UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Desková hra Monopoly v 2D Javě

Tomáš Rychlík

Bakalářská práce
2011

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

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Tomáš RYCHLÍK**
Osobní číslo:	**I07772**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Desková hra Monopoly v 2D Javě**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem bakalářské práce je pomocí programovacího jazyka Java převést klasickou deskovou hru do počítačové podoby.
Teoretická část:
Popis programovacího jazyka Java a výhod spjatých s použitím grafického uživatelského prostředí (s důrazem na 2D knihovnu). Popsat využití jazyka při vývoji a tvorbě širokého spektra počítačových her.
Implementační část:
Zpracování klasické deskové společenské hry Monopoly v Javě pro dva hráče na jednom počítači. Vytvoření grafického vstupně/výstupního rozhraní. Zakomponování zvukových efektů.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

***DAVISON, A.: Programování dokonalých her v Javě. 1. vydání, Computer Press, Brno, 2006. 902 s. ISBN 80-7226-944-5.**
***HEROUT, P.: Java - grafické uživatelské prostředí a čeština. 2. vydání. KOPP, České Budějovice, 2007. 316 s. ISBN 978-80-7232-328-9.**
***HEROUT, Pavel. Java - bohatství knihoven. 2. upravené vydání, KOPP, České Budějovice, 2006. 251 s. ISBN 80-7232-288-5.**

Vedoucí bakalářské práce: **Ing. Zdeněk Šilar**
Katedra informačních 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 01. 05. 2011 Tomáš Rychlík

Poděkování

Chtěl bych velmi poděkovat vedoucímu mé bakalářské práce Ing. Zdeňku Šilarovi za veškerou odbornou pomoc a cenné rady při zpracovávání mé bakalářské práce.

Děkuji také mým rodičům a Mgr. Alici Papouškové a jejím rodičům za veškerou psychickou a hmotnou podporu.

Na závěr také děkuji Martinu Brethovi a Janu Krulichovi za veškeré připomínky během jejich vyčerpávajícího testování praktické části mé bakalářské práce.

Anotace

Cílem bakalářské práce je pomocí programovacího jazyka Java převést klasickou deskovou hru do počítačové podoby. V teoretické části se zaměřím na popis programovacího jazyka Java a výhod spojených s použitím grafického uživatelského prostředí. S tím také souvisí popis jednotlivých komponent a jejich vlastností. V další části se budu zabývat grafikou v Javě, demonstruji práci s obrázky, efekty a také zmíním možnosti vláken a v neposlední řadě také práci se zvukem. V poslední části se zaměřím na samotný popis praktické části bakalářské práce, jejího zpracování a zmíním také některé zajímavé problémy, se kterými jsem se setkal.

Klíčová slova

JAVA, Monopoly, hra, 2D, grafika, GUI, programování, swing, NetBeans

Title

Desktop game Monopoly in 2D Java.

Annotation

The objective of the bachelor´s theses is a transformation of a classical board game into a computer form with the aid of the programming language Java. In the theoretical part I will focus on the description of the programming language Java and advantages connected with the application of the Graphical User Interface. The following description of components and their features are also related to this topic. In the next part I will deal with graphics in Java and I will demonstrate picture work and effects. I will also mention possibilities of the threads and not least the sound work. In the final part I will concentrate on the description of the practical part of the bachelor´s thesis itself and the processing. I will mention certain interesting problems which I have analysed.

Keywords

JAVA, Monopoly, game, 2D, graphic, GUI, programming, swing, NetBeans

Obsah

Seznam zkratek

API	Application programming interface
AWT	Abstract Window Toolkit
EPL	Eclipse Public License
GIF	Graphics Interchange Format
GPL	General Public License
GUI	Graphical User Interface
IDE	Integrated Development Environment
JAR	Java ARchiver
JDK	Java Development Kit
JFC	Java Foundation Classes
JPEG	Joint Photographic Experts Group
JRE	Java Runtime Environment
JVM	Java Virtual Machine
OOP	Object Oriented Programming
PNG	Portable Network Graphics

Seznam obrázků

Seznam tabulek

1 Jazyk Java

1.1 Základní popis programovacího jazyka Java

Programovací jazyk Java vznikal pod hlavičkou firmy Sun Microsystems od devadesátých let dvacátého století. Původní myšlenkou bylo vytvořit kvalitní programovací jazyk pro tvorbu programů pro vestavěná zařízení. Dále se také uvažovalo o využití jazyka při vytváření webových aplikací.

Java je objektově orientovaný jazyk vyšší úrovně, který byl tvořen podle principů programovacích jazyků C a C++. Oficiálně byla Java představena v květnu v roce 1995. Od roku 2006 je jazyk uvolněn, s několika výjimkami, pod licencí GNU GPL.

Hlavní výhody programovacího jazyka Java jsou jednoduchost, nezávislost na platformě a přenositelnost, bezpečnost a výkonnost.

1.2 Jak získat JAVU

V roce 2009 koupila firma Oracle Corporation firmu Sun Microsystem. Z tohoto důvodu lze Javu bezplatně stahovat buď z oficiálních stránek společnosti Oracle[1], nebo přímo z webových stránek java.com[2]. V době psaní této práce je k dispozici nejnovější verze 6 update 24.

Existují dvě možnosti při získávání Javy. Pro spouštění aplikací nebo apletů stačí stáhnout a nainstalovat JRE obsahující JVM. Pro tvorbu aplikací a apletů je nutné stáhnout a nainstalovat rozšířený balíček JDK, který obsahuje IDE a JRE.

1.3 Tvorba a spouštění aplikací

Pro vytváření aplikací v Javě je nutné mít nainstalované JDK, což je základní balíček nabízející nástroje pro tvorbu aplikací v Javě. Pro vytváření složitějších aplikací je pak vhodné využít některé z kvalitních vývojových prostředí, jejichž přehled je uveden v další kapitole.

Nejdůležitější části, které obsahuje balíček JDK, jsou JRE pro běh programů, překladač zdrojového kódu, debugger pro ladění kódu a Java Core API obsahující základní sadu knihovních tříd.

Po naprogramování aplikace a jejím úspěšném zkompilování lze vytvořit spustitelný soubor ve formátu JAR.

[1] http://www.oracle.com/technetwork/java/index.html
[2] http://java.com/en/download/index.jsp

2 Vývojová prostředí

2.1 Programovací nástroje

Pro programování různorodých aplikací se využívají specializované nástroje souhrnně označované jako vývojové prostředí, které bývají nejčastěji označované anglickou zkratkou IDE. V minulosti byla vývojová prostředí přizpůsobena jednomu konkrétnímu programovacímu jazyku. Dnes již existují vývojová prostředí, která umožňují vytvářet aplikace v různých programovacích jazycích. Takové prostředí se označuje jako vícejazykové vývojové prostředí. Klasickým příkladem takovéhoto softwaru je Microsoft Visual Studio či NetBeans. Například v již zmíněném NetBeans lze vytvářet aplikace v programovacím jazyku Java, C, C++, HTML, Perl, Python.

Pro programování aplikací v Javě můžeme využít například některé z těchto vývojových prostředí:

- NetBans,
- Eclipse,
- Oracle JDeveloper,
- BlueJ,
- JEdit,
- JBuilder,
- IntelliJ IDEA,
- JCreator.

Každé z uvedených vývojových prostředí má své výhody i nevýhody. Prostředí NetBeans se věnuje detailněji další kapitola. Eclipse[3] je stejně jako většina výše vyjmenovaných prostředí multiplatformní a je vyvíjeno jako open-source produkt. Eclipse má rozsáhlou komunitu vývojářů a klade silný důraz na rozšiřovaní pomocí zásuvných modulů neboli pluginů. Oracle JDeveloper je profesionální software společnosti Oracle Corporation, který je možno bezplatně získat po registraci na oficiálních stránkách produktu[4]. JDeveloper lze kromě programování v jazyce Java využít také při návrhu webových aplikací, protože podporuje jazyky PHP, JavaScript, HTML, procedurální jazyk PL/SQL a XML. BlueJ je primárně navržen pro výukové účely. Od roku 2009 je plně pod licencí GNU GPL a lze ho bezplatně stáhnout z oficiálních stránek projektu[5]. JEdit[6] je stejně jako BlueJ uvolněn pod licencí GNU GPL. Jeho předností je vysoká možnost vlastní konfigurace, podporuje zvýrazňování syntaxe až pro 130 programovacích jazyků a umožňuje speciální nastavení pro různé jazyky. JBuilder[7] je původem produkt společnosti Borland Software Corporation a spolu s prostředím IntelliJ IDEA[8] patří mezi

[3] http://www.eclipse.org
[4] http://www.oracle.com/technetwork/developer-tools/jdev/downloads/index.html
[5] http://www.bluej.org
[6] http://www.jedit.org
[7] http://www.embarcadero.com/products/jbuilder
[8] http://www.jetbrains.com/idea/

velmi kvalitní, avšak komerční vývojová prostředí. JCreator[9] je komerční vývojové prostředí určené pro operační systémy Windows, je však méně výkonné než konkurenční placená vývojová prostředí a z tohoto hlediska se nehodí pro velké projekty.

Pro porovnání jednotlivých produktů je zde uvedena i přehledová tabulka s nejdůležitějšími údaji.

Tabulka 1 – Porovnání vývojových prostředí

Produkt	Dostupná verze*	Multiplatformní	Placený produkt	Napsáno v programovacím jazyce
BlueJ	3.0.4	Ano	Ne	Java
Eclipse	3.6.2	Ano	Ne	Java
IntelliJ IDEA	10.0.2	Ano	Ano	Java
JBuilder	2008 R2	Ano	Ano	Java
JCreator	5.00	Ne	Ano	C++
JEdit	4.4pre1	Ano	Ne	Java
NetBeans	6.9.1	Ano	Ne	Java
Oracle JDeveloper	11.1.1.4.0	Ano	Ne	Java

*Poslední dostupná verze v době psaní této práce

2.1.1 NetBeans IDE

Ve vývojovém prostředí NetBeans byla naprogramována praktická část této bakalářské práce, proto se na popis tohoto produktu zaměřím detailněji. Veškeré další ukázky související s vlastním programování, například ukázky kódů, byly napsány v tomto vývojovém prostředí.

Vývojové prostředí NetBeans má původ v českém projektu Xelfi. Ten byl jedním z prvních IDE pro tvorbu programů v Javě. Následně byl produkt, z hlediska dalšího vývoje, přejmenován na NetBeans a v roce 2000 o něj projevila zájem společnost Sun Microsystems, která následně produkt převzala. Vznikl tak nový Open Source projekt NetBeans sponzorovaný firmou Sun Microsystems, přičemž se NetBeans zároveň staly i hlavním vývojovým prostředím této firmy.

