ZÁPADOČESKÁ UNIVERZITA V PLZNI

FAKULTA EKONOMICKÁ

Bakalářská práce

Tvorba výukového programu na podporu výuky statistiky na středních školách

Jan Harmady

Plzeň 2012

Prohlašuji, že jsem bakalářskou práci na téma

Tvorba výukového programu na podporu výuky statistiky na středních školách

vypracoval samostatně pod odborným dohledem vedoucího bakalářské práce
za použití pramenů uvedených v přiložené bibliografii.

V Plzni, dne 16.4.2012 ……

podpis autora

Poděkování

Na tomto místě bych rád poděkoval Ing. Mgr. Milanu Svobodovi za cenné rady a odborné vedení při zpracování této bakalářské práce.

Obsah

Úvod

Statistika je předmět, který spojuje ekonomii, výpočetní techniku, český jazyk a v neposlední řadě matematiku. Na některých školách se statistika dokonce vyučuje jako součást matematiky.

K úspěšnému zvládnutí statistiky jsou potřeba nejen teoretické znalosti statistiky, ale především logické myšlení a schopnost porozumět textu. To ze statistiky dělá u mnoha žáků velmi neoblíbený předmět. Žáci často rezignují již po prvních pokusech o pochopení probírané problematiky a snaží se naučit jakési „pseudopostupy“ řešení úloh typu „když je tady tohle, tak to napíšu sem“. Tato snaha žáků usnadnit si práci a méně přemýšlet vychází především z myšlenky, že statistikou projdou a dále už se s ní nesetkají. Tato myšlenka je však zcela mylná. Je sice možné, že žáci nebudou v budoucnu statistická data třídit a analyzovat, ale určitě budou muset využívat data, která z těchto analýz vzejdou.

Snadná cesta k naučení statistiky, respektive k naučení aplikace poznatků získaných z teoretické části statistiky, neexistuje. Schopnost žáků pochopit učivo je sice ovlivněna vrozenou mírou logického myšlení, které jim studium může ulehčit, ovšem bez procvičování a námahy mozkových závitů se statistiku nenaučí nikdo. Proto jsem se rozhodl vytvořit výukový program na podporu statistiky na středních školách. Cílem tohoto programu bude doplnit klasické učebnice, ve kterých jsou příklady neměnné, stále se stejnými čísly i zadáními. To by tento program měl odstranit a generovat dynamicky do zadání nová čísla, tak aby student mohl své znalosti ověřovat na stále „nových“ příkladech.

Na následujících stránkách budou popsány technologie, které byly použity k vytvoření výukové programu na podporu výuky statistiky. Popsán bude programovací jazyk Java, uživatelské rozhraní Swing nebo princip objektově orientovaného programování. Všechna tato témata jsou velmi obsáhlá a v jednom dokumentu prakticky nepopsatelná, proto v této práci jsou popsány jen stručné základy těchto technologií.

V první části je také uvedena charakteristika Sportovní a podnikatelská střední školy, která mi umožnila program testovat a implementovat do výuky.

Ve druhé části práce je popsána struktura programu, funkce jeho tříd a metod. Zároveň také stručná uživatelská dokumentace, díky níž by uživatel měl být schopen program nainstalovat, spustit a úspěšně se v něm orientovat.

Program není nikdy dokonalý, proto jsou v této práci také uvedeny možnosti dalšího vývoje aplikace. Konkrétně dvě cesty vývoje. První jako rozšiřování desktopové aplikace a druhá předělaní programu na servlet a JSP a aplikaci distribuovat jako webovou.

Cílem této práce je navrhnout a vytvořit program v programovacím jazyce Java na podporu výuky statistiky na střední škole. Tento program by měl nahradit nebo doplnit učebnici statistiky pro střední školy.

Aplikace bude obsahovat teoretický popis vybraného učiva a jeho vysvětlení na příkladech. Dalším obsahem programu bude soubor příkladů na procvičení této vybrané středoškolské látky, do kterých se budou dynamicky generovat nové číselné hodnoty.

Ke splnění tohoto cíle bude potřeba navrhnout strukturu programu, jeho funkcionalitu a také algoritmy pro výpočet příkladů.

