UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Databázová aplikace pro evidenci mechatronických stavebnic

Michal Grof

Bakalářská práce
2011

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2010/2011**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Michal GROF**
Osobní číslo:	**I07607**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Databázová aplikace pro evidenci mechatronických stavebnic**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem bak. práce je navrhnout a implementovat databázovou aplikaci pro evidenci mechatronických stavebnic s využitím programovacího jazyka Java. Vytvořená aplikace bude primárně sloužit k evidenci výpujček jak celých stavebnic, tak jednotlivých dílů stavebnic. Aplikace by měla umožňovat sestavení dotazů vedoucích ke zjištění dalších stavů evidence stavebnic (např. možnost kompletace různých typů stavebnic z dostupných dílů apod.).
Teoretická část zahrnuje:
- popis a porovnání možností připojení k databázi v rámci programovacího jazyka Java,
- návrh databáze pro evidenci mechatronických stavebnic,
- návrh aplikace pracující s databází.

Praktická část:
- realizaci databáze,
- vytvoření databázové aplikace.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

***1. GILFILLAN I., Mastering MySQL 4. Sybex, 2003. ISBN 978-0782141627.**

***2. TAYLOR M. Java and SQL. Malcolm Taylor, 2007. ISBN 978-0955676307.**

Vedoucí bakalářské práce: **Ing. Michael Bažant, Ph.D.**
Katedra softwarových technologií

Datum zadání bakalářské práce: **17. prosince 2010**

Termín odevzdání bakalářské práce: **13. května 2011**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2011

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 12.3.2011 Michal Grof

Poděkování

Chtěl bych tímto poděkovat vedoucímu mé bakalářské práce, panu Ing. Michalu Bažantovi, Ph. D. za pomoc, ochotu a důležité rady při zpracování tématu.

Anotace

Tato databázová aplikace slouží k přehledné evidenci mechatronických stavebnic a jednotlivých dílců stavebnic. Uchovává informace o tom, z kolika a z jakých dílců se skládají stavebnice. Udržuje informace o počtu dostupných stavebnic, o výpůjčkách stavebnic a o uživatelích. Vše by mělo být jednoduše dostupné v přehledné grafickém prostředí.

Klíčová slova

Java, MySQL, databázová aplikace, mechatronická stavebnice

Title

Database aplication for evidence of mechatronics kits.

Annotation

This database application is used to clearly evidence of mechatronic components and construction kits. It stores information on how many and what parts are composed of individual kits. It keeps information about the number of kits available, the kits loans, and users. Everything should be easily available within a graphical environment.

Keywords

Java, MySQL, database aplication, mechatronics kit

Obsah

Seznam zkratek

UML.....Unified Modeling Language
SQL.....Structured Query Language
GPL.....General Public License
API.....Application Programming Interface
JVM.....Java Virtual Machine
JDK.....Java Development Kit
JRE.....Java Runtime Enviroment
MFF UK.....Matematicko-fyzikální fakulta Univerzity Karlovy
PHP.....PHP: Hypertext Preprocessor
GUI.....Graphical User Interface
JDBC.....Java Database Connectivity

Seznam tabulek

Seznam obrázků

1 Úvod

Při správě velkého množství mechatronických stavebnic, které se často skládají z poměrně velkého počtu součástek a příslušenství různých typů, vznikají mnohé problémy. Například se součástky pomíchají, některé stavebnice se rozpůjčují a už se neví komu nebo se do stavebnice přimíchají součástky z jiné stavebnice atd. Aplikace, která je výsledkem této bakalářské práce, by měla tyto a podobné problémy vyřešit. Veškeré údaje se vloží do databáze, se kterou spolupracuje uživatelsky přívětivá desktopová aplikace.

Databázová část je postavena na databázovém serveru MySQL, který je dostupný ke stažení zdarma. Což byl jeden z důvodů, proč jsem zvolil zrovna MySQL. Aplikační část je napsána v programovacím jazyce Java pomocí vývojového prostředí NetBeans. NetBeans i Java jsou open-source a mám s nimi za celou dobu studia dobrou zkušenost.

Aplikace jako celek je založena na principu client-server. Databáze běží na serveru a klientská aplikace s databází vzdáleně komunikuje a získává z ní data. Aplikace umožňuje jednoduše a přehledně spravovat databázi stavebnic. Lze si prohlížet jednotlivé stavebnice a jejich jednotlivé dílce. Je možné se zaregistrovat a vypůjčit si nějakou stavebnici. Vše bude zaznamenáno v databázi a přístupné všem ostatním uživatelům.

Druhá kapitola popisuje některé databázové systémy, které mohly být použity při tvorbě databázové části aplikace.

Třetí kapitola stručně nahlíží na jazyk Java. Dozvíme se něco z historie tohoto jazyka, zaměříme se na jeho syntaxi a některá úskalí, která souvisí s interpretací kódu v jazyku Java.

Čtvrtá kapitola se již zabývá realizací databázové části aplikace. Je zde popsáno členění a struktura celé databáze, jednotlivé tabulky v ní a vztahy definované mezi tabulkami.

Pátá kapitola popisuje aplikační část, tedy aplikaci psanou v jazyce Java. Seznámíme se se strukturou programu, s jeho jednotlivými třídami a v několika náhledech uvidíme i design aplikace.

Poslední, šestá kapitola, nám nastíní úskalí propojení relační databáze a programu v jazyce Java pomocí konektorů.

2 Databáze

Pro tvorbu databázové části aplikace byla možnost použít několik různých databázových systémů.

2.1.1 Oracle Database

Oracle Database vyvíjí společnost Oracle Corporation. Jedná se o multiplatformní databázi s velmi pokročilým zpracováním dat. Současná verze je Oracle Database 11g. Pro studijní účely lze databázi používat zdarma. Jinak se jedná o placený software užívaný v mnoha společnostech.[5]

2.1.2 MySQL

MySQL byla vyvinuta švédskou firmou MySql AB, později ji vlastnila firma Sun Microsystems, ale tu pohltila firma Oracle Corporation. MySQL má duální licenci. Je dostupná pod bezplatnou licencí GPL, ale i pod komerční placenou licencí. Je multiplatformní, rychlá, ale podporuje jen jednoduší možnosti práce s daty. Až později se do MySQL implementovaly složitější struktury jako pohledy nebo triggery. Pro svou relativní jednoduchost a volnou dostupnost patří mezi nejčastěji používané databáze.[2]

2.1.3 PostgreSQL

Počátky PostgreSQL sahají do roku 1982, kdy se rozjel projekt Ingras na univerzitě v Berkely v Kalifornii. Později se projekt dostal k open-source vývojářům a v roce 1997 byla vydána PostgreSQL ve verzi 6.0. Databáze ProstgreSQL je ope-source software. Není vlastněna žádnou firmou a je vyvíjena komunitou vývojářů. Primárně byla vyvinuta pro GNU/Linux, ale existují verze pro Windows. ProstgreSQL je stabilní, rychlá a podporuje pokročilé technologie. V současnosti je na vzestupu a a její oblíbenost stoupá.[2]

3 Jazyk Java

Jazyk Java vyvinula v roce 1995 společnost Sun Microsystems. Později jej uvolnila jako open-source, je tedy dostupný bezplatně. Java je univerzální programovací jazyk a může být použit téměř na libovolný druh aplikace. Velká výhoda tohoto jazyka je jeho vazba na internet. Jazyk Java lze použít jak na straně webového klienta, tak na straně webového serveru. Toto je možné díky tomu, že jazyk Java je nezávislý na konkrétní platformě. Lze jej spustit na libovolném operační systému. Jazyk Java je modulární, lze se jej snadno naučit a obsahuje velké množství programových knihoven a rozhraní. Pro jazyk Java existují vynikající vývojová prostředí, která lze dokonce získat zdarma. To všechno jsou důvody velké obliby tohoto jazyka.[3]

Pojem Java lze chápat ve dvou významech. Jako programovací jazyk a také jako prostředí nabízející nejrůznější možnosti a rozhraní.

3.1 Java jako programovací jazyk

Java je objektově orientovaný jazyk, jehož syntaxe je založena na syntaxi jazyka C++. Jazyk Java nabízí vše, co je známé z ostatních programovacích jazyků. Tedy proměnné, cykly, podmíněné příkazy, atd. Java je velmi orientovaná na objektový přístup. Data jsou vytvářena jako instance tříd. Funkčnost je zajištěna formou metod jednotlivých tříd. Podporována je dědičnost, tedy opakované použití stejného kódu a specializace tříd. Ve srovnání s jazykem C++ je Java poněkud jednodušší. Snáze se učí – programátor se nemusí starat o správu paměti a navíc byly odstraněny některé složitější jazykové konstrukce. Java například odstranila ukazatele, které známe z jazyka C++, neumožňuje vícenásobnou dědičnost, nepodporuje šablony nebo přetěžování standardních operátorů. Java obsahuje velmi mnoho již hotových knihoven tříd. Třídy pro tvorbu grafického rozhraní, třídy pro přístup k databázím nebo třídy pro šifrování a kompresi dat. U operačních systémů, např. Windows, existuje řada programových rozhraní pro všechny možné oblasti a přístroje. Podobně Java nabízí své funkce ve formě API (Application Programmers Interface, rozhraní pro aplikační programátory).[3]

3.2 Java jako platforma

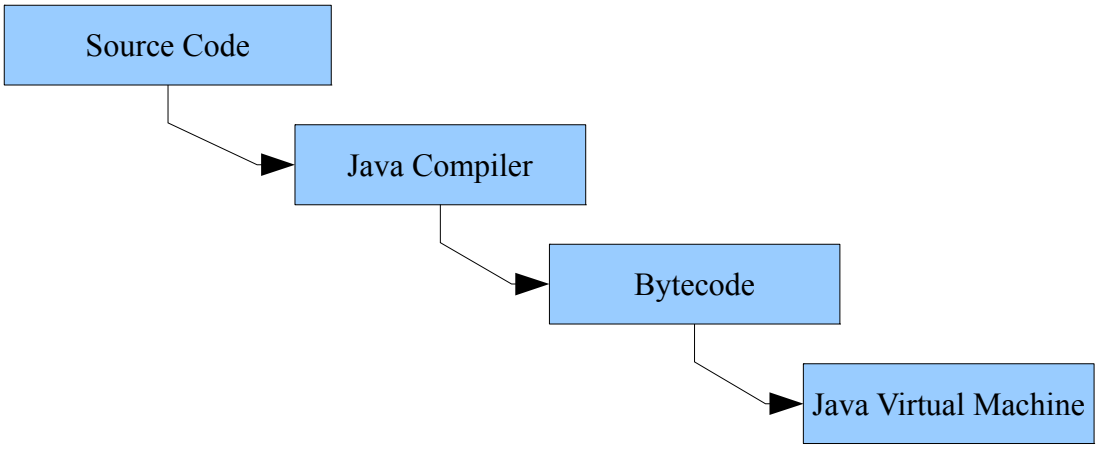
Na Javu lze také pohlížet jako na platformu. Co to znamená? Java ve skutečnosti poskytuje pro všechny operační systémy jedno a to samé rozhraní. Jednotlivá volání jsou za běhu převáděna do skutečné platformy. Toto abstraktně definované prostředí s příslušným emulátorem nazýváme virtuální stroj. Pro programátora se tak Java jeví jako samostatná platforma, nejen jako programovací jazyk.[3]

3.3 Nezávislost na platformě

Java je koncipována tak, aby se dala spustit na nejrůznějších systémech. Není proto vázána na konkrétní hardware nebo operační systém. Místo toho Java běží na abstraktně definovaném virtuálním stroji, který je na cílovém systému emulován.