V současné době má NetBeans pověst vysoce kvalitního IDE především z těchto důvodů:

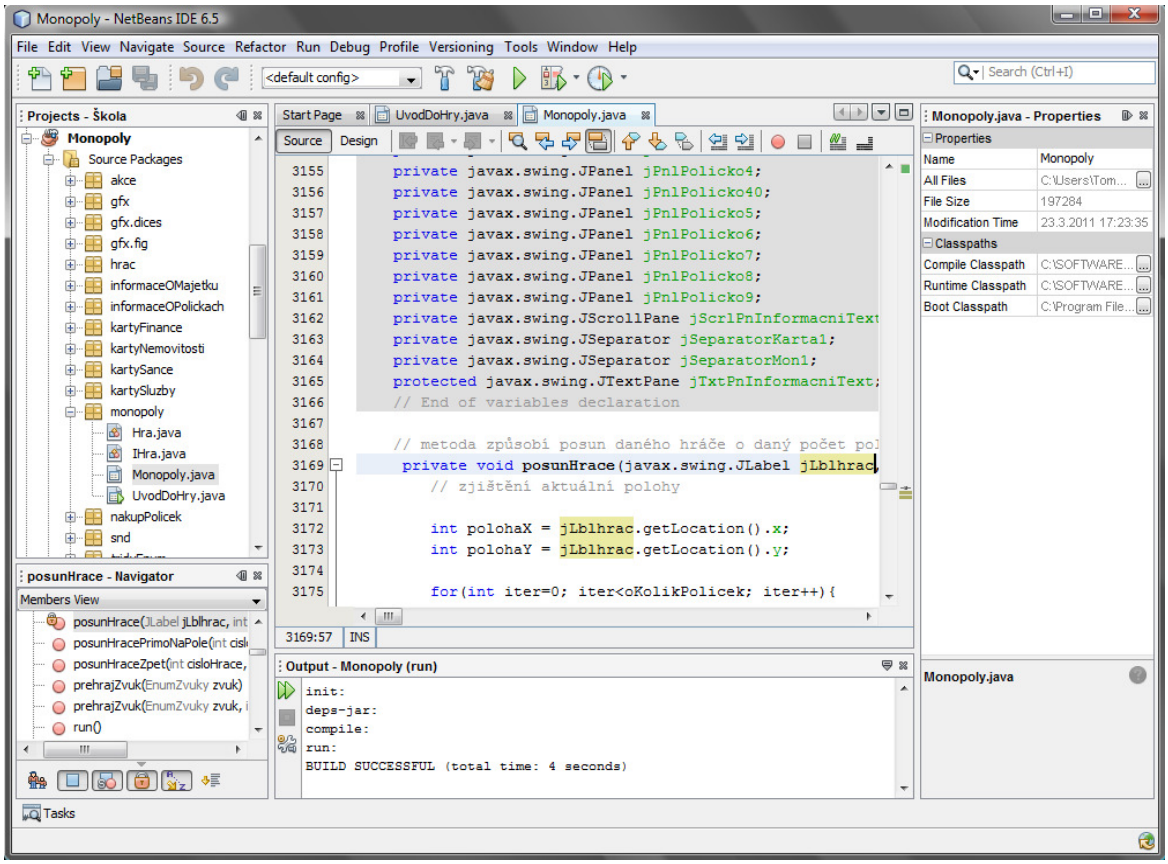
- Jedná se o bezplatný open-source produkt.
- Je nezávislý na operačním systému.
- Krom Javy podporuje také jazyky C, C++, HTML, Perl, Python.
- Má široké možnosti rozšíření pomocí Add-ons balíčků.
- Umožňuje vytvářet specializované aplikace pro mobilní zařízení (využití Java ME) a rozsáhlé informační systémy (využití Java EE).
- Podporuje tvorbu UML diagramů ze kterých lze následně vygenerovat zdrojový kód.
- Umožňuje pomocí GUI Builderu jednoduše vytvářet okenní aplikace.

[9] http://www.jcreator.com

- Nabízí usnadnění programování díky refaktoringu, automatickému generování kódu, inteligentnímu našeptávání či generovaní rozhraní z již hotového zdrojového kódu.
- Podpora Subversion pro týmové programování.

V současné době je k dispozici nejnovější verze NetBeans IDE 6.9.1, kterou lze získat ze stránek NetBeans[10]. Je také možné zvolit si některou z předdefinovaných variant balíčků. Lze tak například stáhnou variantu NetBeans IDE pouze pro programování PHP aplikací, nebo kompletní variantu se sadou nástrojů pro programování aplikací v různých jazycích.

Následující obrázek (Obrázek 1) ukazuje spuštěnou verzi NetBeans IDE 6.5.



Obrázek 1 – Vývojové prostředí NetBeans

2.2 Grafické uživatelské rozhraní

Pod pojmem grafické uživatelské rozhraní, označované anglickou zkratkou GUI, si v dnešní době můžeme představit prakticky jakoukoliv okenní počítačovou aplikaci. Grafické uživatelské rozhraní v podstatě definuje vzhled aplikace a nabízí možnosti jak aplikaci ovládat. To se většinou děje prostřednictvím ovládacího menu a grafických prvků,

[10] http://netbeans.org/downloads/index.html

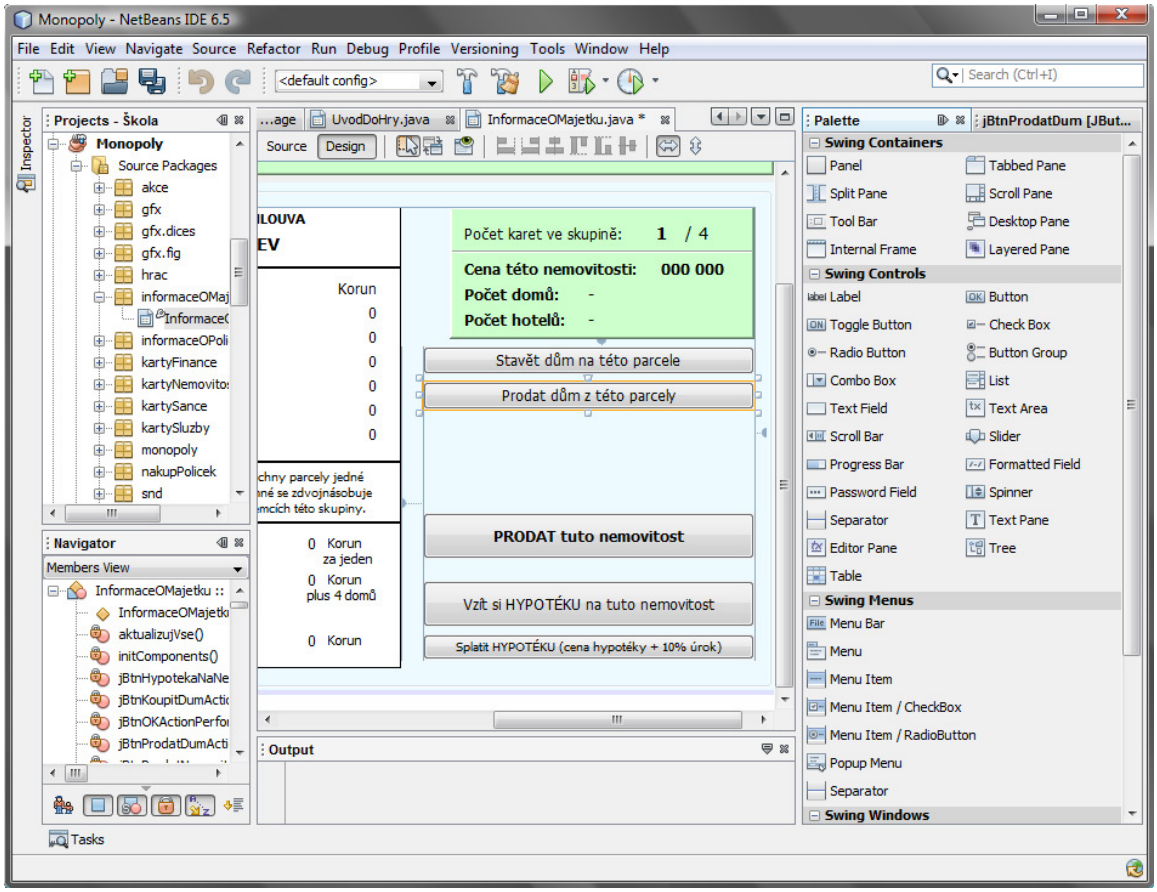
kterými jsou panely, tlačítka, posuvníky, textová pole, seznamy, rámečky a podobně. Pro ovládání se využívá především počítačová myš, mnohem méně pak klávesnice. Z historického hlediska GUI, alespoň tak jak ho známe dnes, vznikal od sedmdesátých let dvacátého století. Jedním z prvních počítačů, které využívaly grafické uživatelské rozhraní, byl stroj s názvem Alto firmy Xerox Palo Alto Research Center. Do té doby byl počítač nejčastěji ovládán pomocí dávkových programů.

Hlavní princip, kterým se grafické uživatelské rozhraní odlišuje od dávkových programů, spočívá v reakci na události. Událost je vyvolaná například kliknutím tlačítka myši na určitý prvek. Pokud má tento prvek zaregistrovaný posluchač události reagující na kliknutí myší, je následně spuštěn kód čekající na tuto událost. Programovací jazyk Java podporuje tvorbu okenních aplikací prakticky od svého vzniku. Princip reakce na události s využitím zaregistrování posluchače události je v Javě od JDK 1.1.

Jednou z velkých výhod jazyka Java je multiplatformnost. Toho lze samozřejmě využít i při psaní grafických aplikací. Program je pak možné bez jakýchkoliv úprav spustit pod libovolným operačním systémem a vzhled GUI bude vždy téměř shodný. Můžou se však vyskytnout drobné rozdíly ve vzhledu některých komponent. Ty jsou způsobeny tím, že operační systémy definují vlastní vzhled těchto komponent. Takže například zaškrtávací políčko, označované jako checkbox, bude mít trochu jinou podobu ve Windows než například v Linuxu. Nicméně rozvržení jednotlivých komponent a jejich funkce zůstanou zachovány beze změny[11].

Pro tvorbu GUI v Javě není potřeba žádných zvláštních nástrojů pro vytváření grafických aplikací. Vše lze naprogramovat „ručně“. Takové programování však bývá pracnější. Dnešní vývojová prostředí většinou nabízejí speciální prostředí pro jednoduchou, avšak efektivní tvorbu okenních aplikací. Stylem „táhni a pusť“ (anglicky označované jako drag-and-drop) lze pak jednotlivé komponenty přidávat do kontejneru, manipulovat s nimi, nastavovat jim různé parametry a podobně. Takovým příkladem může být NetBeans IDE GUI Builder, který je implementován v tomto vývojovém prostředí od verze 5.0. Ukázku novější verze GUI Builderu demonstruje následující obrázek (Obrázek 2).

[11] Od příchodu knihovny Swing je však možné pomocí metody `UIManager.setLookAndFell()` plně převzít kontrolu nad designem aplikace.



Obrázek 2 – GUI Builder ve výjovém prostředí NetBeans IDE 6.5

2.2.1 Paleta AWT a Swing

Java obsahuje dvě knihovny pro tvorbu grafického uživatelského rozhraní. Starší Abstract Window Toolkit (AWT), která je v jazyce Java obsažena od jeho vzniku, tedy od verze JDK 1.0. A novější Swing, které je k dispozici od JDK 1.2. Obě knihovny jsou součástí JFC.

Knihovna Swing má základ v knihovně AWT. Z tohoto důvodu se při programovaní často importují oba balíčky najednou. Některé vlastnosti dokonce knihovna Swing přebírá z ATW bez jakýchkoliv změn, anebo je pouze rozšiřuje o některé nové možnosti. Toto se týká například událostí, které jsou zavedeny přímo z AWT balíčku `java.awt.event`. Swing však oproti AWT přináší řadu vylepšení, je funkčně propracovanější, nabízí možnost využít dvojitou vyrovnávací paměť (double buffering), obsahuje více grafických prvků a rozšiřuje také možnosti správce rozvržení. Rozdíl ve velikostech knihoven je znázorněn v následující tabulce.

Tabulka 2 – Porovnání knihovny AWT a Swing

	Knihovna AWT	Knihovna Swing
Základní balíček pro import	java.awt	javax.swing
Počet tříd v základním balíčku	91	108
Počet rozhraní v základním balíčku	16	21

Počet dalších balíčků (podbalíčků)	10	9
Počet tříd v podbalíčcích	147	312
Počet rozhraní v podbalíčcích	84	55

Počítáno z verze Javy JDK 1.6.0_12 ve vývojovém prostředí NetBeans 6.5.