V rámci této práce je cílem vytvořit pouze základní funkcionalitu a aplikaci distribuovat jako desktopovou aplikaci, ovšem navrženou tak, aby její úprava, ať už přidání dalších funkcionalit nebo změna na aplikaci webovou, byla co možná nejsnazší.

1 Výukový program a výuka statistiky na SŠ

Na většině středních škol je kombinatorika a pravděpodobnost součástí předmětu matematika. Tato bakalářská práce byla vytvořena ve spolupráci se Sportovní a podnikatelskou střední školou, s.r.o. v Plzni, na které je statistika vyučována jako samostatný předmět a je zařazena do třetího ročníku studia na všech studijních programech.

1.1 Výukový program

Výukový (edukační) program (software) je jakékoli programové vybavení počítače plnící alespoň jednu z didaktických funkcí. Didaktické funkce jsou funkce motivační, expoziční, fixační a verifikační. (Dostál, 2009)

Výukové programy lze třídit podle mnoha kriterií. Program vytvářený v rámci této práce je možné popsat jako interaktivní se zpětnou vazbou. Žák v takovém programu může ovlivňovat jeho běh a zároveň program vyhodnocuje vstupy zadané uživatelem. Dále je monouživatelský a didakticky polyfunkční, program ovládá jen jeden uživatel, kterému program v rámci funce expoziční, fixační a verifikační nabízí výuku látky. Program je zaměřen na předmět statistika a je určen žákům středních škol. Program je možné využívat k samostudiu i je možné jej využít učitelem při výuce.

Program, který je výstupem této práce, byl implementován dle obsahu výuky na Sportovní a podnikatelské střední škole v Plzni.

1.2 Sportovní a podnikatelská škola Plzeň

Tato střední škola byla založena v roce 1992 jejím zřizovatelem Mgr. Jaroslavem Šlechtou. Škola svým studentům nabízí čtyřleté středoškolské vzdělání s maturitou v rámci oboru Ekonomika a podnikání. Student si dále může zvolit jeden ze studijních programů, podle svého studijního nebo sportovního zaměření.

Na všech čtyřech oborech školy, které jsou stručně popsány níže, se statistika vyučuje ve třetím ročníku studia jako samostatný předmět. Hodinová dotace předmětu je 2 hodiny. Hodiny probíhají v multimediální učebně, kde žáci mají přístup k počítačům, a tak učitel může při výuce využívat software, včetně toho, který je výstupem této práce.

1.2.1 Firemní management a marketing

Studijní program Firemní management a marketing je určen studentům se zájmem o ekonomii. Pro studium tohoto programu není podmínkou aktivně sportovat. Studenti tohoto programu si, mimo základních středoškolských předmětů jako jsou český jazyk, matematika, atd., osvojí také dva cizí jazyky a získají znalosti z oblastí ekonomie, ekonomiky, marketingu a psychologie.

1.2.2 Sportovní management

Tento studijní program je určen aktivním sportovcům, především fotbalistům a hokejistům. Podmínkou přijetí pro tento program jsou i talentové zkoušky. Rozsah znalostí je stejný jako u programu Firemní management a marketing, rozšířený o sportovní teorii a praxi.

1.2.3 Výpočetní technika

Studijní program Výpočetní technika umožňuje získat běžné znalosti a znalosti z oblastí hardware a software studentům se zájmem o využití výpočetní techniky v podnikání.

1.2.4 Personalistika a psychologie managementu

Absolvent tohoto studijního programu získá znalosti ekonomie, financí a také se naučí pracovat s lidmi. Bude vhodným pracovníkem do sektoru PR nebo HR.

1.3 Učební plán

Na Sportovní a podnikatelské střední škole se statistika vyučuje podle školského vzdělávacího programu (ŠVP) platného od 1. 9. 2009, podle kterého je cílem předmětu: „seznámit žáky se základními postupy při řešení kombinatorických a pravděpodobnostních úloh, zpracování hromadných údajů, jejich analýze a následné interpretaci“. Statistika je odborným předmětem, pro který jsou nezbytné znalosti z matematiky. Dalšími předpoklady jsou znalosti ekonomiky a marketingu, kterých se týká většina zadání příkladů pro procvičování, dále žák využije znalosti z českého jazyka pro správné pochopení textu nejen příkladů, ale také teoretických definic. Předmět také souvisí s psychologií, hospodářským zeměpisem, základy společenských věd a také výpočetní technikou, která usnadňuje zpracování a vyhodnocování dat.

