlom zelenej šípky a umožniť tak vytvorenie alebo zmenu verzie klubu.

7.1.4 Vytvorenie klubu

Klávesová skratka `Ctrl+Alt+K`. Pri vytváraní klubu zadávame mesto, jeho názov, či je domáci a platnosť jeho verzie.

7.1.5 Správa zápasov

Klávesová skratka `Ctrl+Z`. Do vyhľadávacieho formuláru máme možnosť vyhľadávať podľa nasledujúcich parametrov. Číslo zápasu od-do, muži – ženy, doma – vonku, sezóny, kategórie, súťaž a mesto. Zobrazené zápasy môžeme meniť kliknutím na zelenú šípku tlačidlom „+“ nastavujeme hodnotenie zápasu. Máme možnosť hodnotiť spomedzi hráčov, ktorí majú platnú verziu k dátumu zápasu. Môžeme zvoliť vždy jedného alebo viacerých hráčov. Kliknutím na tlačidlo „modré plus“ alebo „zelené mínus“, ktoré vyjadrujú či budeme pracovať s kladne alebo záporne hodnotenými činnosťami. Zdvojená šípka nad/pod týmto tlačidlom nám bude určovať inkrementáciu/dekrementáciu činnosti. Podľa uvedeného nastavenia sa nám zobrazia činnosti: osobné body, technické body, streľba. Po kliknutí na jednotlivé činnosti sa zvoleným hráčom pridajú/odoberú činnosti, podľa vyššie spomenutých nastavení. Odohraný čas nastavujeme príslušnými tlačidlami v sekcii „odohraný čas“, pri podržaní ľavého Altu sa mení čas o 5 minút jedným kliknutím.

7.1.6 Vytvorenie zápasu

Klávesová skratka `Ctrl+Alt+Z.` Pri vytvorení zápasu nastavujeme: sezónu, dátum, muži – ženy, kategória, súťaž. Z tabuľky vyberieme súpera, zdvojenou šípkou máme možnosť vymeniť kluby domáci/hostia. Ďalej vyplňujeme číslo zápasu, rozhodcu 1, rozhodcu 2, delegáta a výsledok zápasu.

7.1.7 Štatistika zápasov

Klávesová skratka `Ctrl+S`. Máme možnosť vyhľadávania podľa čísla zápasu, dátum od-do, muži – ženy, doma – vonku, sezóna, kategória, súťaž, mesto. Výsledok tohto vyhľadávania je štatistika zápasov, ktorú môžeme exportovať do pdf kliknutím na tlačidlo tlačiarne, taktiež môžeme zobraziť detail hodnotenia zvoleného zápasu kliknutím na tlačidlo zelenej šípky. Zvolením tlačidla „W“ sa spustí dolovací nástroj Weka, cez ktorý je potrebné otvoriť vygenerovaný ARFF súbor.

7.1.8 Štatistika hráčov

Klávesová skratka `Ctrl+H`. Po vyplnení podmienky pre vyhľadávanie (meno, priezvisko, číslo zápasu, od-do, muži – ženy, doma – vonku, sezóna, kategória, súťaž, mesto) nám zobrazí do tabuľky výsledok – zoznam štatistík hráčov. Je možné spustiť pomocou tlačidla zelenej šípky detail hodnotenia hráčov.

7.1.9 Pomocník

Klávesová skratka `F1`. Zobrazí návod k programu.

Číslo	Dátum	Sezóna	Kategória	Súťaž	Rozhodca 1	Rozhodca 2	Delegát	Domáci	Hostia	Výsledok
7	1.11.2008	2008/2009	Ž	WHIL	Milan Čiernik	Ivan Odráška	Jana Vavrincová	HK Slovan Duslo Šaľa	Slávia Partizánske	25 : 17

Priezvisko	Post	V+	V-	VK+	VK-	D+	D-	A	Z+	Z-	7+	7-	O+	O-	K	PŠ	PR	GB	S+	S-	P+	P-	K+	K-	PK+	PK-	T+	T-	7+	7-	N+	N-	Min	%	HV
Hippová Zuzana	PK	2	0	0	0	0	0	9	6	2	2	1	13	3	0	0	1		2	2	0	0	2	0	0	0	2	0	1	1	0	0	50	70.0	116
Šťastná Petra	LK	2	0	0	0	0	0	5	3	3	0	1	11	0	0	0	1		1	1	0	1	2	0	0	0	0	0	0	0	0	0	60	60.0	88
Nagyová Jolana	LS	0	0	0	0	0	0	2	1	0	0	0	2	0	0	0	0		1	0	0	1	0	0	0	0	0	0	0	0	0	0	10	50.0	20
Lenčéšová Alena	PK	0	0	0	0	0	0	3	0	0	1	0	3	0	0	0	0		0	0	0	0	1	1	0	0	0	0	0	0	0	0	10	50.0	26
Haláková Simona	PS	2	0	0	0	0	0	6	4	0	0	0	7	0	0	0	0		1	3	1	0	0	0	0	0	1	0	0	0	0	0	60	50.0	90
Drotárová Jana	SS	0	1	0	0	0	0	6	2	3	1	0	16	0	0	0	0		0	1	0	0	0	0	0	0	1	0	1	1	0	0	50	50.0	90
Kováčová Marcela	B	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	21	11	1	0	2	3	0	1	1	0	1	2	0	0	60	60.5	78
Šuleková Lucia	LS	1	0	0	0	0	0	10	3	0	1	0	20	4	0	1	0		1	2	1	0	0	2	1	0	1	0	0	0	0	0	60	50.0	124
Stanková Marcela	P	0	0	0	0	0	0	2	1	0	1	0	0	0	0	0	0		1	2	0	0	0	0	0	0	0	0	1	0	0	0	10	50.0	20
Horvátová Beata	B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.0	0
Hudáková Klaudia	SS	2	1	0	0	0	0	5	1	2	0	0	17	3	0	0	0		1	0	0	2	1	0	0	0	0	0	0	0	0	0	50	50.0	88
Šípková Andrea	SS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.0	0
Gregušová Adela	LK	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.0	0
Lovásová Katarína	B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.0	0
Gulášová Darina	P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.0	0
Molnárová Marcela	LK	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.0	0
Goliášová Eva	LS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.0	0
Dubčeková Jana	B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.0	0
Spolu		9	2	0	0	0	0	48	21	10	6	2	90	10	0	1	2																		74

Hráč	Strieľ	Gólov	Úspešnosť [%]
spojka	19	8	42.1
pivot	6	2	33.3
preskok	1	1	100.0
krídlo	9	6	66.7
trhák	5	5	100.0
7 m hod	5	3	60.0
iný priestor	0	0	0.0
Spolu	45	25	55.6

Brankár	Strieľ	Gólov	Úspešnosť [%]
spojka	32	11	65.6
pivot	1	0	100.0
preskok	1	1	0.0
krídlo	5	3	40.0
trhák	1	0	100.0
7 m hod	3	2	33.3
iný priestor	0	0	0.0
Spolu	43	17	60.5