2.2.2 Komponenty pro GUI

Komponenty jsou základní grafické prvky uživatelského prostředí. Definují vzhled aplikace a lze jimi aplikaci ovládat. Každá komponenta, která je obsažena v knihovně Swing, je potomkem třídy `JComponent` a její název začíná na „J“. Díky tomu je na první pohled patrné, že komponenta není z knihovny AWT.

Komponenty zde popsané jsou obsažené v balíčku `javax.swing`. Bezparametrický konstruktor prvku třídy má pak podobu *`JNazevKomponenty`*`()`. Zde je přehled nejdůležitějších a nejčastěji používaných komponent.

Button – patří mezi velmi často využívané komponenty pro ovládání aplikace. Jedná se o tlačítko, jehož základní funkcí je reakce na kliknutí myší.

ComboBox – komponenta, která v sobě kombinuje tlačítko a rolovací seznam s položkami. Každá položka je jednoznačně určena indexem (začínajícím od nuly). Položky seznamu nemusí být nastaveny napevno, a lze je tedy v běhu programu i přidávat a odebírat. Komponentu lze nastavit i pro editaci záznamu.

CheckBox – zaškrtávací políčko. Nejdůležitější je vlastnost `selected`, určující, zda je políčko zaškrtnuté, či nikoliv. Používají se metody `isSelected()` pro zjištění stavu a `setSelected(boolean)` pro nastavení stavu. ChcekBoxy v aplikaci jsou na sobě navzájem nezávislé. Lze mít zaškrtnuté najednou všechny, žádné, nebo jen některé.

Frame – jedna z nejdůležitějších kontejnerových komponent. Pro aplikaci je to výchozí komponenta. Slouží pro umístění ostatních komponent. Obsahuje základní ovládací prvky, titulek, ikonu a orámování. Lze též nastavit akci pro kliknutí na uzavření okna. Frame pak může celou aplikaci ukončit, nebo se pouze skryje. Lze také nastavit, aby se na uzavření okna nereagovalo.

Label – komponenta nejčastěji užívaná pro zobrazení prostého, ale i formátovaného textu. Zároveň lze tuto komponentu použít pro zobrazení obrázku pomocí metody `setIcon(Icon)`. Pokud nebude u této komponenty zadán žádný text, tato komponenta se nezobrazí. Pro nastavení textu se používá metoda `setText(String)` a pro přečtení `getText()`.

List – komponenta se seznamem položek. Na rozdíl od ComboBox ji lze zvětšit tak, aby zobrazovala všechny položky v ní uvedené. V případě, že položek v Listu je více, než je možné zobrazit, pak List obsahuje podle potřeby posuvník, a to buď svislý, vodorovný, případně oba.

MenuBar – výchozí komponenta, která slouží jako kontejner pro přidávání dalších prvků menu.

MenuItem – položka prvku v menu. Existují i položky se specifickým chováním. CheckBoxMenuItem má stejné vlastnosti jako klasické zaškrtávací políčko s tím, že je umístěno v menu. RadioButtonMenuItem slouží jako přepínač umístěn v menu.

Panel – slouží jako kontejner pro přidávání dalších komponent. Panelů může být v aplikaci více, mohou být ve vrstvách, orámované a lze jimi program zpřehlednit tím, že jednotlivé komponenty jsou pak logicky členěny tak, že každá komponenta patří na nějaký panel. Dále lze využít nastavení rozvržení pro každý panel samostatně. Panel také patří mezi prvky, na které lze kreslit.

PasswordField – komponenta určená pro zadání textu, který je při zadávání zobrazován jako tečky (nebo hvězdičky). Praktické využití je při zadávání hesel.

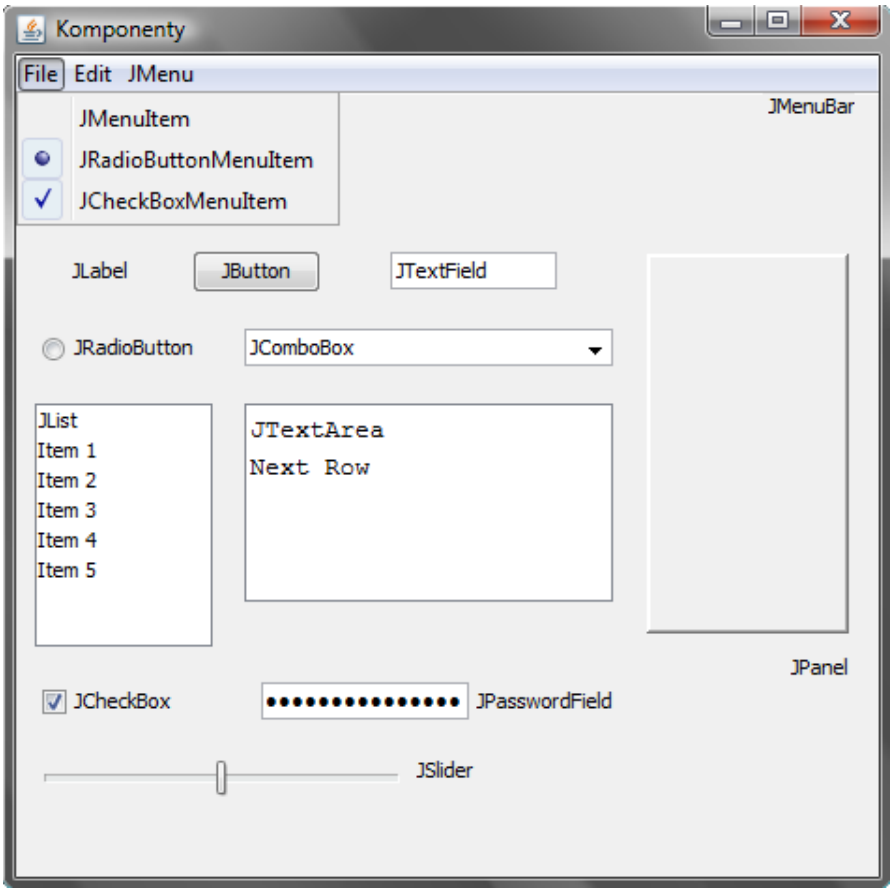
RadioButton – Přepínač. Těchto komponent bývá v aplikaci obvykle několik a slouží ke zvolení jedné varianty z několika nabízených. Každý RadioButton musí být v některé skupině ButtonGroup. V rámci této skupiny lze vždy zaškrtnou pouze jeden RadioButton. Skupin RadioButtonů může být v aplikaci více.

Slider – komponenta nejčastěji využívaná pro jednoduché zvolení hodnoty v daném rozmezí.

TextArea – komponenta slouží pro zadání rozsáhlejšího textu na více řádků. Lze ji definovat velikost pomocí počtu sloupců a řádků, které má obsahovat. Je-li zadaný text delší než uvedený rozsah, pak jsou uvnitř komponenty zobrazeny svislé nebo vodorovné posuvníky.

TextField – komponenta slouží pro zadání textu omezeného na jeden řádek.

Výchozí grafickou podobu komponent znázorňuje následující obrázek (Obrázek 3).



Obrázek 3 - Ukázka grafických komponent

2.2.3 Vlastnosti komponent

Každá komponenta má k dispozici vlastnosti, které komponentu charakterizují. Jelikož jsou komponenty potomky třídy `JComponent`, lze využít řadu metod této třídy pro nastavení, či naopak zjištění hodnoty vlastnosti dané komponenty. Některé metody jsou k dispozici pro každý objekt, který je potomkem třídy `JComponent`, některé metody jsou specifické jen pro určitý druh komponent. Příkladem může být určení pozice komponenty. Metoda pro zjištění pozice `getLocation()` je přístupná u každého prvku typu komponenta. Naopak metoda `setText(String)`, která je typická především pro komponentu Label nebo Button, není k dispozici například u komponenty Panel nebo List. Obdobně metody `setSelected(boolean)` a `getSelected()` jsou k dispozici u prvků typu RadioButton nebo Checkbox, ale například u komponenty PasswordField je nenajdeme. Je to logické chování, neboť komponenta, která neplní funkci zaškrtávacího políčka, nemůže mít metodu na zjištění, zda je tato komponenta zaškrtnuta či nikoliv.

Vlastností komponent a metod, které s nimi nějakým způsobem manipulují, je celá řada. Lze použít metody pro zviditelnění prvku, obarvování komponenty, nastavování různých rámečků či ohraničení a podobně. Například komponenta Panel má v základní nabídce přes dvě stovky metod. Navíc u některých metod je možné zadávat různé parametry. Takovým příkladem u komponenty Panel je metoda pro nastavení pozice, která

může být zadána buď s parametrem typu bod volaná zadáním `setLocation(Point)`, nebo lze zavolat tutéž metodu s parametry určujícími přímo polohu x a y v podobě `setLocation(int, int)`.

Další zajímavou vlastností především u komponenty typu Label, obecně pak u komponent s metodou `setText(String)`, je nastavení formátování podle specifikací HTML. Tuto možnost umožňuje knihovna Swing. Pro použití této vlastnosti se do řetězce pro text komponenty zadá jako první značka `<html>` a následně podle dalších vlastností jazyka HTML lze formátovat zadaný text, nastavovat barvy, nadpisy nebo odstavce.

2.2.4 Práce s kontejnery

Při tvorbě návrhu hraje důležitou roli také samotné rozmístění jednotlivých komponent. Ty jsou umísťovány do kontejnerů (kontejnerových komponent) a nějakým způsobem vůči tomuto kontejneru rozmístěny. O tuto záležitost se stará takzvaný správce rozvržení (anglicky layout manager). Základních správců rozvržení existuje osm a lze použít různé správce pro různé kontejnery. Zároveň má i volba správce rozvržení vliv na některé vlastnosti jednotlivých komponent, především na jejich rozměry.

Výhoda jednotlivých správců rozvržení spočívá v tom, že pro kontejnerové komponenty existuje vždy implicitní správce rozvržení a není potřeba ho pak vytvářet pro každý kontejner zvlášť. Samozřejmě lze pro každý kontejner nastavit libovolného správce rozvržení. Každý správce má určité chování pro něj typické. Komponenty pak podléhají tomuto chování a jsou rozmísťovány dle daných pravidel.

Existující správci rozvržení jsou:

- Absolute Layout,
- Border Layout,
- Box Layout,
- Card Layout,
- Flow Layout,
- Grid Bag Layout,
- Grid Layout,
- Null Layout.

3 Grafika v Javě

3.1 Java 2D

Java 2D API obsahuje třídy, které se využívají při práci s pokročilou počítačovou grafikou, při kreslení geometrických tvarů nebo při práci s obrázky.

3.2 Třída Graphics2D

Tato třída je vlastně potomek třídy `Graphics` obsažené v knihovně AWT. Zajišťuje veškeré operace pro práci s grafikou. Přináší však také celou řadu vylepšení a odstraňuje některé předchozí nedostatky. Stejně jako třída `Graphics` je i `Graphics2D` součástí balíčku AWT.

Přednosti třídy Graphics2D:

- Sofistikovanější nástroje pro práci s geometrickými útvary.
- Kreslení čar tloušťky 1 pixel a větší.
- Využití množinových operací pro tvorbu geometrických tvarů.
- Poskytuje možnosti pro otáčení kreslených objektů.
- Umožňuje upravovat a manipulovat se souřadnicovým systémem.
- Kvalitní správa barev a formátování textu.
- Vyplňování nakreslených objektů barvou či texturou.
- Přináší nástroje pro jednodušší práci s obrázky.
- Lze kreslit téměř na všechny objekty, jež jsou potomky třídy `JComponent`.