3.3.1 Java Virtual Machine

Java je interpretovaný jazyk. Zdrojový kód je zpracován prostředím, ve kterém program běží, příkaz po příkazu je překládán do binárního kódu tzv. Bytecode. Tento Bytecode se potom spustí na virtuálním stroji Javy, který nazýváme Java Virtual Machine (JVM). Tento virtuální stroj musíme mít nainstalovaný v našem systému, abychom mohly spouštět programy napsané v Javě. Běhové prostředí s JVM nutné pro spouštění programů napsaných v Javě se nazývá Java Runtime Enviroment (JRE). Pro vývoj v Javě je nutný tzv. Java Developmnet Kit (JDK). Obsahuje jak běhové prostředí tak, vývojovou platformu. JDK i JRE jsou dostupné pro operační systémy Windows, Solaris a Linux.[3]



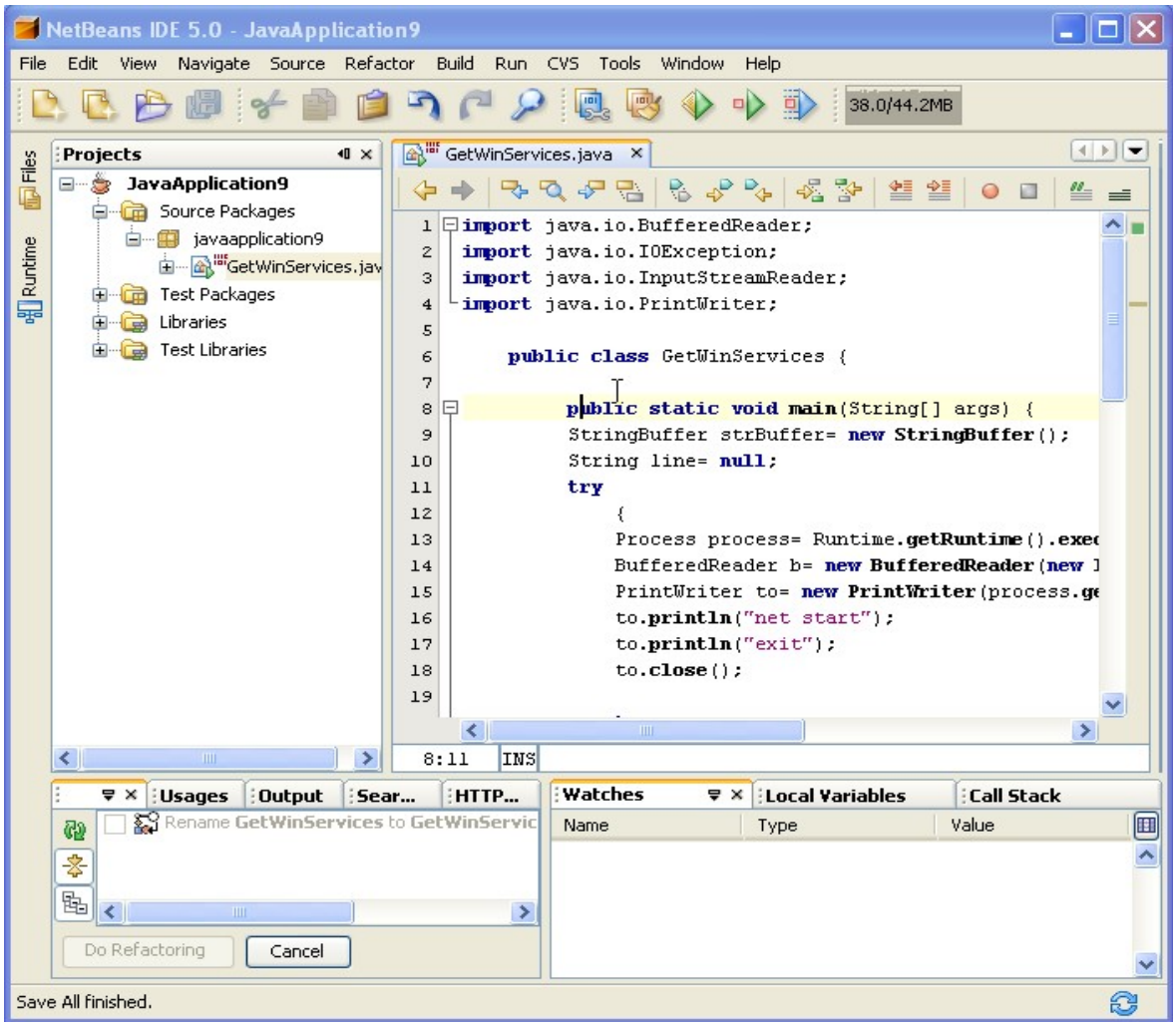
Obrázek 1: Překlad kódu jazyka Java [3]

3.4 Vývojové prostředí

Pro Javu existuje několik velmi dobrých vývojových prostředí. Většinou jsou dostupná zdarma. Můžeme jmenovat například Eclipse, JDeveloper nebo NetBeans.

3.4.1 NetBeans

Moje oblíbené prostředí je NetBeans. Pracuji s ním od mých prvních krůčků v jazyce Java. NetBeans je dostupné zdarma pro systémy Windows, Linux, Solaris i Mac OS. NetBeans původně vzniklo na MFF UK v Praze. Později bylo koupeno společností Sun Microsystems a uvolněno jako open-source. NetBeans nabízí programátorovi vše, co potřebuje pro pohodlné programování. Obsahuje i komponenty pro snadný návrh grafického uživatelského prostředí (GUI). Samotné NetBeans je napsáno v jazyku Java. Existuje pro něj mnoho zásuvných modulů, díky kterým můžeme rozšířit jeho funkčnost. Můžeme doinstalovat podporu jazyka C/C++, Ruby nebo PHP.[1]



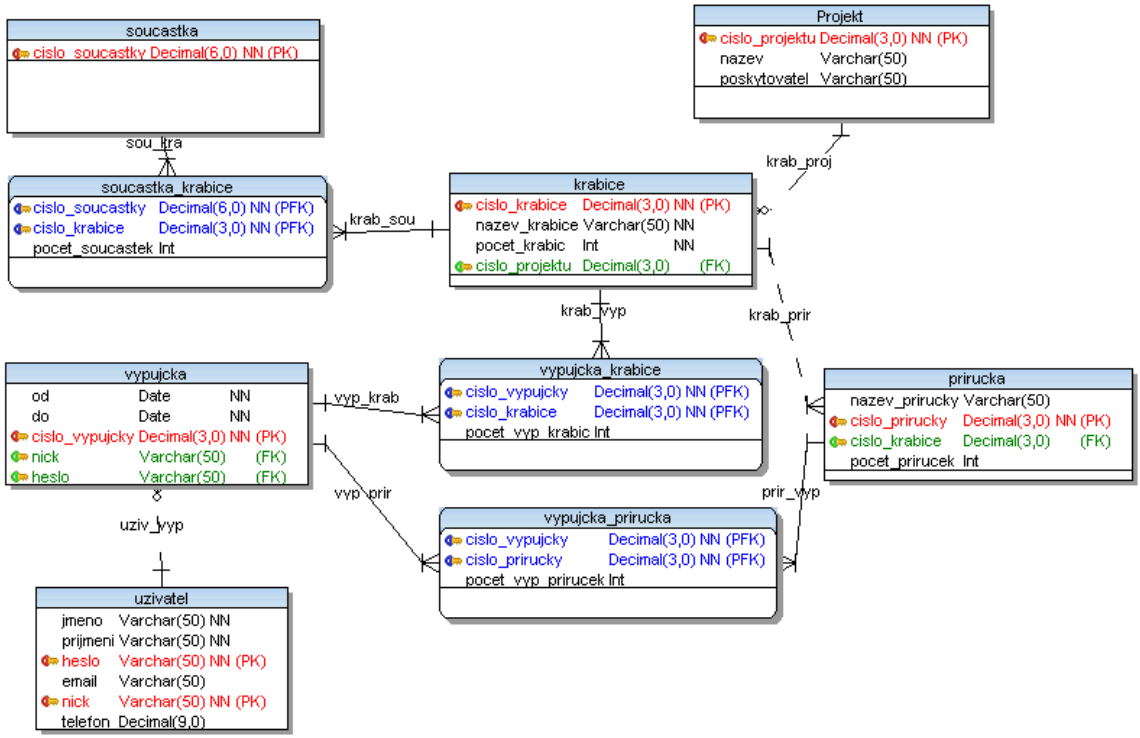
Obrázek 2: Vývojové prostředí NetBeans

4 Databáze Katalog

Databázová část aplikace je tvořena databází Katalog. V této databázi jsou uchovávána veškerá data potřebná pro funkci celé aplikace.

4.1 E-R Diagram

E-R diagram nám zobrazí strukturu databáze Katalog, její jednotlivé tabulky a vztahy mezi tabulkami. Detailní informace o vazbách mezi jednotlivými tabulkami a o samotných tabulkách jsou obsaženy v následujících podkapitolách.



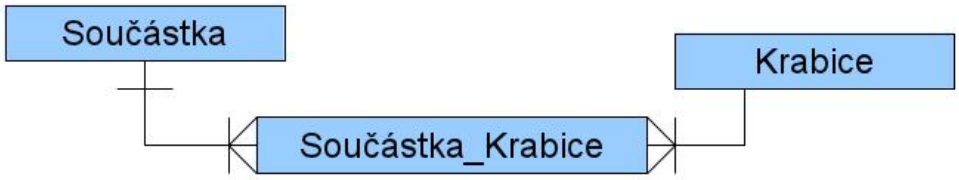
Obrázek 3: E-R Diagram databáze Katalog

4.2 Popis vztahů mezi tabulkami

Mezi jednotlivými tabulkami existují přesně definované vztahy. Tyto vztahy lze vyčíst přímo z E-R diagramu výše. Zde je popíšeme trochu podrobněji.

4.2.1 Vztah Součástka – Krabice

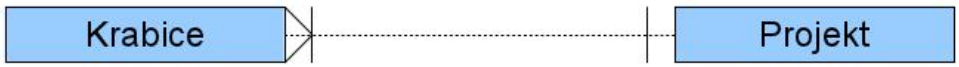
Je to velmi důležitý vztah definující sestavení jednotlivých krabic z jednotlivých součástek. Součástka se může vyskytovat v jedné nebo několika krabicích. Naopak krabice může obsahovat mnoho součástek a musí obsahovat minimálně jednu součástku. Krabice bez součástek neexistuje. Vztah je definován přes spojovací tabulku `Soucastka_krabice`. V této tabulce se uchovávají informace o tom, která součástka je součástí jaké krabice a kolik těchto součástek je v této krabici. Existují i samostatné součástky, například senzory nebo napájecí zdroje, které je nutné také evidovat. Aby se neporušil tento vztah, existuje fiktivní krabice „Samostatne soucastky“, ke které jsou přiřazeny všechny samostatné součástky.