Obsah výukového programu byl na základě ŠVP rozdělen do čtyř tematických celků.

1.3.1 Kombinatorika

V této první části je žák seznámen s faktoriálem a kombinačními čísly. Naučí se využívat základní kombinatorická pravidla, tedy kombinatorické pravidlo součtu a kombinatorické pravidlo součinu. Dále jsou žákovi vysvětleny variace, permutace a kombinace, které následně uplatňuje při řešení příkladů.

1.3.2 Pravděpodobnost

V druhém celku je žákovi vysvětlen pojem náhodný jev a jeho pravděpodobnost. Žák se naučí vyjádřit pravděpodobnost jednoho jevu a následně pracovat s pravděpodobnostmi více náhodných jevů. Tedy vypočítá pravděpodobnost jejich sjednocení a průniku.

1.3.3 Grafy

Celek grafy je zaměřen na prezentaci statistických dat, žák se naučí, poté co jsou mu objasněny pojmy statistický soubor, znak, četnost, číst v tabulkách relativních četností, ze kterých je schopen sestavit vhodný graf. Žák je seznámen grafy sloupkovými, výsečovými a spojnicovými.

1.3.4 Charakteristiky znaku

Poslední látka zařazená do výukového programu nazvaná charakteristiky znaku obsahuje dvě charakteristiky popisující statistický soubor, a to charakteristiky polohy a charakteristiky variability. Žák je seznámen především s aritmetickým průměrem prostým a váženým, dále s pojmy modus a medián. Ve druhé části charakterizující variabilitu, je vysvětleno variační rozpětí, rozptyl a směrodatná odchylka.

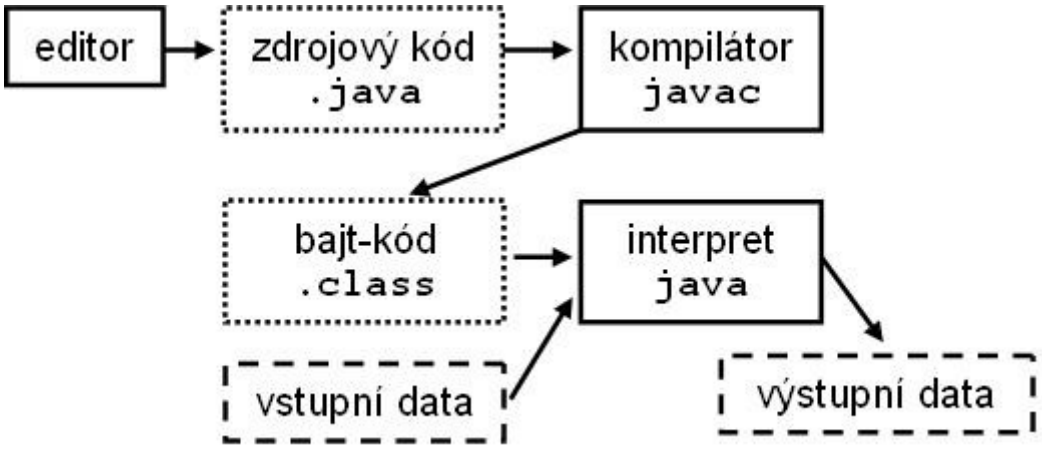
2 Java

K tvorbě výukového programu pro statistiku byl vybrán programovací jazyk Java. Tento jazyk byl zvolen zejména s ohledem na vlastnosti jazyka, které jsou posány níže, a to přenositelnost, robustnost a snadnou práci s ním, která je způsobena objektovou orientací a snadnou čitelností kódu. Oproti jazykům jako C nebo nebo C# je Java pomalejší, ovšem rozdíl v rychlosti je v dnešních verzích Javy pro uživatele zanedbatelný. Jazyk Java je také možné využít pro tvorbu webové aplikace. I v této fázi vývoje aplikace by bylo vhodné využít jazyka Java, jelikož často používané php převyšuje především díky typové kontrole, objektové orientaci a také rychlosti. Java je kompilována pouze při prvním spuštěním a dále se interpretuje jen bajtkód, v php musí proces kompilace a interpretace proběhnout při každém spuštění skriptu.