Protože je třída `Graphics2D` potomkem třídy `Graphics`, před použitím je nutné v metodě `paint()` provést přetypování například následujícím způsobem:

```
public void paint(Graphics g){
      Graphics2D g2d = (Graphics2D) g;
}
```

Po přetypování lze již u objektu `g2d` volat metody třídy `Graphics2D`. Samozřejmě je také nutné importovat třídy z balíčku awt:

```
import java.awt.Graphics;
import java.awt.Graphics2D;
```

3.3 Práce s obrázky

Obrázky jsou nedílnou součástí grafických aplikací. Často je výhodnější začlenění klasického obrázku do programu než jeho vytváření pomocí programovacích metod. S obrázky je možné pak jednoduše pracovat. Lze je zvětšovat, zmenšovat, zviditelnit je nebo je naopak skrýt, rozmazávat, otáčet či s nimi pohybovat. Nejčastěji používané formáty obrázků používaných v aplikacích jsou GIF, JPEG a PNG.

Pro práci s obrázky lze využít různé třídy z balíčku AWT, ale i Swing. V balíčku `java.awt` je pravděpodobně nejpoužívanější třída `Image`. Další užitečné třídy a rozhraní, například pro použití filtrů, se nacházejí v balíčku `java.awt.image`. Swing zajišťuje práci i s obrázkovými ikonami používaných u grafických komponent. Nejpoužívanější třída je pak `ImageIcon` z balíčku `javax.swing`. Třídu `ImageIcon` lze, navzdory trochu matoucímu názvu, použít samozřejmě i pro práci s klasickými obrázky. Navíc lze také využít třídy z balíčku `javax.imageio` především pro rychlejší načítání obrázku.

V současné době je kladen velký důraz na rychlost běhu aplikací. Při práci s obrázky můžeme narazit na problém, kdy například čekání na načtení velkého obrázku může způsobit zvýšení latence programu. Z tohoto důvodu se při načítání obrázku využívá vyrovnávací paměť (buffer) a speciální třídy pro bufferované načítání obrázku. V této souvislosti se využívá nejčastěji třída `BufferedImage` a `ImageIO`.

V programu pak může být načtení obrázku realizováno mnoha způsoby. Jeden z možných způsobů, kdy je zároveň zachycena výjimka při načítání, je tento:

```
BufferedImage image;
try{
      image = ImageIO.read(getClass().getResource(„obrazek.png“));
}
catch(IOException ex){
      // kód pro ošetření výjimky
}
```

Kompletní ukázka načtení a zobrazení obrázku v GUI aplikaci s využitím třídy `Graphics2D` je uvedena v příloze.

Jak již bylo řečeno, obrázky jsou i součástí grafických komponent. Zobrazení obrázku v komponentě *`mujLabel`* typu `JLabel` lze pak realizovat následujícím způsobem:

```
ImageIcon ikona;
ikona = new ImageIcon(getClass().getResource(„ikona.gif“));
mujLabel.setIcon(ikona);
```

3.4 Animace a efekty

Výsadou Javy 2D je využití množství nástrojů pro aplikaci animačních a obrázkových efektů. Příkladem takového efektu může být zmizení obrázku, jeho rozmazání nebo plynulý posun na nové souřadnice. U některých formátů obrázků lze využít také fakt, že jedna barva může být nastavena jako průhledná.

Efekty jsou aplikovány na obrázky třídy `BufferedImage` přičemž nejčastěji používané jsou tyto efekty:

- Konvoluce – rozmazání obrázku, zaostření, detekce hran.
- Afinní transformace – změna rozměrů, otáčení.

- Změna barev – vytvoření negativu, zesvětlení.

Uvažujme například situaci, kdy je potřeba daný obrázek zobrazit rozmazaně na plochu kreslicího plátna. Pro tyto účely existuje třída `ConvolveOp`. Před samotným použitím této třídy je však potřeba vyřešit následující věci:

- Zajištění načtení obrázku.
- Vytvoření matice (označuje se jako *jádro*) pro rozmazání.
- Správné zobrazení upraveného obrázku s využitím metody `drawImage(BufferedImage, BufferedImageOp, int, int)` ze třídy `Graphics2D`.

Výsledná část kódu pak může mít například tuto podobu:

```
Graphics2D g2d;
      // načtení obrázku
BufferedImage image = null;
try{
      image = ImageIO.read(getClass().getResource(„obrazek.png“));
}
catch(IOException ex){
      // kód pro ošetření výjimky
}
      // vytvoření matice
float prvekMatice = 1.0f / 9.0f;
float matice[] = {
      prvekMatice, prvekMatice, prvekMatice,
      prvekMatice, prvekMatice, prvekMatice,
      prvekMatice, prvekMatice, prvekMatice
};
Kernel jadroEfektu = new Kernel(3, 3, matice);
      // vytvoření a použití třídy ConvolveOp
ConvolveOp efektRozmazani;
efektRozmazani = new ConvolveOp(jadroEfektu, ConvolveOp.EDGE_ZERO_FILL,
null);
      // zobrazení výsledného obrázku
g2d.drawImage(obrazek, efektRozmazani, 0, 0);
```

Použití efektu je ukázáno na následujícím obrázku (Obrázek 4). Obrázek vlevo je v originální nezměněné podobě. Na pravý obrázek byl použit efekt rozmazání zprůměrováním. Pro dosažení tohoto efektu byl použit kód uvedený výše.

Obrázek 4 - Ukázka použití efektu rozmazaní

Animace v aplikacích lze vytvořit například překrýváním jednotlivých obrázků ze skupiny obrázků. Efektní je také zobrazit obrázek na krátký okamžik na určitých souřadicích, poté mu nastavit nové souřadnice a celé kreslící plátno znovu překlestit, a zobrazit tak obrázek na nové pozici. Využít lze také obrázky, které samy o sobě tvoří animaci. Nejčastěji se toto realizuje pomocí animovaného obrázku typu GIF.

3.5 Kreslení

Třída `Grahics2D` obsahuje řadu metod pro kreslení grafických objektů. Jejich skládáním lze pak vytvářet složitější tvary či obrázky. Jednotlivé grafické objekty lze kreslit dvěma způsoby. První možností je vykreslit nastavenou barvou pouze obrys objektu. Název metody má tvar `draw`*`NazevTvaru`*. Druhá možnost spočívá ve vyplnění plochy objektu nastavenou barvou. Název má tvar `fill`*`NazevTvaru`*. Je možné využít postupně obě možnosti, a nakreslit tak například tvar vyplněný jednou barvou s obrysem druhé barvy.

Jednotlivé metody pro nakreslení obrysu tvaru jsou:

- draw3DRect,
- drawArc,
- drawLine,
- drawOval,
- drawPolygon,
- drawPolyline,
- drawRect,
- drawRoundRect.

Metody pro výplň tvaru jsou následující:

- fill3DRect,

- fillArc,
- fillOval,
- fillPolygon,
- fillRect,
- fillRoundRect.

Krom výše uvedených metod existují i metody `drawShape(Shape)` a `fillShape(Shape)`, které přebírají jako parametr instanci třídy `Shape`. Kreslení objektů není nikterak složité. Následující kód vložený do překryté metody `paint(Graphics)` by způsobil nakreslení červeného čtverce bez výplně se zaoblenými rohy.

```
Graphics2D g2d;
g2d.setColor(Color.RED);
g2d.drawRoundRect(50,50,200,200,100,100);
```

3.6 Vlákna

Pro běh více částí aplikace současně se využívá vláken. Ve Swingu jsou všechna volání a reakce na události součástí jednoho vlákna. Tento fakt se tedy týká i zobrazování grafických objektů, volání metody `reapint()` nebo `update()` a podobně. Vlákna lze vytvářet ručně za pomoci třídy `Thread`. Takto je možné vytvořit vlákno, které zobrazuje například animaci nějakého objektu nezávisle na další manipulaci s aplikací. Případně lze využít vlákna pro přehrání zvukového souboru a podobně.

Po vytvoření instance ze třídy `Thread` lze zavolat metodu `start()`, která vlákno spustí. Při realizaci se také využívá rozhraní `Runnable`. To obsahuje jedinou metodu `run()` obsahující kód, který se má provést ve vlákně. Jedna z výhod vláken spočívá ve využití metody `sleep(int)` třídy `Thread`, která dané vlákno „uspí“ na určitou dobu, jež je zadána jako parametr metody. Vhodným použitím metody `sleep(int)` a změnou parametrů metody `setLocation(int, int)` u komponent lze jednoduše dynamicky pohybovat s grafickými prvky typu JComponent.

Následující kód demonstruje jeden ze způsobů vytvoření nového vlákna.

```
Thread noveVlakno = new Thread() {
      public void run(){
            // kód spuštěný po zavolání metody start()
      }
};
noveVlakno.start();
```

3.7 Zvuky a zvukové efekty

Od verze Javy 1.2 je možné v aplikacích přehrávat hudební soubory. Do té doby bylo možné přehrát zvuk pouze v apletech. Původně se používala pouze metoda `play()` třídy `Applet`. Tento způsob však přináší řadu nepříjemností. Pokud je potřeba přehrát

stejný zvukový soubor znovu, je nutné ho znovu celý načíst. Samotné načítání také nevyvolá žádnou výjimku, a to ani v případě, že data nebylo vůbec možné načíst. Při přehrávání zvuku se tedy nově využívá třída `AudioClip`. Tato třída odstraňuje předchozí problémy a navíc dovoluje využít metody `loop()` a `stop()` pro opakované přehrávání zvukového souboru a pro jeho zastavení. Lze také přehrávat větší množství hudebních formátů než dříve. Výhoda také spočívá v možnosti přehrávat více zvukových souborů najednou tak, že vytvoříme více instancí třídy `AudioClip`. Každá taková instance může pak přehrát zvuk nezávisle na ostatních. Nicméně i přes veškeré výhody má i třída `AudioClip` několik nedostatků. Jedním z nich je nedostatek informací o právě přehrávaném zvuku. Nelze proto například zjistit, zda přehrávání daného zvuku již skončilo či nikoliv.

Následující příklad kódu demonstruje situaci, kdy jsou načteny dva různé zvukové efekty ze dvou různých zdrojů. Oba jsou následně současně jedenkrát přehrány pomocí metody `play()`.

```
try{
      AudioClip zvuk1, zvuk2;
      URL url1 = new URL(„file:./efekty/efect.waw“);
      URL url2 = new URL(„file:./zvuky/whistle.waw“);
      url1.openConnection().getInputStream().close();
      zvuk1 = Applet.newAudioClip(url1);
      zvuk1.play();
      url2.openConnection().getInputStream().close();
      zvuk2 = Applet.newAudioClip(url2);
      zvuk2.play();
}catch(Exception ex){
      // kód pro ošetření výjimky
}
```

3.8 Čeština v GUI

Zobrazení češtiny v GUI aplikacích z mého pohledu není takový problém, jak by se na první pohled mohlo zdát. Háčky a čárky nad písmeny jsou zobrazeny normálně. Stejně tak i přehlasovaná písmena, speciální znaky a podobně. Java totiž používá kódování Unicode, které zahrnuje i českou abecedu. Z tohoto důvodu není problém nastavit tlačítku název třeba „Příliš žluťoučký kůň“. Problém by mohl nastat v případě načítání externího zdroje s českým pojmenováním. V předchozí kapitole (3.7) se načítal soubor s názvem „efect.waw“. Naštěstí zde ale nenastává problém ani v případě, že se zdrojový zvukový efekt jmenoval „Příliš žluťoučký kůň.waw“.