Obrázek 4: Popis vztahu Soucastka-Krabice

4.2.2 Vztah Krabice – Projekt

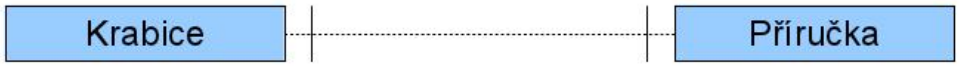
Krabice jsou pořizovány a financovány v určitých projektech. Aby bylo možné přiřadit stavebnici k určitému projektu je nutné definovat tento vztah. V rámci projektu existuje alespoň jedna nebo více stavebnic. Zatímco krabice patří právě do jednoho projektu.



Obrázek 5: Popis vztahu Krabice-Projekt

4.2.3 Vztah Krabice – Příručka

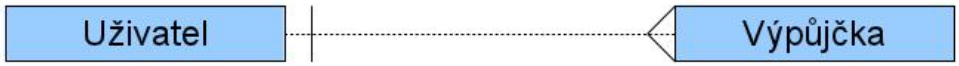
Ke každé krabici patří i příručka. Příručka obsahuje informace o krabici, návody na různé sestavení krabice a seznam součástek. Pro každou krabici tedy existuje (může existovat) příručka. Podle toho definujeme tento vztah.



Obrázek 6: Popis vztahu Krabice-Prirucka

4.2.4 Vztah Uživatel – Výpůjčka

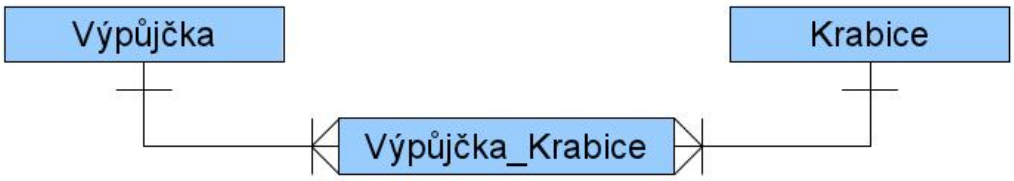
Je nutné, aby si aplikace uchovávala informace o uživatelích, kteří se zaregistrují. Tyto informace jsou v tabulce `Uzivatel`. Každý uživatel si může vypůjčit krabici nebo příručku ke krabici. Jednotlivé výpůjčky spravuje tabulka `Vypujcka`. Existuje tedy vztah, kdy každému uživateli náleží jedna nebo více výpůjček. Samozřejmě může být zaregistrován uživatel bez jakékoliv výpůjčky. Opačně ale každá výpůjčka náleží pouze jedinému zaregistrovanému uživateli.



Obrázek 7: Popis vztahu Uzivatel-Vypujcka

4.2.5 Vztah Výpůjčka – Krabice

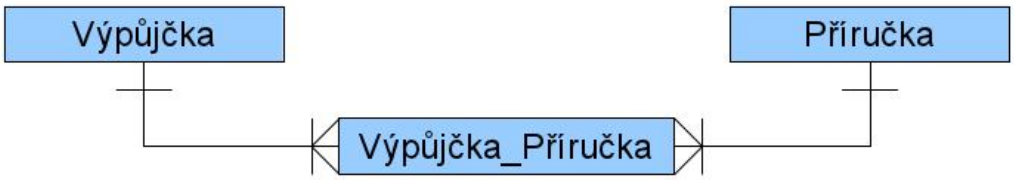
K výpůjčce náleží vypůjčená krabice. Protože výpůjčka může obsahovat více různých krabice a krabice se může vyskytovat ve více výpůjčkách, existuje spojovací tabulka `Vypujcka_krabice`, která definuje tento vztah. V této tabulce se tedy uchovávají čísla výpůjček a k nim související číslo vypůjčené krabice a počet těchto vypůjčených krabic.



Obrázek 8: Popis vztahu Vypujcka-Krabice

4.2.6 Vztah Výpůjčka – Příručka

Uživatel si může vypůjčit i příručku ke krabici. Proto je nutné definovat i vztah mezi výpůjčkou a příručkou. Podobně jako u vztahu Výpůjčka_krabice si lze vypůjčit více příruček a zároveň příručka se může vyskytovat ve více výpůjčkách. Proto je nutné vytvořit další spojovací tabulku `Vypujcka_prirucka`. Tato tabulka uchovává čísla jednotlivých výpůjček, k nim čísla vypůjčených příruček a počet těchto půjčených příruček.



Obrázek 9: Popis vztahu Vypujcka-Prirucka

4.3 Popis tabulek

Následuje podrobnější popis tabulek. V tomto popisu u názvů tabulek i jejich sloupců používám pro přehlednost českou diakritiku, i když v databázi jsou tyto názvy bez diakritiky.

4.3.1 Tabulka Součástka

Tabulka `Soucastka` slouží k evidenci jednotlivých součástek krabic. Každá jednotlivá součástka má svůj unikátní kód, který se uchovává v této tabulce v datovém typu `Decimal (8,0)`. Datový typ `Decimal` ukládá desetinná čísla s pevnou desetinnou čárkou ve formě řetězce. Proto je vhodný k ukládání velkých čísel.

Tabulka 1: Součástka

Součástka	
číslo_součástky	Decimal (8,0)

4.3.2 Tabulka Krabice

Tabulka `Krabice` uchovává informace i celých krabicích. Tj. název dané krabice, počet dostupných krabic k vypůjčení a číslo projektu, do kterého krabice patří. Jako primární klíč slouží unikátní číslo krabice. Název krabice se uchovává v řetězcovém datovém typu `Varchar`. Protože se s počtem dostupných krabic často manipuluje, je zvole typ `Integer`.

Tabulka 2: Krabice

Krabice	
číslo_krabice	Decimal (3,0)
číslo_projektu	Decimal (3,0)
název_krabice	Varchar (50)
počet_krabic	Int

4.3.3 Tabulka Součástka_krabice

Tabulka `Soucastka_krabice` slouží jako spojení tabulek `Krabice` a `Soucastka`. Protože v každé krabici je mnoho součástek a každá součástka může být ve více krabicích, vzniká nám relace M: N. Abychom tuto relaci odstranili, je tato tabulka nutná. Tabulka tak uchovává i informaci o počtu kusů konkrétní součástky v konkrétní krabici.

Tabulka 3: Součástka_krabice

Součástka_krabice	
číslo_součástky	Decimal (9,0)
číslo_krabice	Decimal (3,0)
počet_součástek	Int

4.3.4 Tabulka Projekt

Jednotlivé krabice jsou členěny v určitých projektech. Tabulka `Projekt` eviduje tyto projekty. Každá krabice se může, ale i nemusí, vyskytovat v jednom projektu.

Tabulka 4: Projekt

Projekt	
číslo_projektu	Decimal (3,0)
název	Varchar (50)
poskytovatel	Varchar (50)

4.3.5 Tabulka Příručka

Ke každé stavebnici existuje příručka. V příručce jsou návody a postupy pro krabici. Tyto příručky si lze samostatně vypůjčit. Tabulka `Prirucka` uchovává informace o těchto příručkách, např. počet dostupných příruček k vypůjčení atd.

Tabulka 5: Příručka

Příručka	
číslo_příručky	Decimal (3,0)
číslo_krabice	Decimal (3,0)
název_příručky	Varchar (50)
počet_příruček	Int

4.3.6 Tabulka Uživatel

Informace o uživatelích jsou uchovány v tabulce `Uzivatel`. Každý uživatel má svůj `nick`, neboli přezdívku, a `heslo`, pod který se přihlašuje do aplikace. Dále je zaznamenané skutečné jméno a příjmení uživatele a jeho kontaktní informace. Uživatelé v této tabulce nemají nic společného s databázovými uživateli. Aplikace jako celek přistupuje k databázi přes jediného databázového uživatele, který má přidělena práva na změny v celé databázi Katalog.

Tabulka 6: Uživatel

Uživatel	
nick	Varchar (50)
heslo	Varchar (50)
jméno	Varchar (50)
příjmení	Varchar (50)
email	Varchar (50)
telefon	Decimal (9,0)

4.3.7 Tabulka Výpůjčka

Každý uživatel si může vypůjčit krabici nebo příručku na určitý čas. K evidenci výpůjček slouží tabulka `Vypujcka`. V tabulce je uloženo od jakého data do jakého data je výpůjčka realizována a jakému uživateli tato výpůjčka patří. Každá výpůjčka je označena unikátním číslem.

Tabulka 7: Výpůjčka

Výpůjčka	
číslo_výpůjčky	Decimal (3,0)
nick	Varchar (50)
heslo	Varchar (50)
od	Date
do	Date

4.3.8 Tabulka Výpůjčka_krabice

Jelikož v každé výpůjčce může být více krabic a může být ve více výpůjčkách (více uživatelů si může půjčit stejnou krabici, je-li dostatečný počet kusů na vypůjčení) je nutná spojovací tabulka `Vypujcka_krabice`. Tato tabulka odstraní relaci M: N. Tabulka přiřazuje jednotlivým výpůjčka krabice a pamatuje si počet vypůjčených kusů krabice.

Tabulka 8: Výpůjčka_krabice

Výpůjčka_krabice	
číslo_výpůjčky	Decimal (3,0)
číslo_krabice	Decimal (3,0)
počet_vyp_krabic	Int

4.3.9 Tabulka Výpůjčka_příručka

Obdobnou funkci má i tabulka `Vypujcka_prirucka`. Tabulka přiřazuje výpůjčce jednotlivé vypůjčené příručky a jejich počet.

Tabulka 9: Výpůjčka_příručka

Výpůjčka_příručka	
číslo_výpůjčky	Decimal (3,0)
číslo_příručky	Decimal (3,0)
počet_vyp_příruček	Int

4.4 Použité databázové objekty

Správa vypůjčených krabic a vypůjčených příruček je řešena už na úrovni databáze Katalog. Je jednoduší a spolehlivější tento problém implementovat na úrovni databáze něž na vyšší úrovni samotné aplikace. V databázi proto existují triggery nad tabulkami `Vypujcka_Krabice` a `Vypujcka_Prirucka`.

4.4.1 Triggery nad tabulkou Výpůjčka_krabice

První trigger zajišťuje správný počet dostupných krabic při vypůjčování.

```
create trigger trigger_povlozeni
after insert on vypujcka_krabice
for each row
begin
      update krabice set krabice.pocet_krabic =
      krabice.pocet_krabic - new.pocet_vyp_krabic
      where krabice.cislo_krabice = new.cislo_krabice;
end;
```

Z kódu SQL je vidět, že tento trigger se spustí po vložení nové hodnoty do tabulky `Vypujcka_krabice`. Vložení do této tabulky znamená, že si uživatel právě vypůjčil nějakou krabici. V proměnné `new.pocet_vyp_krabic` si trigger uchová počet těchto vypůjčených krabic. Dále se pomocí příkazu update upraví hodnota počtu dostupných krabic v tabulce `Krabice`. Tedy od počtu dostupných krabic se odečte počet vypůjčených krabic. Samozřejmě u správné krabice. Takže číslo krabice se musí rovnat číslu půjčované krabice, které si trigger uchovává v proměnné `new.cislo_krabice`, což nám zajistí kód za klíčovým slovem `where`.