Java je tedy objektově orientovaný programovací jazyk, který je v současné době jedním z nejrozšířenějších programovacích jazyků na světě. Java je zjednodušenou verzí jazyka C, ze kterého vychází. Technologie Java zahrnuje programovací jazyk a platformu.

Java je programovací jazyk vyšší úrovně. Ve většině literatury je popisován jako jednoduchý jazyk, který je přenositelný a nezávislý na architektuře. Tato přenositelnost a nezávislost Javy je její velkou výhodou, aplikace napsaná v jazyce Java bude fungovat stejně na všech operačních systémech. Aplikace totiž běží na Java virtuálním stroji (JVM), který tyto dvě vlastnosti zaručí. Dalšími vlastnostmi jsou robustnost a bezpečnost, jazyk Java je velmi spolehlivý, nedovoluje programátorovi psaní konstrukcí, které bývají příčinou chyb. Každá proměnná musí mít nadefinovaný datový typ, neboť Java používá silnou typovou kontrolu. Java je jazykem hybridním, to znamená, že je zároveň jazykem interpretovaným i kompilovaným. Program se nejprve pomocí kompilátoru přeloží do bajtkódu, který je následně interpretován virtuálním strojem (JVM). (Zakhour, 2007)

Obrázek 1 : Překlad a interpretace programu v jazyce Java



Zdroj: Přednášky KIV/PPA1 na ZČU, 2012, A. Netrvalová

Bylo zmíněno, že technologie Java osahuje i platformu. Platforma je softwarové prostředí umožňující spouštět programy. Tato platforma Java funguje nad jinými platformami (operačními systémy jako Windows, Linux, atd.). Platforma Java zahrnuje komponenty Virtuální stroj jazyka Java (JVM) a Aplikační programové rozhraní Java (API). (Pecinovský, 2004)

2.1 Historie jazyka Java

Jazyk Java začala vyvíjet společnost Sun Microsystems v roce 1991. Nový programovací jazyk se snažila vytvořit modifikací jazyků C a C++ a společnost ho pojmenovala Oak, ovšem programovací jazyk s takovým názvem již existoval a proto musel být přejmenován, jazyk dostal nový název Java. Tento nově vznikající jazyk byl původně určen pro spotřební elektroniku (pračky, mikrovlnné trouby, atd.). Sun představil Javu na konferenci v roce 1995. Tento jazyk v tomto roce již počítal s rostoucí důležitostí webu a webových aplikací, na využití na webu vývojáři společnosti pracovali od roku 1993. První technologií pro tvorbu webových aplikací byl Applet. Ten byl ale zatlačen s příchodem nové Java platformy J2EE podporující vývoj webových aplikací (servlet). (Herout, 2007)

Při svém uvedení obsahoval jazyk Java pouhých 211 veřejných tříd. Jazyk ovšem sklidil velký úspěch, což vedlo k jeho rychlému rozšiřování. Další verze, která vyšla dva roky po prvním uvedení, již byla rozšířena na 477 veřejných tříd. Jednalo se o verzi 1.1 a novinkou této verze bylo také zavedení vnitřních a vnořených tříd.

V roce 1998 vyšla další verze jazyka Java 1.2. V této verzi došlo k přepracování knihoven a zařazení nových, jako například JFC Swing. Počet veřejných tříd této verze byl 1524.

Verze 1.3, která přišla v roce 2000, přinesla zrychlení části API a rozšíření o další třídy. Verze 1.4 se věnovala hlavně zrychlení kódu.

Významné rozšíření jazyka přinesla verze 1.5, která je častěji nazývána jako Java 5. Tato verze byla vydána v roce 2004 a obsahovala již 3270 veřejných tříd.