3.9 Java 3D

Kromě nástrojů pro programování 2D grafiky, kreslení a práce s multimédii poskytuje Java také rozsáhlé nástroje pro tvorbu a zobrazení 3D grafiky. Java 3D API obsahuje velké množství nástrojů pro práci s takzvaným „grafem scény“ (případně „3D scény“ nebo pouze „scény“), tvorbu trojrozměrných geometrických tvarů, detekce kolizí, rozšířené možnosti ozvučení scény, nástroje pro práci s osvětlením, tvorbu 3D animací

a podobně. Na úvod je nutné poznamenat, že balíček Java 3D nebývá standardní součástí Javy a je potřeba ho dodatečně stáhnout[12]. Prakticky existují dvě verze knihovny. První využívá rozhraní OpenGL, druhá využívá DirectX Graphics. Obě varianty jsou však již delší dobu obsaženy v jednom instalačním balíčku. Není proto třeba při stahování balíčku specifikovat, kterou variantu požadujeme, jako tomu bylo do verze 1.3.2. OpenGL nebo DirectX se pak stará o samotné zpracování a vykreslování grafických dat.

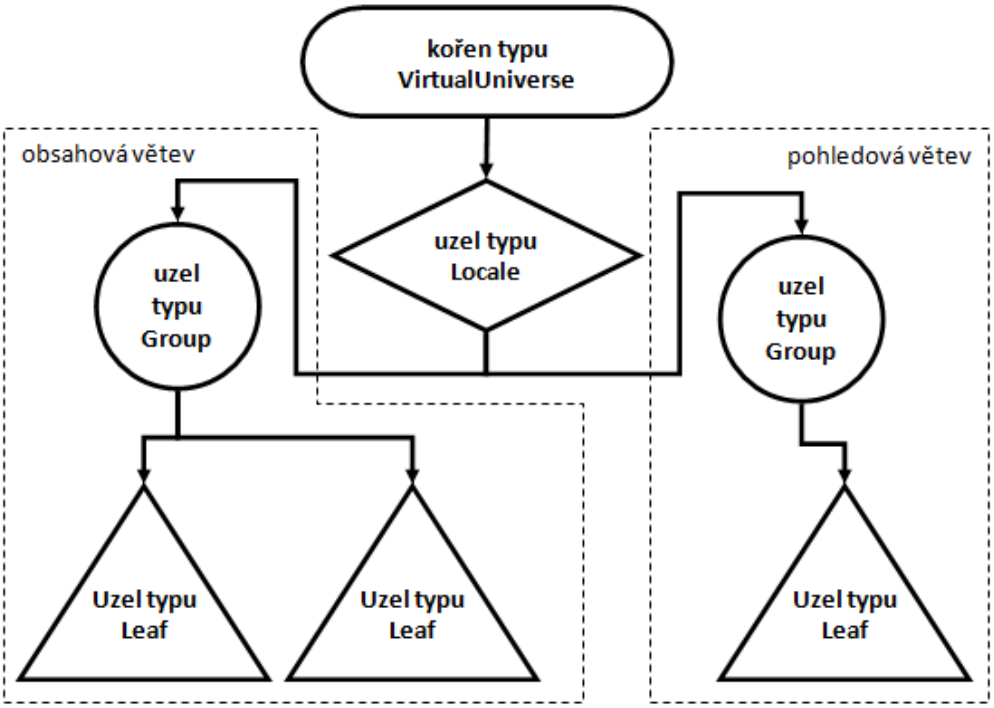
Java 3D má širokou podporu a existuje také velké množství návodů jak pracovat s trojrozměrnou grafikou v Javě. Jedním z webů, které jsou věnovány 3D Javě, je http://java3d.java.net/, ze kterého je možné také stáhnout nejnovější verzi Javy 3D. V době psaní této práce byla nejnovější dostupná verze 1.5.2. Množství rozsáhlých návodů lze pak nalézt na stránkách http://java.sun.com/developer/onlineTraining/java3d/index.html.

Jak již bylo řečeno v úvodu, Java 3D používá pro organizaci graf scény. Jeho logické uspořádání má podobu stromové strukturu. Kořenem této struktury je objekt třídy `VirtualUniverse` reprezentující nějaký virtuální prostor. Poté následuje uzel typu `Locale` reprezentující polohu grafu scény v daném virtuálním prostoru. Jednotlivé další uzly struktury se pak dělí na dvě skupiny. Objekty typu `Group`, které mohou obsahovat další potomky, a objekty typu `Leaf` tvořící listy struktury, což mohou být již konkrétní grafické objekty, osvětlení nebo zvuky. Výchozí třída pro definici tvaru je `Shape3D`. Třídy `Geometry` a `Appearance` jsou určeny pro geometrii a vzhled tvaru.

Ukázku jednoduchého grafu scény zachycuje následující obrázek (Obrázek 5). Graf scény tvoří kořen typu `VirtualUniverse`, uzel typu `Locale` a z něj vycházející dvě hlavní větve. Levá, které se označuje jako *obsahová*, a pravá, která se označuje jako *pohledová*. Obsahová větev nese informace o jednotlivých grafických objektech, tedy údaje o jejich geometrii a tvaru. V tomto případě se nejedná o funkční řešení, nicméně u reálného řešení by trojúhelníky v obsahové části představovaly již konkrétní grafické objekty s definovaným tvarem a vlastnostmi. V pohledové části by trojúhelníky obsahovaly informace pro pozorovatele. Trojúhelník by pak například reprezentoval třídu `ViewPlatform`, obsahující informace o tom, kde se pozorovatel nachází a co v tomto případě vidí.

Obecně se dá tvrdit, že se pohledové části v aplikacích od sebe nijak zásadně neliší, naproti tomu obsahové části se odlišují hodně.

[12] http://www.oracle.com/technetwork/java/javase/tech/index-jsp-138252.html



Obrázek 5 – Jednoduchý graf scény

Přestože má knihovna Java 3D značnou podporu nejen ze strany společnosti Sun Microsystem, existuje i množství alternativ, jak pracovat s trojrozměrnou grafikou v Javě bez použití knihovny Java 3D.

Příkladem takové alternativy může být například:

- Aviatrix
- free-D2
- GL4Java
- JiD
- jME
- JOGL
- LWJGL
- OpenMind
- Xith3D

4 Deskové hry

4.1 O deskových hrách

Deskové hry jsou původem z Německa. V dnešní době jsou však již rozšířeny v celém světě a zájem o ně je stále vysoký. Moderní deskové hry mají většinou složitější pravidla a obsahují řadu různorodých herních mechanizmů. Některé herní postupy se dokonce snaží co nejvíce se přiblížit realitě. Často je v deskových hrách zastoupen prvek náhody, ovšem novější hry se snaží tento prvek spíš potlačit a naopak kladou velký důraz na rozhodování hráče a na jeho strategické schopnosti. Moderní deskové hry mají také často více či méně složitý příběh, který doplňuje zážitek z hraní.

4.2 Deskové hry v počítačové podobě

Deskových her převedených do počítačové podoby existuje nepřeberné množství. V dnešní době se jedná především o realizace v podobě on-line her. Dokonce existuje i on-line hra Monopoly, ale její provedení je velmi zastaralé a hru samotnou se mi nepodařilo vůbec spustit.

Mému projektu je nejblíže realizace podobné klasické deskové hry Dostihy a sázky. Hru s názvem Dostihy 3000 Deluxe, kterou je možné hrát ve více lidech na jednom počítači, vytvořil Martin Mikloš. Hra samotná je určena pro dva až čtyři hráče, možností je i hra jednoho hráče proti počítači. Vytvořena byla v prostředí Borland Delphi.

Obrázek 6 - Ukázka konkurenční hry Dostihy a sázky

5 Hra Monopoly v Javě

Hra je napsána v programovacím jazyce Java, vytvořena byla ve vývojovém prostředí NetBeans IDE 6.5. Je určena pro dva hráče, kteří hrají na jednom počítači. Princip hry je tahový, kdy se jednotliví hráči střídají v hraní.

5.1 O hře monopoly

Hra Monopoly je původně klasická desková hra pro obchodování s nemovitostmi. Je to jedna z nejznámějších a nejrozšířenějších deskových her na světě. Existuje také množství specifických variací této hry.

Princip hry je založen na schopnostech hráčů, nicméně velkou roli hraje také náhoda. Hráči hází kostkami a pohybují svými figurkami po čtvercové hrací desce. Jednotlivá políčka na hracím plánu představují většinou ulice, které jsou rozděleny do barevných skupin a které se dají koupit, ale existují i speciální políčka, která mají pro hru zvláštní význam. Smyslem hry je vlastnit co nejvíce majetku, chytře obchodovat a nezbankrotovat. Hráči tedy nakupují ulice a v případě, že vlastní všechny ulice jedné barvy, mohou na nich stavět domy nebo hotely. Hráči, kteří se zastaví na políčkách vlastněných soupeřem, pak platí poplatky vlastníkovi. Poplatky jsou tím vyšší, čím více domů nebo hotelů v dané ulici majitel postavil. Hra je navíc obohacena o políčka, na kterých rozhoduje náhoda. Tato políčka jsou označena jako „Šance“ a „Pokladna“ a hráčům, kteří se na nich zastaví, mohou například nařídit jít do vězení, či zaplatit poplatky za vlastněné nemovitosti nebo posunout svou figurku na jiné pole ve hře.

5.2 Implementace hry Monopoly

5.2.1 Balíčková struktura

Celý projekt hry Monopoly se skládá celkem z 16 balíčků. Ty lze rozdělit do tří částí: balíčky s grafikou, balíčky se zvukem a balíčky s třídami tvořící strukturu celé hry.

Balíčky s grafikou obsahují veškeré obrázky potřebné pro hru. Většinou se jedná o obrázky typu PNG případně GIF. Jejich výhodou je totiž možnost nastavit jednu barvu jako průhlednou, čehož bylo téměř vždy využito. Jednotlivé balíčky jsou tedy tyto:

- `gfx`,
- `gfx.dices`,
- `gfx.fig`.

Prakticky největší množství obrázků obsahuje balíček `gfx`. Jedná se o obrázky domečků, ikon, některých políček (například pole „Jdi do vězení“), obrázky průvodce a podobně. Balíčky `gfx.dices` a `gfx.fig` jsou spíše pro přehlednost a obsahují obrázky týkající se pouze kostek a herních figurek.

Balíček zajišťující ozvučení hry je pouze jeden a má název `snd`. Obsahuje v sobě jednotlivé zvukové efekty, ale zároveň i třídy, které se starají o ozvučení aplikace.

Zbývající balíčky obsahují třídy a rozhraní k těmto třídám a starající se o jednotlivé části hry. Celkem je těchto balíčků dvanáct a z jejich názvu je částečně zřejmé, jakou část hry obstarávají:

- `akce`,
- `hrac`,
- `informaceOMajetku`,
- `informaceOPolickach`,
- `kartyFinance`,
- `kartyNemovitosti`,
- `kartySance`,
- `kartySluzby`,
- `monopoly`,
- `nakupPolicek`,
- `navod`,
- `tridyEnum`.

Součástí projektu je ještě jeden balíček s názvem `zTest`. Ten byl však využit pouze pro testovací účely a na samotnou hru nemá žádný vliv.

5.2.2 Grafická podoba

Jedním z hlavních cílů bylo přenést do počítačové podoby nejenom hru samotnou, ale i její atmosféru. Ta je vytvořena z části i díky grafickému provedení hry a některým detailům. Typickým příkladem pro deskovou variantu hry jsou kovové herní figurky, barevné domečky, pan Monopol (v angličtině „Mr. Monopoly“ nebo také „Rich Uncle Pennybags“), který je maskotem hry a podobně. Z tohoto důvodu jsem se snažil zachovat co možná největší věrnost předloze.