Tento druhý trigger zajistí správný počet dostupných krabic při vrácení výpůjček.

```
create trigger trigger_posmazani
after delete on vypujcka_krabice
for each row
begin
      update krabice set krabice.pocet_krabic =
      krabice.pocet_krabic + old.pocet_vyp_krabic
      where krabice.cislo_krabice = old.cislo_krabice;
end;
```

Z kódu SQL vyčteme, že tento trigger se spustí po smazaní nějakého záznamu z tabulky `Vypujcka_Krabice`. Smazání tohoto záznamu znamená, že uživatel zrušil výpůjčku a vrátí určitou krabici. Je tedy nutné upravit počet dostupných krabic. V proměnné `old.cislo_krabice` je uchováno číslo krabice, která byla vrácena. V proměnné `old.pocet_vyp_krabic` je uchován počet vrácených krabic. Proto trigger upraví počet dostupných krabic v tabulce `Krabice` tak, že ke stávajícímu počtu dostupných krabic přičte počet vrácených krabic.

4.4.2 Triggery nad tabulkou Výpůjčka_příručka

Obdobně jako při vypůjčování krabic řeší aktuálně dostupný počet příruček triggery. Princip a syntaxe těchto triggerů jsou velmi podobné jako triggery nad tabulkou `Vypujcka_krabice`. Proto se jim nebudeme věnovat tak podrobně.

První trigger po vložení záznamu do tabulky `Vypujcka_prirucka` snižuje počet dostupných příruček v tabulce `Prirucka`.

Druhý trigger se spustí po vymazání záznamu v tabulce `Vypujcka_Prirucka`, tj. po zrušení výpůjčky některé příručky. Trigger navýší počet aktuálně dostupných příruček o právě vrácené příručky.

4.5 Důležité SQL příkazy použité v aplikaci

Obslužná aplikace získává data z databáze pomocí SQL dotazů. Některé z nich jsou složitější, proto se na ně podíváme podrobněji.

- ✔ Pro zjištění údajů o aktuálně vypůjčených příruček a uživatelů, kteří tyto výpůjčky realizovali, je nutné spojit čtyři tabulky. Jména uživatelů získáme z tabulky `Uzivatel`, kterou spojíme s tabulkou `Vypujcka`, abychom získali informace o výpůjčce. Výslednou tabulku ještě spojíme přes tabulku `Vypujcka_prirucka` s tabulkou `Prirucka`, abychom získali název vypůjčené příručky a počet vypůjčených kusů.

```
select jmeno, prijmeni, prirucka.nazev_prirucky,
vypujcka_prirucka.pocet_vyp_prirucek, vypujcka.od, vypujcka.do
from uzivatel, vypujcka, vypujcka_prirucka, prirucka
where uzivatel.nick = vypujcka.nick
and vypujcka.cislo_vypujcky =
     vypujcka_prirucka.cislo_vypujcky
and prirucka.cislo_prirucky =
     vypujcka_prirucka.cislo_prirucky;
```

- ✔ Pro zjištěni počtu kusů konkrétní součástky v konkrétní krabici je třeba spojit tabulky `Krabice` a `Soucastka_Krabice` přes číslo krabice. Poté vybereme restrikcí jen záznamy týkající se konkrétní krabice a nakonec vybereme záznam pro konkrétní součástku.

```
select soucastka_krabice.pocet_soucastek
from soucastka_krabice, krabice
where soucastka_krabice.cislo_krabice =
     krabice.cislo_krabice and krabice.nazev_krabice like
          'E-Tech, Fischer Technik'
and soucastka_krabice.cislo_soucastky = 37679;
```

- ✔ Tento SQL dotaz není použit přímo v aplikaci, ale má důležitou roli při plnění databáze Katalog daty o krabicích a součástkách, z nichž jsou krabice složeny. Nejdříve se vkládaly údaje do tabulky `Soucastka_krabice`, tzn. že se přiřadila konkrétní součástka ke konkrétní krabici a doplnil se i kolik kusů těchto součástek je obsaženo v této krabici. Tím ovšem vznikne problém, protože jedna součástka může být součástí více krabic, ale v tabulce `Soucastka` se může vyskytovat každá součástka pouze jednou. Tento příkaz nám tedy vypíše čísla součástek, které jsou nové, tzn. že nejsou součástí už jiných dříve zadaných krabic. Tyto součástky jsou tedy následně doplněny do tabulky `Soucastka` a máme zajištěno, že nebudou vznikat duplicity.

```
select cislo_soucastky
from soucastka_krabice
where cislo_soucastky not in (
      select cislo_soucastky
      from soucastka_krabice
      where cislo_krabice <> 6
);
```

- ✔ V aplikaci je třeba určit, kolik kusů součástek je v celé databázi. Protože se součástka může vyskytovat v několika krabicích je nutné spojit tabulky `Soucastka_krabice` a `Krabice`. Navíc v databázi je vždy několik kusů každé krabice. Souhrnně řečeno v tabulce `Soucastka_krabice` je uchován počet kusů každé součástky v konkrétní krabici. V tabulce `Krabice` je uveden počet kusů celé krabice v databázi. Musíme tedy tyto dva údaje vynásobit, aby nám vyšel celkový počet kusů součástky v krabici. Nakonec musíme výsledek sečíst pomocí příkazu `SUM`, protože součástka se může vyskytovat ve více krabicích a seskupit dle čísla součástky pomocí příkazu `GROUP BY`, tím získáme celkový počet kusů každé součástky napříč celou databází.

```
select soucastka_krabice.cislo_soucastky as  SOUCASTKA, SUM
(soucastka_krabice.pocet_soucastek *
           krabice.pocet_krabic) as POCET
from soucastka_krabice, krabice
where soucastka_krabice.cislo_krabice =
      krabice.cislo_krabice
group by soucastka_krabice.cislo_soucastky;
```

Aplikace obsahuje ještě mnoho SQL dotazů. Jsou to ale jednoduché příkazy na spojení dvou tabulek. Není zapotřebí je zde uvádět. Jsou dostupné ve zdrojových kódech na přiloženém CD.

5 Aplikace Katalog

Nad databází Katalog pracuje obslužná aplikace Katalog. Pomocí této aplikace budou uživatelé přistupovat k datům uložených v databázi.

5.1 Popis struktury aplikace

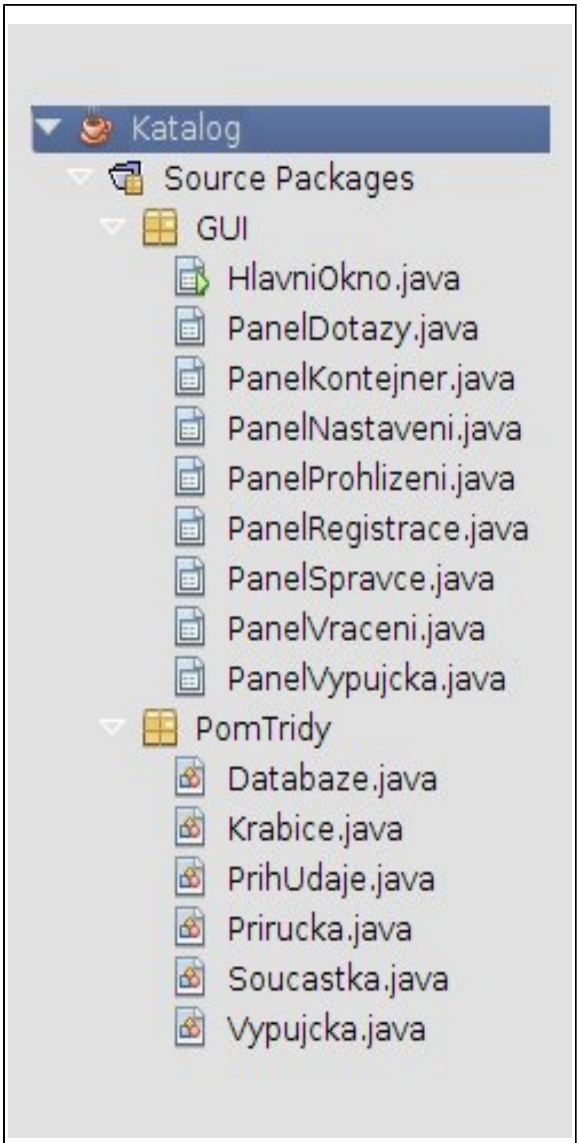
Aplikace je pro jednoduší použití rozdělena do dvou balíčků. Každý balíček obsahuje mnoho tříd nutných pro běh aplikace.

- ✔ **Balíček PomTřídy**

Tento balíček obsahuje tzv. pomocné třídy. Tyto třídy jsou navrženy s ohledem na databázové tabulky. Lze říci, že entitě z ze struktury databáze odpovídá třída v jazyce Java. Například k tabulce `Soucastka` existuje třída `Soucastka`, která zapouzdřuje informace o součástkách. Další důležitou třídou je třída `Databaze`, která slouží pro připojení aplikace k databázi (viz kapitola 6 Propojení MySQL a Java).

- ✔ **Balíček GUI**

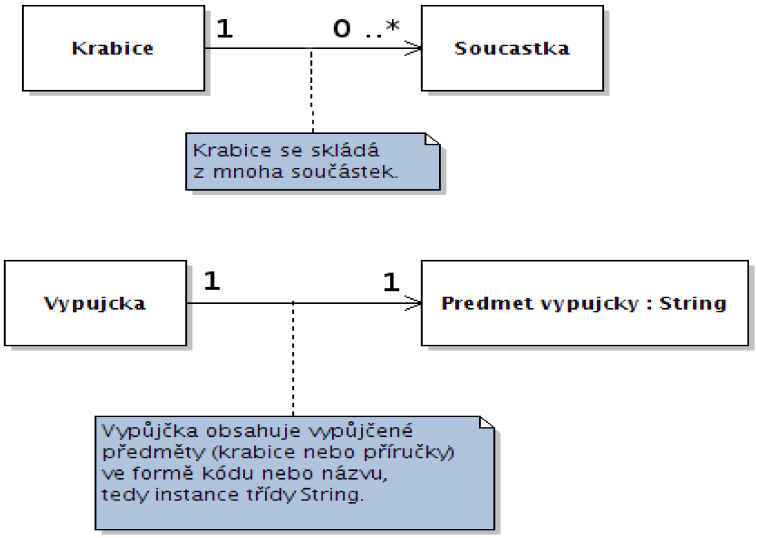
V tomto balíčku jsou obsaženy všechny grafické třídy. Většina obslužného kódu a algoritmů jsou obsaženy právě v těchto třídách.



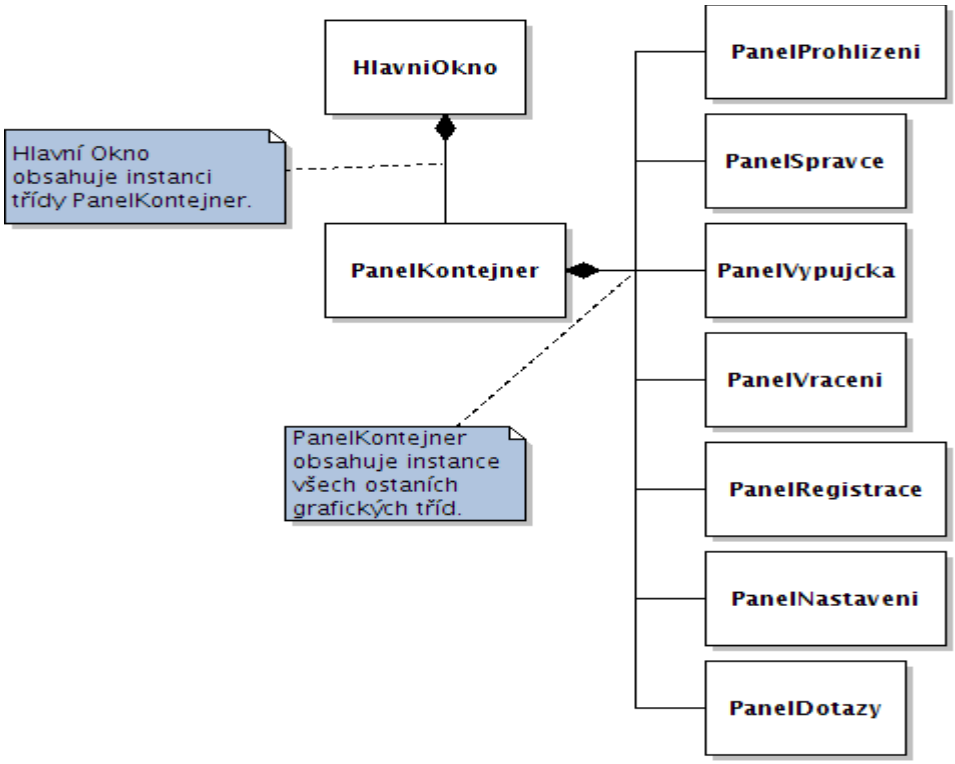
Obrázek 10: Struktura balíčků a tříd

5.2 Class diagramy balíčků

Pro pochopení vztahů mezi některými třídami jsou vhodné class diagramy jazyka UML.



Obrázek 11: Class diagram balíku PomTridy



Obrázek 12: Class diagram balíku GUI

5.3 Třídy v balíčku PomTřídy

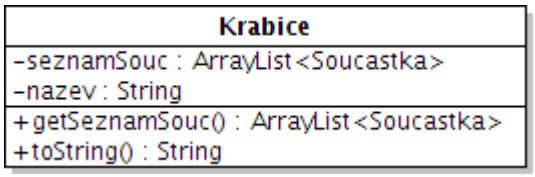
5.3.1 Třída Soucastka

Tato třída zapouzdřuje informace o součástkách z nichž jsou složeny krabice. Pro identifikaci součástky je použitý jedinečný kód, který se uchovává v atributu `kod`. Dále ke každé součástce náleží její `obrazek`, což je malá ikona použitá v aplikaci pro lepší přehlednost.

Obrázek 13: Ikona třídy Soucastka

5.3.2 Třída Krabice

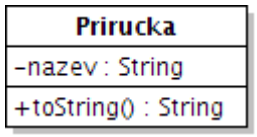
Třída `Krabice` odpovídá tabulce `Krabice`. Každá krabice je složena z mnoha součástek a má svůj název. Toto zapouzdřuje třída `Krabice`.



Obrázek 14: Ikona třídy Krabice

5.3.3 Třída Prirucka

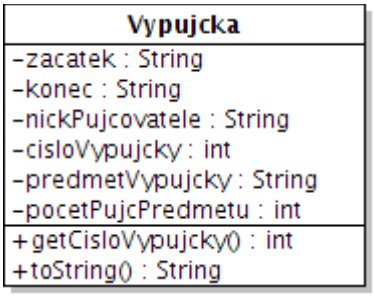
Tabulce `Prirucka` odpovídá stejnojmenná třída. Pro účely aplikace stačí uchovávat pouze název příručky.



Obrázek 15: Ikona třídy Prirucka

5.3.4 Třída Vypujcka

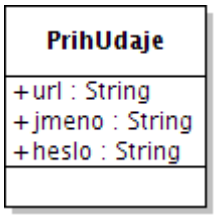
Pro práci s výpůjčkami v aplikaci je nutná třída `Vypujcka`. Zapouzdřuje informace o výpůjčce. Číslo výpůjčky je unikátní identifikátor každé výpůjčky. Dále třída obsahuje předmět výpůjčky, počet půjčených předmětů, přezdívku uživatele. Začátek a konec jsou uložená data platnosti výpůjčky.



Obrázek 16: Ikona třídy Vypujcka

5.3.5 Třída PrihUdaje

Tato třída má pouze funkci uchovávat informace nutné k propojení databáze s aplikací. Pro připojení k MySQL databázi potřebujeme tzv. url databáze. Tedy cestu ke stroji, na kterém MySQL databáze běží. Například pro připojení k databázi na lokálním počítači bude mít url tvar „localhost/katalog“. Dále potřebujeme informaci o uživateli, který se k databázi připojuje. MySQL si vede tabulku databázových uživatelů. Každý uživatel může mít různá práva přístupu k jednotlivým tabulkám uloženým v databázi. Uživatel se tedy identifikuje pomocí jména a hesla. Atributy jsou veřejné a mají k nim přístup všechny ostatní třídy. Tyto údaje si aplikace ukládá do souboru.



Obrázek 17: Ikona třídy PrihUdaje

5.4 Třídy v balíčku GUI

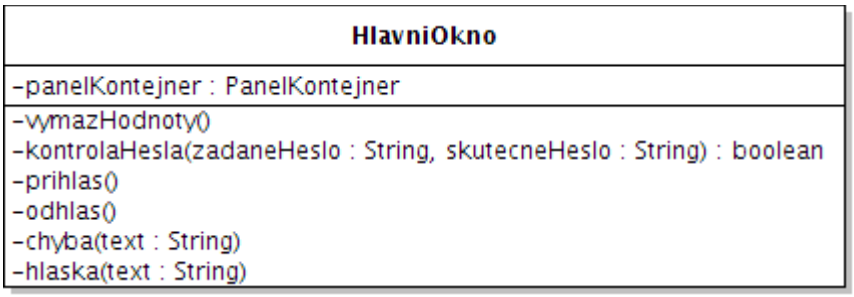
Dále uvedené ikony grafických tříd jsou zjednodušeny. Nejsou v nich uváděny metody a atributy automaticky generované prostředím NetBeans. Uvádím pouze své vlastní metody a atributy důležité pro funkčnost aplikace napsané ručně.

Pro názornost si ukážeme náhled na spouštěnou aplikaci. Základní je princip záložkového systému panelů. Každému panelu odpovídá jedna grafická třída.

Obrázek 18: Náhled na aplikaci

5.4.1 Třída HlavniOkno

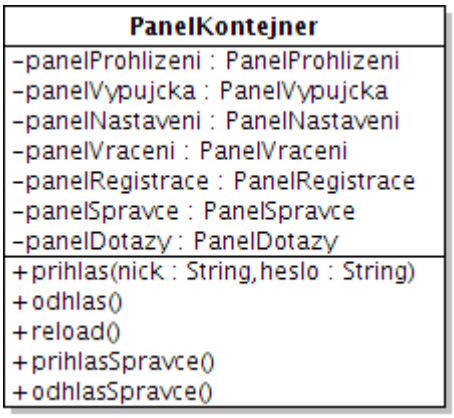
Hlavní okno aplikace je třída typu `JFrame`. Je to hlavní kontejner aplikace, všechny ostatní grafické třídy jsou jeho součástí. V hlavním okně je implementováno přihlašování a odhlašování uživatelů do celé aplikace. O to se starají metody `prihlas()` a `odhlas()`. Jediným atributem je instance třídy `Kontejner`. Metody `chyba()` a `hlaska()` slouží k tvorbě upozorňujících modální dialogů. `chyba()` slouží hlavně k obsluze výjimek. Uživateli se tak zobrazí informace o problému v aplikaci. `hlaska()` zobrazuje pouze informativní dialog, například po úspěšném přihlášení apod. Tyto dvě metody jsou součástí i některých další tříd.



Obrázek 19: Ikona třídy HlavniOkno

5.4.2 Třída PanelKontejner

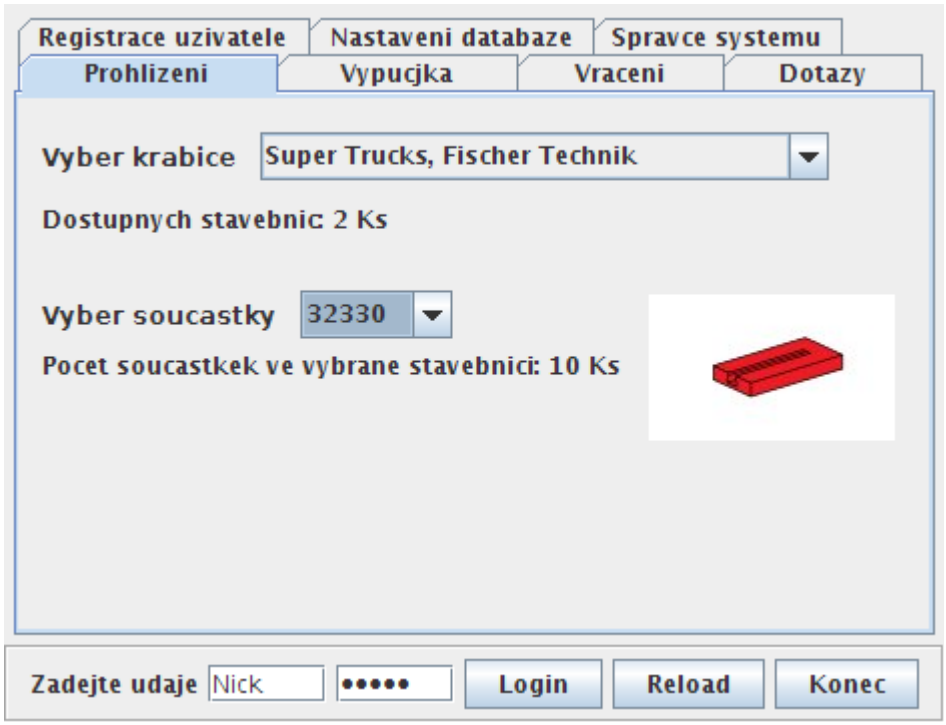
Třída `PanelKontejner` obsahuje všechny ostatní panely a stará se o záložkový systém panelů. Všechny ostatní grafické panely jsou atributy této třídy. Metody `prihlas()` a `odhlas()` pouze předávají informace o správném přihlášení/odhlášení uživatelů v třídě `HlavníOkno` dalším třídám. Při přihlášení správce se uplatňují metody `prihlasSpravce()` a `odhlasSpravce()`. Pochopitelně správce má více možností pracovat s aplikací. Může si například prohlížet výpůjčky všech ostatních uživatelů apod. Z toho důvodu se musí oddělit přihlášení správce a obyčejného uživatele. Metoda `reload()` se používá při znovunačtení celé aplikace, například z důvodu změny nastavení databáze.