Hrací plán má tvar čtverce, po jehož okraji jsou jednotlivá políčka, po kterých se hráčova figurka pohybuje. Samotný hrací plán, barvy políček a celkové provedení, dle mého názoru, věrně kopírují vzhled originálu.

Výslednou grafickou podobu celé aplikace zobrazují následující obrázky (Obrázek 7 a Obrázek 8). První obrázek zachycuje úvodní obrazovku. Na druhém obrázku je ukázka již rozehrané hry. Herní figurky mají tvar klobouku a psa. Na červených a tyrkysových políčkách jsou vidět již postavené domy.

Obrázek 7 - Úvodní obrazovka hry Monopoly

Obrázek 8 - Ukázka hry Monopoly

5.2.3 Vytvoření hry

Hlavní třídou tvořící zároveň vstupně výstupní rozhraní je třída `Monopoly` z balíčku monopoly. Při spuštění hry je však nejdříve volána třída `UvodDoHry`. Ta tvoří spustitelnou třídu, která obdrží základní informace od hráčů. Před spuštěním hry je kontrolována vyplněnost polí. Pokud hráč ponechá pole se jménem nezměněné (výchozí nastavení je „hráč 1“ a „hráč 2“), je hráč na tuto skutečnost upozorněn, ale může s tímto jménem pokračovat. Po spuštění hry není možné změnit jména ani zvolené obrázky hráčů.

Pokud je vše v pořádku, je možné spustit hru kliknutím na tlačítko „SPUSŤ HRU“. Následně je vytvořena instance `novaHraMonopoly` třídy `Monopoly`. Použit je parametrický konstruktor, který obdrží údaje o volbách hráčů. Aktuální okno je skryto a okno nové hry je naopak zobrazeno pomocí metod `setVisible(boolean)`. Okno je zarovnáno na střed obrazovky, pět pixelů od horního okraje obrazovky.

Třída `Monopoly` obsahuje několik privátních atributů, které poskytují především informace o aktuálním stavu hry. Zároveň jsou vytvořeny instance tříd nutných pro hru.

Nejdůležitější je pravděpodobně instance třídy `Hra`, pomocí níž lze přistoupit k informacím o jednotlivých hráčích. Tyto informace uchovává třída `Hrac`, jejíž instance jsou vytvořeny a uloženy do `poleHracu[]` ve třídě `Hra`.

Dále jsou vytvořeny instance tříd jednotlivých herních balíčků.

```
IBalicekKaretNemovitosti balicekNemovitosti = new
BalicekKaretNemovitosti();
IBalicekKaretFinance balicekFinanci = new BalicekKaretFinance();
IBalicekKaretSance balicekSance = new BalicekKaretSance();
IBalicekKaretCeskeDrahy balicekCeskeDrahy = new BalicekKaretCeskeDrahy();
IBalicekKaretVerejneSluzby balicekSluzby = new
BalicekKaretVerejneSluzby();
```

Konstruktor třídy `Monopoly` inicializuje veškeré grafické komponenty a zároveň vytvoří herní balíčky karet z výše uvedených instancí tříd.

```
balicekNemovitosti.vytvorBalicek();
balicekFinanci.vytvorBalicek();
balicekSance.vytvorBalicek();
balicekCeskeDrahy.vytvorBalicek();
balicekSluzby.vytvorBalicek();
```

Dále jsou v konstruktoru nastaveny některé parametry hry na výchozí hodnoty. Je ztmaveno políčko druhého hráče a jako aktivní hráč je nastaven první hráč. V tomto okamžiku je již hra plně připravena.

5.2.4 Třídy Hrac a Hra

Třída `Hra` v sobě udržuje informace o jednotlivých hráčích a poskytuje přístupové metody k nastavování hodnot některých atributů jednotlivým hráčům. Příkladem může být například nastavení financí hráči. Příslušná metoda třídy `Hra` obdrží veškeré potřebné

informace a následně danému hráči například přičte peníze nebo naopak danou částku odečte. Samotný kód pro daný příklad má pak tuto podobu:

```
public void setFinanceHrace(int cisloHrace, int castka,
EnumObecneMoznosti stavFinance){
      if(stavFinance == EnumObecneMoznosti.PRICTI ){
            poleHracu[cisloHrace].dostanPenize(castka);
      }
      if(stavFinance == EnumObecneMoznosti.ODECTI ){
            poleHracu[cisloHrace].zaplatPenize(castka);
      }
            if(stavFinance == EnumObecneMoznosti.NASTAV ){
            poleHracu[cisloHrace].setPocetPenez(castka);
      }
}
```

Třída `Hrac`, jak je z názvu patrné, obsahuje veškeré informace o jednotlivém hráči. Uchovává informace o jménu hráče, stav financí, počet nemovitostí, zda je hráč ve vězení nebo naopak kolikrát může hráč bezplatně opustit vězení a podobně. Při vytvoření instance `Hrac` se konstruktor postará o naplnění všech atributů buď výchozími hodnotami, nebo hodnotami předanými jako parametry. Samozřejmě rozhraní této třídy poskytuje *set* a *get* metody pro nutnou změnu některých těchto atributů v průběhu hry.

5.2.5 Posun figurek po herním plánu

První akcí, kterou hráč většinou vykoná ihned po spuštění samotné hry, je kliknutí na animované tlačítko „Hodit kostkami“. Pokud je možné hodit kostkami, je zavolána metoda `hodKostkami` a po zkontrolování některých parametrů většinou nastane posun figurky hráče na nějaké políčko.

V předchozí teoretické části v kapitole 3.6Vlákna byla jednoduchá ukázka vytvoření vlákna. Dynamické posouvání herních figurek bylo řešenou také pomocí vláken. Realizace byla provedena však druhou možností, a sice implementací rozhraní `Runnable`.

Třída `Monopoly` pro potřeby vláken implementuje rozhraní `Runnable`. Zároveň obsahuje metodu `start()`, která vytvoří nové vlákno, a metodu `run()`, která volá metody pro posun figurek. V metodě run je zjištěno, který hráč je na tahu, a následně je volána metoda `posunHrace(JLabel, int)`.

Metoda `posunHrace(JLabel, int)` má jako parametry odkaz na komponentu zobrazující obrázek hráče a číslo udávající počet políček, o které se má daná figurka posunout. V novém vláknu je při posunu vždy zjištěna pozice figurky. Poté je pomocí metody `setLocation(int, int)` nastavena nová poloha figurky a aktuální vlákno je za pomoci metody `sleep(int)` uspáno na jednu milisekundu. Posun figurky je opakován tak dlouho, dokud není dosaženo cílového umístění herní figurky.

5.2.6 Vykonání akce na políčku

Téměř na každém z políček je potřeba vykonat nějakou událost spojenou s tímto políčkem. Například pokud se hráč zastaví na políčku nemovitosti, kterou nikdo nevlastní, je mu okamžitě nabídnuta možnost nákupu této nemovitosti. O veškeré tyto události se stará třída `Akce` z balíčku `akce`.

Jedinou veřejně přístupnou metodou třídy `Akce` je `vykonejAkciNaPolicku`. Tato metoda obdrží jako hodnoty parametrů především odkazy na jednotlivé balíčky karet a odkaz na daného hráče. Následně je vyhodnoceno, na jakém ze čtyřiceti políček se hráčova figurka nachází. Podle čísla políčka je pak vyhodnoceno, jaká akce k danému políčku přísluší a případně která z privátních metod třídy `Akce` se v tomto případě využije. Tyto privátní metody jsou:

- `akceCeskeDrahy`,
- `akceFinance`,
- `akceNemovitosti`,
- `akceSance`,
- `akceVerejneSluzby`.

Jednotlivé názvy metod naznačují, jakou operaci zajišťují. Mějme například situaci, kdy se hráč, díky svému hodu kostkami, nachází na políčku s nemovitostí. Na tomto políčku je volána metoda `akceNemovitosti`. Nejprve je zjištěno, zda-li tuto nemovitost nějaký hráč vlastní. Pokud ne, je zobrazena nabídka ke koupi. Pokud políčko vlastní nějaký hráč, je zjištěno, kdo je tento hráč. Pokud políčko aktuálnímu hráči nepatří, následuje rozhodování o samotném políčku. Nejprve je zjištěno, zda je políčko v hypotéce. Pokud ano, nic se neděje. Pokud ne, má hráč povinnost zaplatit určitou částku majiteli tohoto políčka. Zjišťuje se tedy, zda jsou na daném políčku postaveny nějaké domy či hotely. Následně jsou provedeny příslušné hotovostní transakce. Na závěr se aktualizuje tabule hráčů pro zobrazení změn.

Podobný systém rozhodování obsahují i metody `akceCeskeDrahy` a `akceVerejneSluzby`. V metodách `akceFinance` a `akceSance` jsou zase realizovány operace spojené s náhodným výběrem karty typu „Šance“ nebo „Finace“.

5.2.7 Karty nemovitostí a služeb

Pro hru Monopoly jsou karty nemovitostí a služeb velmi důležité. Nákup parcel a stavba domů jsou podstatou hry. Pro nemovitosti je výchozí balíček `kartyNemovitosti`. Ten obsahuje dvě třídy a dvě rozhraní k těmto třídám. Třída `KartaNemovitost` nese údaje o každé kartičce nemovitostí. Obsahuje sedmnáct privátních atributů nesoucí informace například o čísle nemovitosti, názvu, ceně, ceně za domy, kdo je vlastníkem nemovitosti a podobně. Pro vytvoření jednotlivé karty je použit konstruktor, který nastaví veškeré atributy a tím v podstatě definuje jednu kartu z balíčku nemovitostí.

Samotný balíček karet všech nemovitostí pak vytváří a spravuje třída `BalicekKaretNemovitosti`. Krom vytvoření herního balíčku také poskytuje přístupové metody k jednotlivým kartám. Samotná třída obsahuje jen jeden privátní atribut, kterým je pole prvků typu `KartaNemovitost`. Toto pole pak reprezentuje balíček karet. Ve třídě Monopoly je vytvořena instance všech herních balíčků. Pro balíček nemovitostí je to následující kód, který v bezparametrickém konstruktoru pouze vytvoří pole pro dvaadvacet prvků (tolik je karet nemovitostí).

```
IBalicekKaretNemovitosti balicekNemovitosti = new
BalicekKaretNemovitosti();
```

Vytvoření herního balíčku, který už obsahuje reálné karty, je zajištěno zavoláním metody `vytvorBalicek()`. Ukázku vytvoření reálné karty zobrazuje následující kód, který se nachází v metodě `vytvorBalicek()`.

```
balicekKaret[0] = new KartaNemovitost(
                        11,
                        "Klimentská ulice",
                        6000,
                        1,
                        5000,
                        5000,
                        3000,
                        200,
                        1000,
                        3000,
                        9000,
                        16000,
                        25000,
                        0,
                        0, new java.awt.Color(153, 51, 0)
                );
```