Obrázek 20: Ikona třídy PanelKontejner

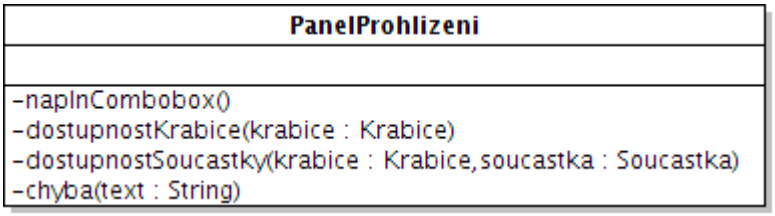
5.4.3 Třída PanelProhlizeni

`PanelProhlizeni` slouží k přehlednému prohlížení všech krabic a součástek v databázi (viz obr.). V prvním comboboxu si vybereme konkrétní krabici. Aplikace nám ukáže počet krabic dostupných k vypůjčení. V druhém combobuxu se nám načtou všechny součástky krabice vybrané v prvním comboboxu. Při prohlížení součástek se nám zobrazuje, kolik kusů této součástky je v krabici z prvního comboboxu a malý náhled, jak tato součástka vypadá.



Obrázek 21: Náhled aplikace – Prohlížení

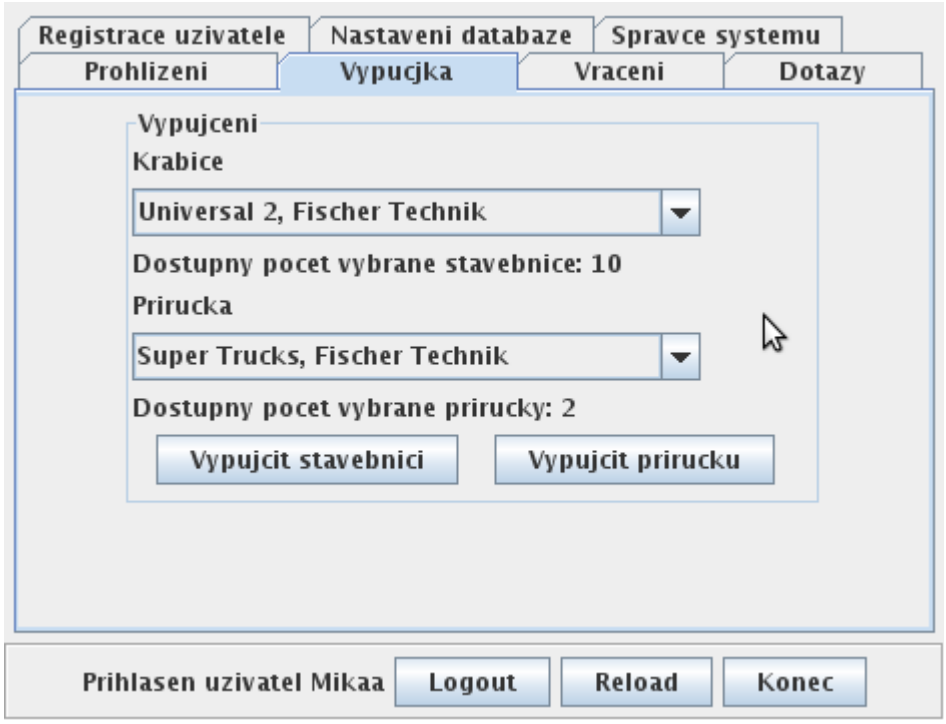
Metoda `naplnCombobox()` se stará o plnění comboboxů jednotlivými krabicemi a součástkami. Zobrazování informací o dostupnosti krabice/součástky zajišťují metody `dostupnostSoucastky(…)` a `dostupnostKrabice(…)`.



Obrázek 22: Ikona třídy PanelProhlizeni

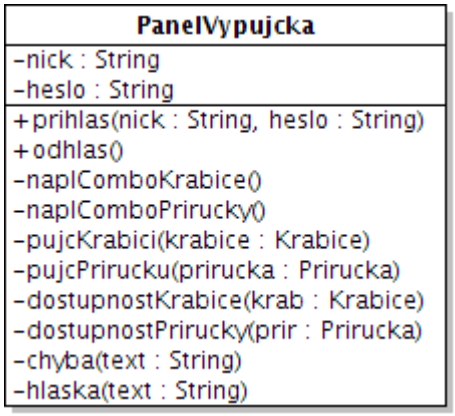
5.4.4 Třída PanelVypujcka

`PanelVypujcka` v aplikaci zajišťuje vypůjčování krabic a příruček k nim. Uživatel si po přihlášení jednoduše vybere, kterou krabici (příručku) si chce vypůjčit a klikne na tlačítko Vypujcit stavebnici (Vypujcit prirucku). Poté se ho aplikace zeptá, kolik kusů dané krabice si chce vypůjčit. Dále uživatel zadá předpokládaný datum vrácení krabice a nakonec se daná výpůjčka zaznamená do databáze.



Obrázek 23: Náhled aplikace – Výpůjčka

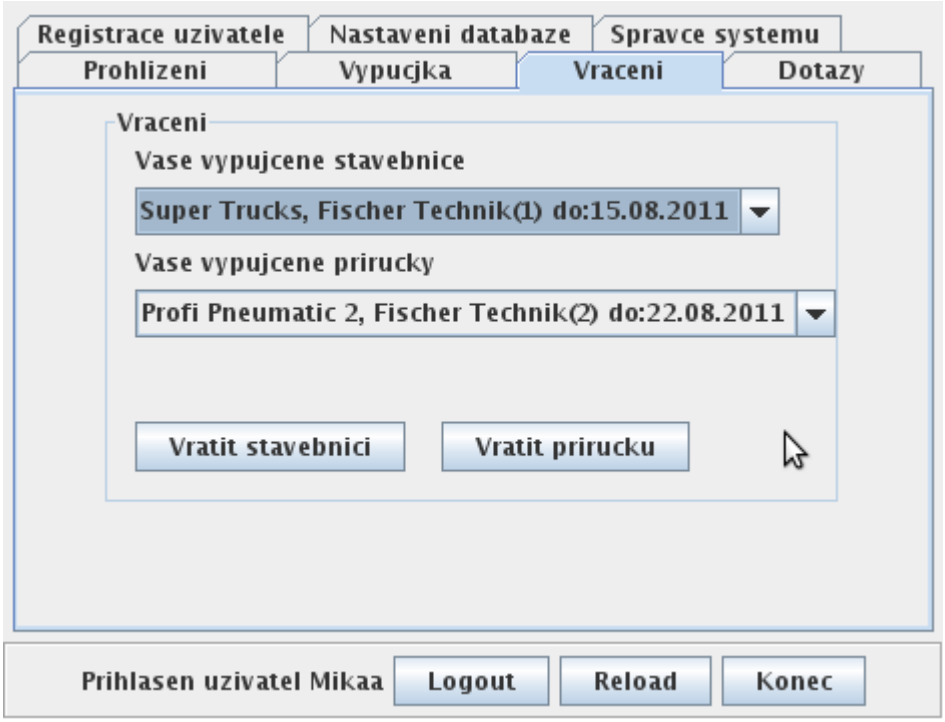
Je nutné si pamatovat uživatele, který je přihlášen, proto již zmíněné atributy `nick` a `heslo`. Metoda `prihlas()` zajistí správné přihlášení uživatele. Metody `naplComboKrabice()` a `naplComboPrirucky()` naplní dané comboboxi krabicemi a příručkami tak, aby si mohl uživatel pohodlně některou položku vybrat. Metody `dostupnostKrabice(…)` a `dostupnostPrirucky(…)` zobrazují počty dostupných stavebnic a příruček k vypůjčení. Samotné vypůjčení, tedy vytvoření záznamu o výpůjčce v databázi zajistí metody `pujcKrabici(…)` nebo `pujcPrirucku(…)`. Při odhlášení uživatele se uplatňuje metoda `odhlas()`, která zajistí korektní a bezpečné odhlášení.



Obrázek 24: Ikona třídy PanelVypujcka

5.4.5 Třída PanelVraceni

`PanelVraceni` naopak zajišťuje vrácení už vypůjčených stavebnic, popřípadě příruček. Uživatel v comboboxech vidí svoje aktuálně vypůjčené předměty. Může se podívat kolik dní mu zbývá z výpůjčky, kolik kusů a jaké krabice má vypůjčeno atd. Po vybrání určité výpůjčky může kliknutím na tlačítko danou výpůjčku zrušit, tedy vrátit daný předmět. V databázi se samozřejmě po vrácení provedou změny.



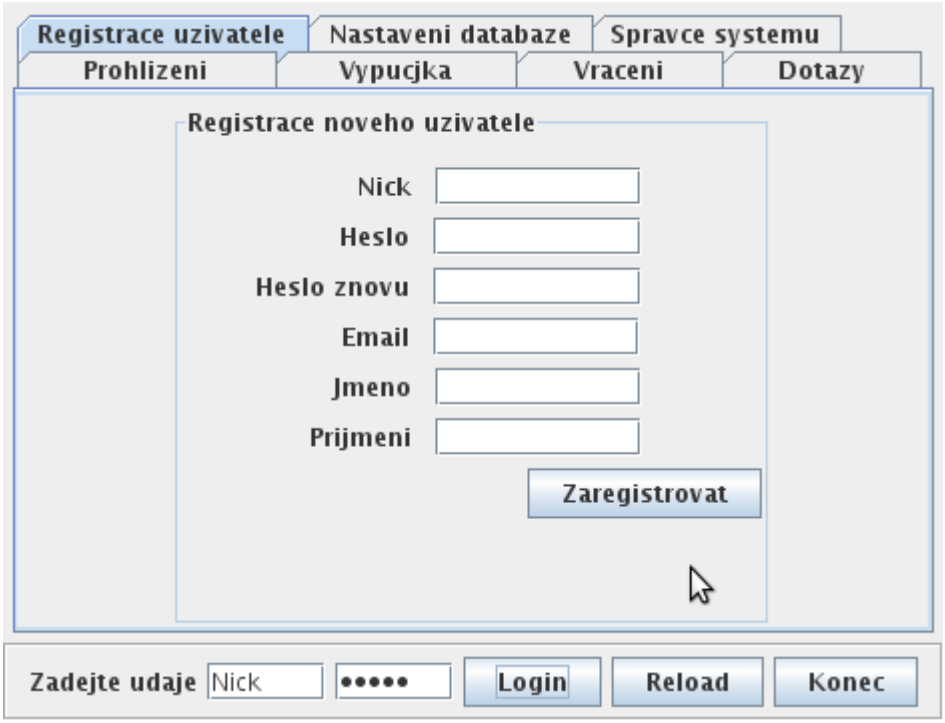
Obrázek 25: Náhled aplikace – Vrácení

Je třeba identifikovat právě přihlášeného uživatele pomocí atributů `nick` a `heslo`. Metody `naplnComboVypujckyKrabice(…)` a `naplnComboVypujckyPrirucky(…)` naplní jednotlivé comboboxi aktuálními výpůjčkami přihlášeného uživatele. Metoda `kontrola()` ještě zkontroluje oba comboboxi. Při vrácení se aktivují metody `vratKrabici(…)` nebo `vratPrirucku(…)`, které zaznamenají informace do databáze. Metody `prihlas()` a `odhlas()` plní stejnou funkci jako u `PanelVypujcka`.