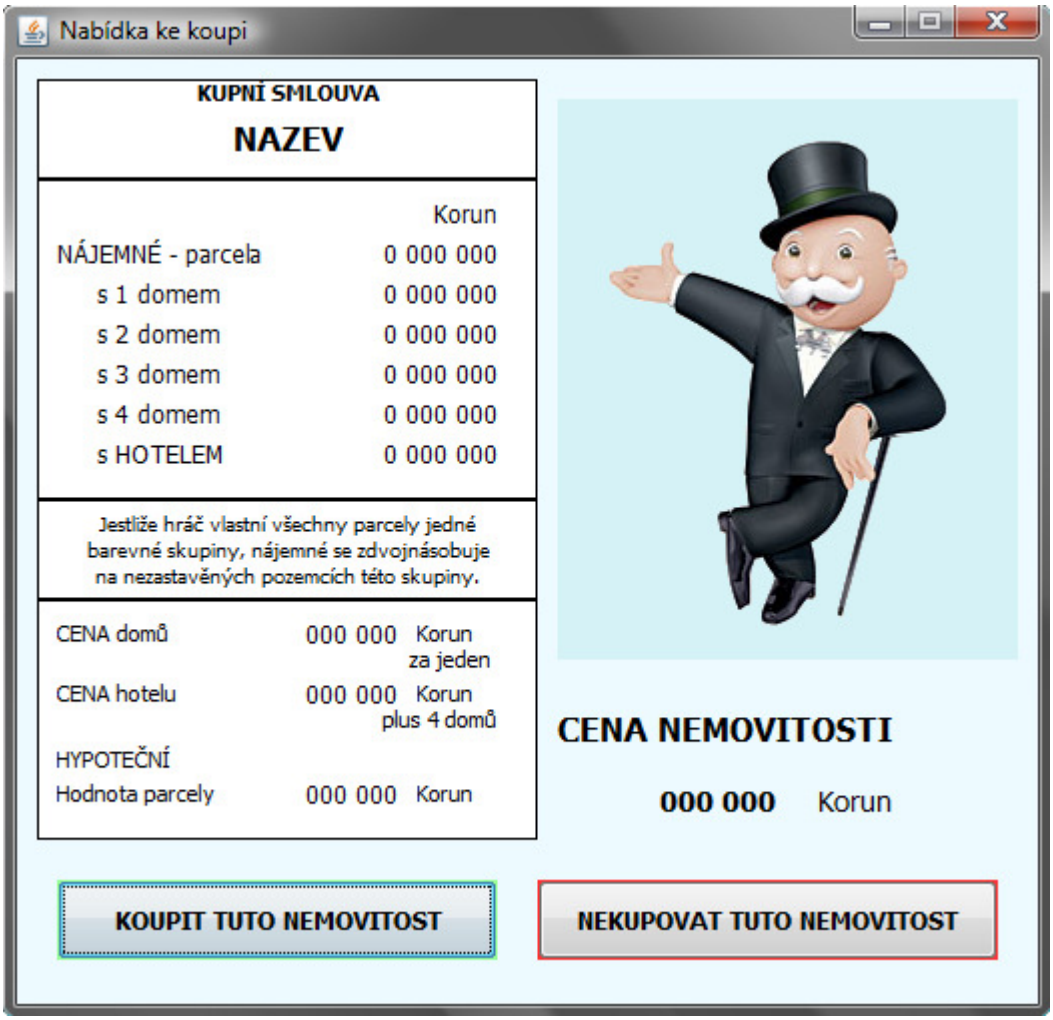
Podobným způsobem jsou vytvořeny i ostatní karty, a tím pádem i celý balíček.

Dále obsahuje třída `BalicekKaretNemovitosti` také metody pro vyhledání jednotlivých karet podle různých kritérií. Především pak také poskytuje metody pro zjištění například počtu karet v barevné skupině a podobně.

Na velmi podobném principu, jež využívá třída `BalicekKaretNemovitosti`, jsou založeny i třídy karet veřejných služeb a třídy spravující balíčky těchto služeb. Výchozí balíček má název `kartySluzeb` a obsahuje čtyři třídy a čtyři rozhraní k těmto třídám. Dvě třídy jsou určeny pro definici karty typu „České dráhy“ a „Veřejné služby“. Druhé dvě třídy spravují balíčky karet „Českých drah“ a „Veřejných služeb“.

Nákup jednotlivých políček je řešen graficky. Jednotlivé třídy jsou v balíčku `nakupPolicek`. Při nákupu například nemovitosti je vytvořena instance třídy `PolickoNemovitostiNakup`. Její konstruktor obdrží mezi parametry mimo jiné i parametr typu KartaNemovitost. Pomocí tohoto parametru je v konstruktoru třídy definován výsledný vzhled kartičky (název, barva, popisky cen). Ukázku grafické podoby

zachycuje následující obrázek (Obrázek 9). Pro zajímavost ještě uveďme, že obrázek vpravo je náhodně generován ze čtyř dalších možností. Tlačítky „KOUPIT TUTO NEMOVITOST“ a „NEKUPOVAT TUTO NEMOVITOST“ může hráč danou nemovitost získat do svého vlastnictví, či nikoliv. Samozřejmě je ověřováno, zda při případném nákupu je na účtu hráče dostatek financí.



Obrázek 9 - Výchozí grafická podoba nákupu políčka

5.2.8 Správce majetku hráče

Důležitou součástí celého herního systému je takzvaný správce majetku. Ve hře je u aktivního hráče kdykoliv možné kliknout na tlačítko „Spravovat majetek“. Zde jsou všechny důležité informace o majetku hráče.

Výchozím balíčkem je `informaceOMajetku`. Ten obsahuje třídu `InformaceOMajetku`, která je opět grafickým vstupně výstupním rozhraním. Konstruktor této třídy obdrží od třídy `Monopoly` veškeré potřebné údaje pro zobrazení všech údajů. Komponenta typu `JList` s názvem `jListSeznamNemovitosti` zobrazuje veškeré nemovitosti a podniky veřejných služeb vlastněných hráčem. Zde je možné po kliknutí na položku zobrazit její detailní informace v panelu uprostřed okna.

V okamžiku, kdy je nějaká karta takto zobrazena, je možné s ní pomocí tlačítek vpravo provádět některé operace. U každé zvolené karty je testováno, zda není v hypotéce. Pokud ano, je na tuto skutečnost hráč upozorněn velkým červeným nápisem vpravo u karty. Hypotéka má totiž několik omezení. Nemovitost v hypotéce nelze prodat, a pokud to typ karty standardně umožňuje, není možné na ní stavět domy.

Na parcelách lze stavět domy a hotely, a tím zvýšit poplatek za „návštěvu“ dané parcely. Pokud má hráč všechny karty jedné barvy a má dostatek financí, je možné dům na parcele postavit. V opačném případě je hráč upozorněn vyskočením varovné hlášky, že danou operaci nemůže z určitých důvodů provést. V případě, že hráč zakoupí dům na zvolené parcele, je zavolána metoda `aktualizujVsechnyDomy()` ze třídy `Monopoly`, která provedené změny okamžitě graficky zobrazí na herním plánu.

5.2.9 Šance a Finance

Balíčky `kartySance` a `kartyFinance` obsahují třídy, které jsou důležité při náhodných událostech, jež vznikají v okamžiku, kdy se hráč zastaví na políčku „Finance“ nebo „Šance“.

Konstruktor třídy Monopoly vytváří instance tříd `BalicekKaretFinance` a `BalicekKaretSance` a podobně jako u nemovitostí jsou pak vytvořeny oba herní balíčky pomocí metody `vytvorBalicek()`.

Typů karet Finance a Šance existuje několik druhů. Karty mohou hráči nařizovat například posun na nějaké políčko nebo zaplacení hotovosti. Proto je vytvořeno několik tříd podle typu karet. Obecně se však vždy jedná a kartu typu „Finance“ nebo „Šance“. Z tohoto důvodu bylo přímo ideální využít vlastnost OOP – dědění.

Příkladem může být karta z balíčku „Finance“ nařizující hráči jít na nějaké políčko. Veškerá podoba třídy reprezentující tento typ karet má následující podobu:

```
public class KartaFinanceJdiNaPole extends KartaFinance implements
IKartaFinanceJdiNaPole{
	private int cisloPolicka;
	public KartaFinanceJdiNaPole() {
	}
	public KartaFinanceJdiNaPole(int id, int poleCislo, EnumTypyKaret
typ, String text) {
		this.cisloPolicka = poleCislo;
		this.idKarty = id;
		this.textKarty = text;
		this.typKarty = typ;
	}
	public int getCisloPolicka() {
		return cisloPolicka;
	}
	public void setCisloPolicka(int cisloPolicka) {
		this.cisloPolicka = cisloPolicka;
	}
}
```

Dalším typem karty může být karta nařizující hráči zaplatit finanční obnos. Ukázka kódu vypadá následovně:

```
public class KartaFinanceZaplaceni extends KartaFinance implements
IKartaFinanceZaplaceni{
	private int castka;
	public KartaFinanceZaplaceni() {
	}
	public KartaFinanceZaplaceni(int id, int castka, EnumTypyKaret typ,
String text) {
		this.castka = castka;
		this.idKarty = id;
		this.textKarty = text;
		this.typKarty = typ;
	}
	public int getCastka() {
		return castka;
	}
	public void setCastka(int castka) {
		this.castka = castka;
	}
}
```

Obě tyto karty dědí, jak je patrné z klíčového slova `extends` od třídy `KartaFinance`. Tělo této třídy obsahuje následující část kódu:

```
	protected String textKarty;
	protected int idKarty;
	protected EnumTypyKaret typKarty;

	public KartaFinance(int id, String textKarty, EnumTypyKaret
typKarty) {
		this.idKarty = id;
		this.textKarty = textKarty;
		this.typKarty = typKarty;
     }
```

Třída `BalicekKaretFinance` pak jako ostatní podobné třídy zajišťuje naplnění privátního atributu pole typu `KartaFinance` různými druhy karet financí. V konstruktoru jsou pak krom jiného i následující řádky kódu:

```
balicekKaret[6] = new KartaFinanceObdrzeni(2, 5000,
EnumTypyKaret.OBDRZENIHOTOVOSTI, "Za prodej akcií obdržíte 5 000 korun");
balicekKaret[11] = new KartaFinanceZaplaceni(12, 1000,
EnumTypyKaret.ZAPLACENIHOTOVOSTI, "Zaplaťte pokutu 1 000 korun.");
balicekKaret[15] = new KartaFinanceJdiNaPole(14, 0, EnumTypyKaret.POSUN,
"Odeberte se na start.");
```

Třída `BalicekKaretFinance` dále obsahuje ještě jednu metodu s názvem `vyberNahodnouKartuZBalicku()`. Zde je využita třída `Random`, jak je patrné z následující ukázky.

```
public KartaFinance vyberNahodnouKartuZBalicku(){
	KartaFinance kartaZBaliku = null;
	int vybranaKarta=0;
	Random nahoda = new Random();
```

```
        vybranaKarta = nahoda.nextInt(16);
        kartaZBaliku = this.balicekKaret[vybranaKarta];
        return kartaZBaliku;
}
```

Na stejném principu, který byl uveden výše, jsou založeny i karty typu `KartaSance`.

5.3 Ovládání hry

Ovládání hry monopoly bylo vytvořeno tak, aby bylo co možná nejvíce intuitivní a nebylo zbytečně složité.

Po spuštění hry se zobrazí obrazovka s volbou jmen a zástupných obrázků pro oba hráče. Je kontrolována vyplněnost políček, takže hráč nemůže zadat jako své jméno prázdné pole. Volba jednotlivých obrázků reprezentujících figurky ve hře je realizována pomocí přepínačů. Nelze zvolit stejný obrázek u obou hráčů současně. Po zadání jmen a výběru obrázku je možné spustit samotnou hru Monopoly.

Hlavní herní obrazovka se skládá ze dvou částí. První část tvoří herní deska. Ta obsahuje jednotlivá políčka, po kterých se herní figurky pohybují. Na každé políčko je možné kliknou, a zobrazit tak podrobnější informace o něm. Druhou část obrazovky pak tvoří informační a ovládací prvky hry.

V pravé horní části se nachází bublina s informačním textem. Zde jsou například uvedeny informace o hodu kostkami, který hráč je momentálně na tahu a podobně. Pod touto bublinou následuje panel s údaji obou hráčů.