Obrázek 26: Ikona třídy PanelVraceni

5.4.6 Třída PanelRegistrace

Tento panel slouží k registraci nových uživatelů v aplikaci. Uživatel vyplní důležité informace o své osobě. Pro pozdější přihlášení a práci s aplikací je důležitý přihlašovací `nick` a `heslo`. Další údaje se použijí například k upozornění na to, že uživatel nevrátil své výpůjčky včas apod.



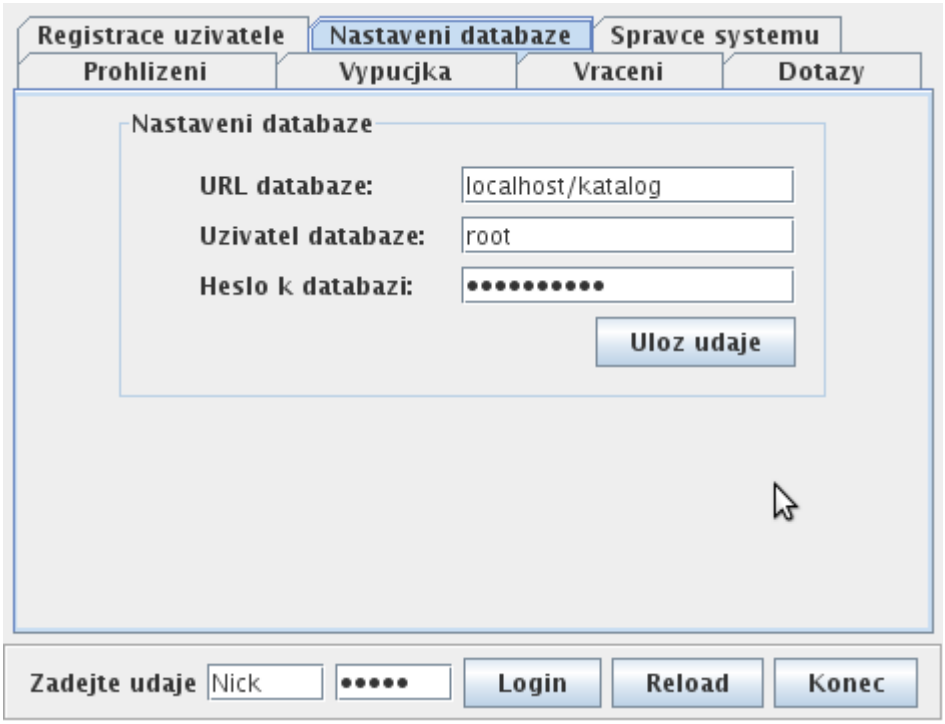
Obrázek 27: Náhled aplikace – Registrace

Důležitá je metoda `registruj()`, která zajistí celý proces registrace. Pro kontrolu překlepů zadává uživatel svoje heslo dvakrát. Zda zadal obě hesla stejná nám zjistí metoda `kontrolaHesla(…)`. Po úspěšné registraci metoda `vymaz()` vymaže uživatelem zadaná údaje z aplikace. Tyto údaje jsou samozřejmě uloženy do databáze.

Obrázek 28: Ikona třídy PanelRegistrace

5.4.7 Třída PanelNastavení

Pro nastavení přístupových údajů nutných pro přihlášení do databáze se používá `PanelNastaveni`. Tento panel úzce spolupracuje se třídou `PrihUdaje` (viz kapitola 5.2.5). Pro úspěšné připojení k databázi je nutné zadat URL – neboli cestu ke stroji, na kterém databáze běží a název databáze. Aplikace má přednastaveno připojovat se k databázi na lokálním počítači, tedy k počítači, na kterém běží jak aplikace, tak databáze. Uživatel a heslo slouží k identifikaci databázového uživatele, který má právo prohlížet tabulky v databázi Katalog.



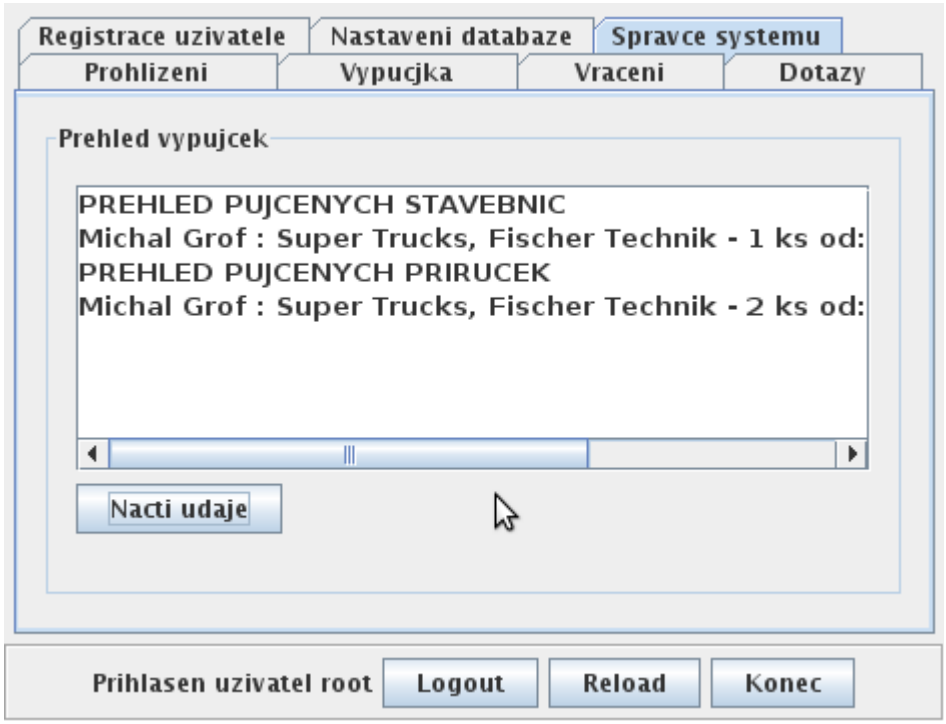
Obrázek 29: Náhled aplikace – Nastavení databáze

Jen dvě metody používá `PanelNastaveni`. Údaje o nastavení si aplikace ukládá do souboru, proto metoda `nactiUdaje()` načte tyto hodnoty ze souboru do aplikace. Po případné úpravě těchto údajů pak metoda `ulozUdaje()` opět uloží do souboru. Při změně těchto nastavení se musí aplikace znovu spustit, aby mohla načítat informace z databáze. To jednoduše provedeme pomocí tlačítka Reload v hlavním okně aplikace (viz obr.).

Obrázek 30: Ikona třídy PanelNastaveni

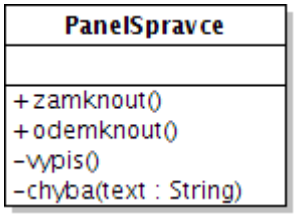
5.4.8 Třída PanelSpravce

`PanelSpravce` je přístupný pouze správci aplikace. Ten si může přehledně zobrazit všechny vypůjčené předměty všech ostatních uživatelů. Tlačítko Nacti udaje tyto informace získá z databáze a přehledně je zobrazí. Pro běžné uživatele je tato možnost nepřístupná.



Obrázek 31: Náhled aplikace – Správce systému

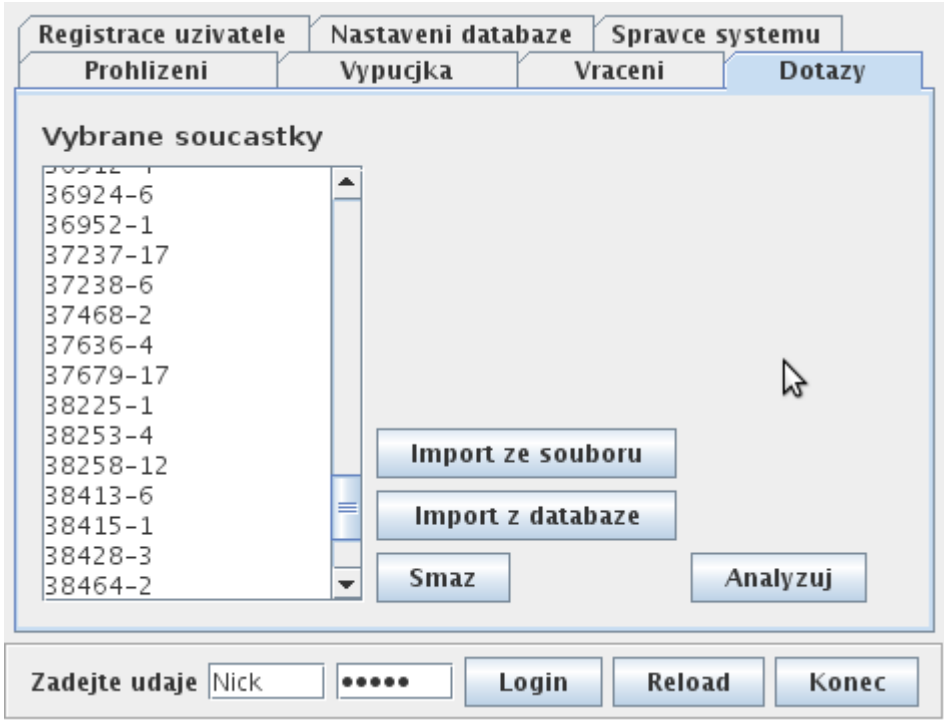
Při přihlášení správce se panel odemkne metodou `odemknout()` a obdobně se panel zamkne při odhlášení správce pomocí metody `zamknout()`. Metoda `naplnTabulku()` načítá informace z databáze a zobrazuje je správci.



Obrázek 32: Ikona třídy PanelSpravce

5.4.9 Třída PanelDotazy

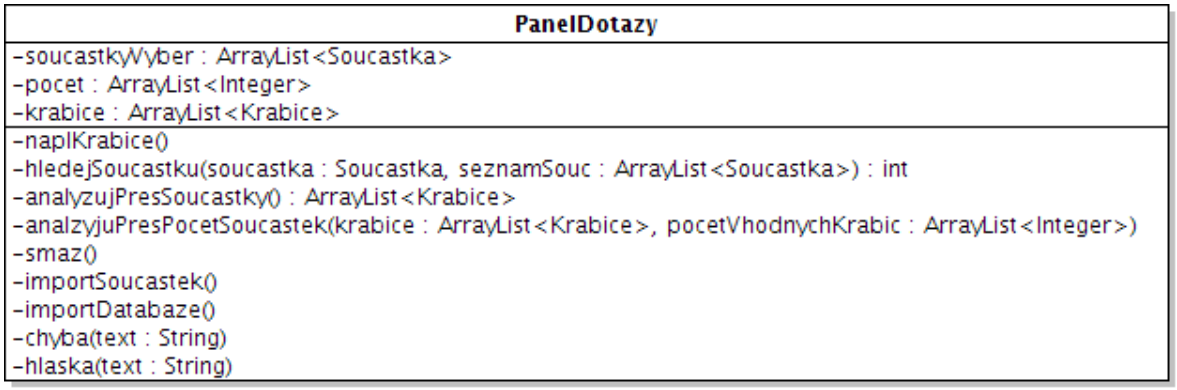
Tento panel umí zodpovídat dotazy nad databází. Řekněme, že uživatel má k dispozici určitou množinu součástek a chce vědět, jaké krabice by z těchto součástek mohl sestavit. Uloží seznam součástek do textového souboru naimportuje jej do aplikace přes `PanelDotazy` a aplikace vyhodnotí, jaké krabice z těchto součástek lze sestavit. Další možností je, že uživatel chce mít přehled, kolik a jakých krabic lze sestavit ze všech aktuálně dostupných součástek. Může tady naimportovat seznam všech součástek přímo z databáze automaticky pomocí tlačítka Import z databáze.