V panelu s přehledem jsou uvedeny základní informace obou hráčů: jméno, obrázek, finance a drobný přehled majetku. Když hráč není na tahu, má svou část tabulky neaktivní. Aktivní hráč má svou část tabulky zvýrazněnou a také si může zobrazit detailní informace o svém majetku kliknutím na tlačítko „Spravovat majetek“. V kartě „Majetek hráče“ jsou uvedeny veškeré informace pro daného hráče. Nachází se zde také seznam veškerých nemovitostí a ovládací panel pro operace s těmito nemovitostmi. Při zvolení některé nemovitosti v seznamu se detaily o této nemovitost zobrazí ve vymezeném prostoru vedle seznamu nemovitostí. Pokud je možné nakupovat domy a hotely pro vybrané políčko, lze tak učinit pomocí tlačítka z pravého panelu. V opačném případě je hráč při pokusu o koupi domů a hotelů upozorněn, že tak učinit nemůže. Stejná pravidla platí i při pokusu prodat dům nebo hotel. Pokud je nutné kartu prodat, může tak hráč učinit kliknutím na políčko „Prodat tuto nemovitost“. V případě, že se karty vzdát nechce, lze ji zatížit hypotékou kliknutím na tlačítko „Vzít si HYPOTÉKU na tuto nemovitost“. Pokud je nemovitost v hypotéce, je na tuto skutečnost upozorněno velkým červeným nápisem „V HYPOTÉCE“. Zavřít tento přehled lze kliknutím na levé spodní tlačítko pod listem nemovitostí s nápisem „OK“.

Další část ovládací části hry tvoří panel s tlačítky „Hodit kostkami“ a „Ukončit tah“. Pokud hráč může hodit kostkami, pak na tlačítku probíhá animace, která na tuto

skutečnost upozorňuje. V případě kliknutí na tlačítko „Hodit kostkami“ proběhne posun hráčovy figurky na herním poli. Zároveň je na tlačítku graficky zobrazena i hodnota, která padla na kostkách. Druhé tlačítko umožňuje ukončit tah, a dát tak možnost hrát druhému hráči.

Ve spodní části obrazovky jsou umístěna čtyři tlačítka menu. První zelené tlačítko slouží pro zobrazení základních informací o hře. Oranžové tlačítko slouží pro nastavení. Po kliknutí na toto tlačítko je tak možné zapnout nebo vypnout veškeré zvuky ve hře. Modré tlačítko slouží pro zobrazení detailního návodu ke hře Monopoly. Červené tlačítko ukončuje hru.

Při každém hodu kostkami se figurka hráče posune na určité políčko. Zde nastane většinou nějaká akce. Může se například zobrazit okno pro nákup nemovitosti nebo se okamžitě provede nějaká speciální akce například zaplacení hotovosti. Pokud se hráč zastaví na políčku, které může koupit, okamžitě se zobrazí nabídka ke koupi. Pokud hráč nemovitost koupit nechce, nemusí. V rámci kola má hráč stále možnost kdykoliv se rozhodnout, zda nemovitost koupit, či nikoliv. Po ukončení tahu již hráč nemovitost dodatečně koupit nemůže. Pokud se hráč zastaví na některém z políček s nápisem „ŠANCE“ nebo „POKLADNA“, okamžitě mu bude vylosována a zobrazena náhodná karta z daného balíčku těchto speciálních karet.

5.4 Problémy

Jedním z možných problémů může být velikost okna herní obrazovky. Ta je pevně nastavena na velikost 1006 pixelů šířky a 697 pixelů výšky. Toto nastavení se nedá z pohledu uživatele změnit a okno s hrou proto nelze zvětšovat ani zmenšovat. Důvod pro toto řešení spočívá v zachování přehlednosti a grafické podoby aplikace. Předpoklad je, že většina dnešních uživatelů používá minimální rozlišení monitoru 1024x762 nebo větší.

Druhým problémem může být situace, kdy uživatel, který si chce hru zahrát, nemá nainstalované JRE. V tomto případě se soubor Monopoly nechová jako spustitelný soubor, ale jako klasický adresář.

5.5 Rozšíření hry

Hra Monopoly byla od začátku vytvářena pro dva hráče na jednom počítači. Teoreticky je však možné rozšířit ji na hru pro více hráčů (stále na jednom počítači). Toto rozšířený by znamenalo částečně přepracovat grafické rozhraní aplikace. Grafické úpravy by byly nutné u třídy `UvodDoHry`. Zde by bylo nutné nahradit stávající zobrazení novým, reagujícím na zadaný počet hráčů. Další grafické úpravy by pak byly u třídy `Monopoly`. Pravý panel nyní obsahuje základní informace pro oba hráče. Při rozšíření by bylo potřeba upravit ho pouze pro zobrazení informací jednoho, v tu chvíli aktivního, hráče. Výhodou by však byl větší prostor tedy možnost zobrazit více informací. Z logického hlediska by se musely pozměnit minimálně třídy `Monopoly`, `Hra`, `Hrac`. Řízení hry by poté připadalo mnohem více na třídu `Hra`, než je tomu dosud.

Další možné rozšíření spočívá ve vytvoření síťové hry. Při této realizaci by mohla být zachována varianta hry pro dva hráče. Princip hry by byl založen na modelu klient-server. Pro realizaci by byla nutná přítomnost obou hráčů, přičemž jeden z hráčů by tvořil server, tedy vytvořil novou hru, a druhý hráč by se k nově vytvořené hře připojil. Pro tyto účely by vznikly nové třídy pro oddělení síťových mechanizmů od herních.

6 Závěr

Mým cílem bylo, kromě popisu samotné praktické části hry Monopoly, také seznámit zájemce s programováním 2D grafických aplikací v Javě. Vysvětlit, co je to GUI a jak ho v Javě efektivně využít pro tvorbu desktopových aplikací. Dalším cílem bylo ukázat, jak aplikaci oživit obrázky a jak ji ozvučit. Dále jsem se v krátkosti zmínil o trojrozměrné grafice v Javě. Snažil jsem se jednotlivé uvedené principy vysvětlit nejenom teoreticky, ale i pomocí krátkých ukázek zdrojových kódů prakticky realizovat daný problém. Doufám, že se mi podařilo můj záměr splnit.

Samotná hra Monopoly a její další vývoj jistě není u konce. Jak je uvedeno v kapitole 5.5, budu se nadále snažit realizovat alespoň první uvedené rozšíření. Díky pečlivému testování hry mými přáteli jsem byl také upozorněn na některé drobnosti, či naopak nápady jak hru vylepšit. Těchto připomínek si velice cením, proto bude i nadále mým hlavním cílem vylepšovat stávající verzi hry.

Literatura

Buchalcevová, Alena a Pitka, Lukáš. 2007. *Vývojové prostředí NetBeans.* Praha : Oeconomica, 2007. ISBN 978-80-245-1206-8.

Darwin, Ian F. 2006. *Java Kuchařka programátora.* Brno : Computer Press, 2006. ISBN 80-251-0944-5.

Davison, Andrew. 2006. *Programování dokonalých her v Javě.* Brno : Computer Press, 2006. ISBN 80-7226-944-5.

Herout, Pavel. 2006. *Java - bohatství knihoven.* České budějovice : KOPP, 2006. ISBN 80-7232-288-5.

—. 2007. *Java - grafické uživatelské prostředí a čeština.* České Budějovice : KOPP, 2007. ISBN 978-80-7232-328-9.

—. 2001. *Učebnice jazyka Java.* České Budějovice : KOPP, 2001. ISBN 80-7232-115-3.

Jelínek, Lukáš. 2005. Java (20) - vlákna. *Linuxsoft.cz.* [Online] 2. listopad 2005. [Citace: 12. duben 2011.] http://www.linuxsoft.cz/article.php?id_article=1006.

Kovařík, Jaromír. Moderní společenské hry. *Hrajeme.* [Online] [Citace: 17. duben 2011.] http://www.hrajeme.cz/Hrajeme/StaticText.aspx?id=PAGE_O_HRACH.

Kuželka, Ondřej. 2003. Java - pokročilá grafika (operace s obrázky). *interval.cz.* [Online] 16. prosinec 2003. [Citace: 12. duben 2011.] http://interval.cz/clanky/java-pokrocila-grafika-operace-s-obrazky/.

—. 2003. Java - pokročilá grafika pro všechny. *interval.cz.* [Online] 16. květen 2003. [Citace: 5. duben 2011.] http://interval.cz/clanky/java-pokrocila-grafika-pro-vsechny/.

—. 2003. Java a 3D grafika - graf scény. *interval.cz.* [Online] 19. listopad 2003. [Citace: 16. duben 2011.] http://interval.cz/clanky/java-a-3d-grafika-graf-sceny/.

—. 2003. Java a 3D grafika - úvod. *interval.cz.* [Online] 29. říjen 2003. [Citace: 16. duben 2011.] http://interval.cz/clanky/java-a-3d-grafika-uvod/.

RAFAJ, Nikola. 1999. Slunce nad NetBeans. *Živě.cz.* [Online] 22. 10 1999. [Citace: 21. 3 2011.] http://www.zive.cz/clanky/slunce-nad-netbeans/sc-3-a-8698/default.aspx.

Semecký, Jiří. 2003. Naučte se Javu - grafické uživatelské rozhraní 1. *interval.cz.* [Online] 26. Červen 2003. [Citace: 28. Březen 2011.] http://interval.cz/clanky/naucte-se-javu-graficke-uzivatelske-rozhrani-1.

—. 2003. Naučte se Javu - grafické uživatelské rozhraní 2. *interval.cz.* [Online] 9. červenec 2003. [Citace: 28. Březen 2011.] http://interval.cz/clanky/naucte-se-javu-graficke-uzivatelske-rozhrani-2.

Tišnovský, Pavel. 2010. Squeak a Smalltalk - historie vývoje grafického uživatelského rozhraní. *ROOT.CZ.* [Online] 9. Září 2010. [Citace: 26. Březen 2011.] http://www.root.cz/clanky/squeak-a-smalltalk-historie-vyvoje-grafickeho-uzivatelskeho-rozhrani.

Příloha A – Zdrojový kód demonstrující práci s obrázky v GUI

```
package ukazka;

import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;
import java.io.IOException;
import javax.imageio.ImageIO;

public class UkazkaObrazku extends javax.swing.JFrame {
      public UkazkaObrazku(){
            initComponents();
      }
      @Override
      public void paint(Graphics g){
            Graphics2D g2d = (Graphics2D) g;
            BufferedImage bufferedImage = null;
            try{
                  bufferedImage =
                  ImageIO.read(getClass().getResource(„obrazek.bmp“));
            } catch (IOException ex) {
                  // osetreni vyjimky
            }
            g2d.drawImage(bufferedImage, null, 20, 50);
      }
      @SuppressWarnings("unchecked")
      private void initComponents() {
            setDefaultCloseOperation(
                  javax.swing.WindowConstants.EXIT_ON_CLOSE);
            javax.swing.GroupLayout layout =
                  new javax.swing.GroupLayout(getContentPane());
            getContentPane().setLayout(layout);
            layout.setHorizontalGroup(
                  layout.createParallelGroup(
                  javax.swing.GroupLayout.Alignment.LEADING)
                  .addGap(0, 400, Short.MAX_VALUE));
            layout.setVerticalGroup(
                  layout.createParallelGroup(
                  javax.swing.GroupLayout.Alignment.LEADING)
                  .addGap(0, 300, Short.MAX_VALUE));
            pack();
      }
      public static void main(String args[]) {
            java.awt.EventQueue.invokeLater(new Runnable() {
                  public void run() {
                        new NewJFrame().setVisible(true);
                  }
            });
      }
}
```