Obrázek 33: Náhled aplikace – Dotazy

Místo klasických polí u atributů jsou použita `ArrayList<…>` z jazyka Java. `ArrayList<…>` poskytuje jednodušší a komplexnější práci než klasické pole. Tedy pole `soucastkyVyber` obsahuje všechny součástky, nad kterými se spouští dotazy na sestavovaní stavebnic. Počet kusů každé součástky je uložen v poli `pocet`. Plnění polí `soucastkyVyber` a `pocet` probíhá vždy stejně a to tak, že každému indexu součástky v poli `soucastkyVyber` přesně odpovídá index v poli `pocet`. Tedy na stejném indexu v jednom poli je součástka a v druhém počet kusů této součástky. Pole `krabice` se plní všemi krabicemi v databázi. Z toho pole se posléze získávají informace nutné pro dotazování.

Metoda `importSoucastek()` naimportuje (tedy naplní pole `soucastkyVyber` a `pocet`) seznam součástek z textového souboru. Obdobně metoda `importDatabaze()` naimportuje seznam všech součástek z databáze. Metoda `smaz()` vymaže pole `soucastkyVyber` a `pocet` pro další dotaz. Metoda `hledejSoucastku()` vyhledává index určité součástky v poli součástek. Vrací tedy číslo indexu, na kterém se součástka nachází. Není-li součástka v poli, vrací metoda číslo -1. Při dotazu se nejdříve spouští metoda `analyzujPresSoucastky()`. Tato metoda zkontroluje, jaké krabice lze sestavit z výběru součástek, a vrátí je v poli. Takže víme, jaké krabice lze sestavit ze součástek ve výběru, ale nevíme, kolik kusů krabic lze sestavit. To nám zodpoví metoda `analyzujPresPocetSoucastek(ArrayList<Krabice> krabice, ArrayList <Integer> pocetVhodnychKrabic)`. Protože metoda vrací více než jednu hodnotu, jsou návratové hodnoty předávány pomocí parametrů. Metoda tedy v parametru `ArrayList<Krabice> krabice` vrací pole vhodných stavebnic a v parametru `ArrayList <Integer> pocetVhodnychKrabic` vrací počet kusů těchto krabic.



Obrázek 34: Ikona třídy PanelDotazy

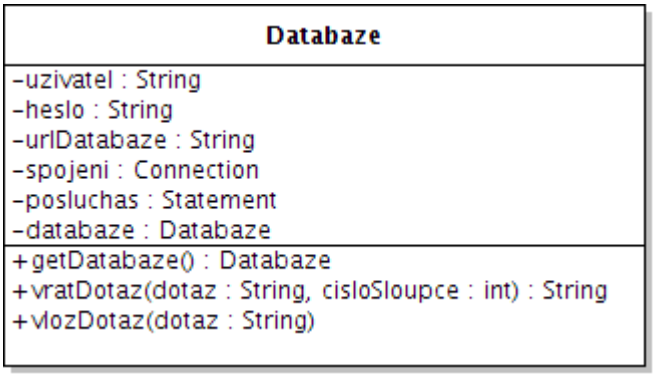
6 Propojení MySQL a Java

Propojení aplikace napsané v jazyku Java a databáze MySQL není zrovna triviální věc. Pro připojení programu napsaného v jazyce Java k databázi MySQL budeme potřebovat speciální konektor. Jedná se o tzv. Java Database Connectivity (JDBC). JDBC poskytuje rozhraní pro propojení relačních databází a jazyka Java. Pro každý databázový systém si musíme obstarat tento konektor. Pro MySQL je ke stažení na stránkách www.mysql.com, kde najdeme i konektory pro další programovací jazyky.[4]

V aplikaci se o připojení k MySQL databázi stará třída Databaze.

6.1 Třída Databaze

Pro identifikaci databázového uživatele a samotné databáze musíme mít v atributech jméno uživatele (`uzivatel`), jeho heslo (`heslo`) a cestu k počítači na kterém MySQL databáze běží (`urlDatabáze`). Třídy `Connection a Statement` jsou nutné pro připojení k databázi. Instance třídy `Connection` zajišťuje přímo spojení s databází. Instance třídy `Statement` provádí datové operace nad databází. Instanci `Statement` nám poskytuje instance třídy `Connection` (viz níže).



Obrázek 35: Ikona třídy Databaze

Je vhodné, aby v celé aplikaci existovala pouze jediná instance třídy `Databaze`. Nedochází tím k vícenásobnému přístupu do databáze. Proto je v této třídě použit návrhový vzor Singleton, který tento problém elegantně vyřeší. Vše funguje tak, že třída obsahuje statickou instanci sebe samé (`private static Databaze databaze`) a veřejnou statickou metodu `public static Databaze getDatabaze()`. V této metodě je volán konstruktor samotné třídy `Databaze`, který je privátní. Tím je zajištěno, že ostatní třídy budou mít přistup k instanci třídy `Databaze` a zároveň to bude jediná instance v rámci celé aplikace.[6]

```
public static Databaze getDatabaze(){
        if (databaze == null) {
            databaze = new Databaze(PrihUdaje.url, PrihUdaje.jmeno,
                PrihUdaje.heslo);
        }
        return databaze;
}
```

6.1.1 Konstruktor třídy Databaze

Konstrukce třídy probíhá následovně: Nejprve naplníme atributy `urlDatabaze`, `uzivatel` a `heslo`. Poté je nutné zaregistrovat konektor pomocí příkazu `Class.forName („")` aby se mohlo s databází pracovat.

```
private Databaze (String urlDatabaze, String uzivatel,
      String heslo){
            this.urlDatabaze = urlDatabaze;
            this.uzivatel = uzivatel;
            this.heslo = heslo;
            Class.forName ("com.mysql.jdbc. Driver");
}
```

6.1.2 Metoda String vratDotaz (String dotaz, int cisloSloupce)

Tato metoda umí provádět SQL dotaz `SELECT`. SQL dotaz se předá pomocí parametru `dotaz`. Parametr `cisloSloupce` určuje, který sloupec ve výsledku SQL dotazu se použije. Výsledek se vrátí jako datový typ `String` s tím, že jednotlivé vrácené hodnoty jsou odděleny znakem"\n". Lze tedy jednoduše pomocí metody `trim()` rozdělit tento řetězec na jednotlivé hodnoty.

Instanci třídy `Connection` spojení získáme pomocí `DriverManageru`. Proto bylo nutné v konstruktoru správně zaregistrovat konektor. Metodě `getConnection()` se předává řetězec ve tvaru „jdbc:mysql://host:port/název_databáze". Nyní nám `spojeni` (instance třídy `Connection`) poskytne instanci třídy `Statetment`, `posluchac` přes metodu `createStatement()`. Právě `posluchac` umí pomocí metody `executeQuery(…)` spustit námi požadovaný `SELECT` nad databází.

```
public String vratDotaz (String dotaz, int cisloSloupce){
      this.spojeni = DriverManager.getConnection
            ("jdbc: mysql://" + this.urlDatabaze,
            this.uzivatel, this.heslo);
      this.posluchac = spojeni.createStatement ();
      ResultSet vysledek = this.posluchac.executeQuery (dotaz);
      String strVysledek = "";
      while (vysledek.next ()) {
            strVysledek += vysledek.getString (cisloSloupce);
            strVysledek += "\n";
      }
      return strVysledek;
}
```

6.1.3 Metoda void vlozDotaz (String dotaz)

Pro SQL příkazy `UPDATE`, `INSERT`, `DELETE` slouží metoda `vlozDotaz(…)`. Postup je obdobný jako u metody `vratDotaz(…)`. Rozdíl je v tom, že `posluchac` volá metodu e`xecuteUpdate(…)`. A navíc se po skončení dotazu volá metoda spojeni `close()`, která zavře spojení. To způsobí `COMMIT` nad databází a okamžité zapsání nových hodnot.

```
public void vlozDotaz (String dotaz){
      this.spojeni = DriverManager.getConnection
            ("jdbc: mysql://" + this.urlDatabaze,
            this.uzivatel,   this.heslo);
      this.posluchac = spojeni.createStatement ();
      this.posluchac.executeUpdate (dotaz);
      this.spojeni.close ();
}
```

7 Závěr

Myslím si, že aplikace v tomto stavu je připravena na provoz, i přestože některé chyby vyloučit nemůžu. To už je otázka testování v reálném nasazení. Do budoucna by se aplikace mohla ještě rozšířit o další funkce. Například není zcela vyřešena správa uživatelů. Chybí možnost smazat uživatele nebo změnit kontaktní informace u již registrovaného uživatele. Obdobně to platí i o přidávání dalších nových stavebnic nebo součástek. Toto se prozatím musí řešit ručními zásahy do databáze, z čehož plyne nebezpečí poškození integrity dat.

Práce jak s databázovým systémem MySQL, tak s jazykem Java mě určitě obohatila. Získal jsem nové zkušenosti s rozsáhlejší aplikací napsanou dle pravidel OOP. Rozšířil jsem také své znalosti v oblasti tvorby struktury databáze a samozřejmě složitějšího dotazování pomocí jazyka SQL. Zajímavé bylo také plnit celou databázi údaji o celkovém obsahu přes 2000 záznamů.

Seznam literatury

[1] Netbeans.org [online]. 2011 [cit. 2011-04-27]. Vítejte u NetBeans a na stránkách www.netbeans.org. Dostupné z WWW: <http://netbeans.org/index_cs.html>.

[2] LinuxSoft.cz [online]. 2007 [cit. 2011-04-27]. Seriál o MySQL. Dostupné z WWW: <http://www.linuxsoft.cz/article_list.php? id_kategory=232>.

[3] HAWLITZEK, Florian. *Java 2: příručka programátora*. první vydání. Praha: Grada Publishing, 2002. 316 s. ISBN 80-247-9060-2.

[4] HREBENAR, Jiří. Java – jednoduchý přístup k databázi MySQL. Blogy Živě [online]. 24. 8. 2009, 1, [cit. 2011-04-26]. Dostupný z WWW: <http://programovani.blog.zive.cz/2009/08/java-jednoduchy-pristup-k-databazi-mysql/>.

[5] RNDr. Žák, David, Ph. D.. *Databázové systémy I.* (přednášky) Pardubice: UPCE, 2009.

[6] HREBENAR, Jiří. Java - návrhový vzor Singleton. Blogy Živě [online]. 28. 8. 2009, 1, [cit. 2011-05-06]. Dostupný z WWW: <http://programovani.blog.zive.cz/2009/08/java-navrhovy-vzor-singleton/>.