Ve verzi 1.6 k zásadním změnám nedošlo, bylo opět rozšířeno API.

Zatím poslední verzí je Java 1.7. Tuto verzi již nevydala společnost SunMicrosystems, protože byla v roce 2009 koupena společností Oracle. James Gosling, tvůrce jazyka Java, který stál v roce 1991 u jeho zrodu, byl velkým kritikem tohoto spojení a odešel do společnosti Google. Mezi těmito dvěma společnostmi se stále vedou spory o patentech týkajících se jazyka Java a Androidu (platforma pro mobilní zařízení od společnosti Google). (Herout, 2007)

2.2 Objektově orientované programování

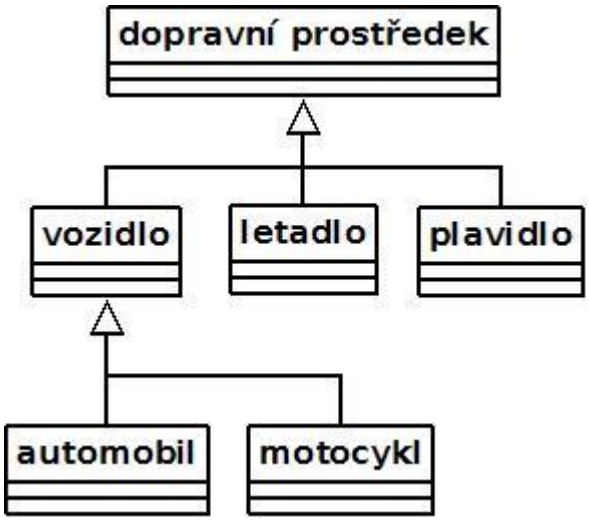
Základem objektově orientovaného programování jsou objekty, které se vyznačují stavem a chováním. Softwarové objekty koncepčně odpovídají reálným objektům kolem nás, které také mají stav (kniha může být otevřená, zavřená, …) a chování (knihu můžeme otevřít, zavřít, otočit list, …). U softwarových objektů se stav nazývá proměnná a chování metoda. Stav objetu můžeme měnit pomocí metod, k proměnným totiž není možné přistupovat zvenku, to by porušovalo jeden ze základních principů objektově orientovaného programování – zapouzdření dat. (Zakhour, 2004)

Výhodou objektů a objektově orientovaného programování je, že programátor ve svém kódu může využít již existující objekty, které vytvořil dříve nebo je vytvořil jiný programátor. Programátor využívající cizí objekt nemusí znát kód tohoto objektu, což je další z výhod objektově orientovaného programování, stačí znát metody v objektu obsažené a jejich funkci. Neposlední výhodou je snadná opravitelnost kódu, pokud objekt obsahuje chybu, nemusíme předělávat celý program, ale stačí opravit nebo nahradit daný objekt.

Dalším důležitým pojmem objektově orientovaného programování je třída. Třída je jakási šablona, podle které objekty vznikají. Tyto vzniklé objekty nazýváme instancemi třídy. V každé třídě jsou deklarovány proměnné, tedy uvedeny stavy, které objekt obsahuje. A také implementovány metody, které tyto stavy mění. Každá třída, také obsahuje konstruktor. Ten je volán při vzniku instance třídy a většinou je v něm stavům objektu přiřazen výchozí stav.

Objekty v reálném světě mají často stejné stavy nebo chování. Proto objektově orientované programování zavádí dědičnost. Třída dědí jak proměnné, tak i metody. Třída, která dědí, se nazývá potomek, třída od které je děděno se nazývá rodič. Potomek dědí vše, co obsahuje rodič a navíc může být rozšířen o další proměnné nebo metody. To umožňuje programátorovi, aby se u odděděných tříd věnoval jen novým funkcím objektu.

Obrázek 2 : Příklad dědičnosti



Zdroj: vlastní zpracování, 2012

Na předchozím obrázku (obr. 2) je znázorněna dědičnost. Dopravní prostředek je nadřazená třída, která nemá žádného rodiče. Může mít v stavy jako kapacita náladového prostoru, maximální počet převážených osob, atd. její metody pak budou sloužit k práci s těmito stavy. Potomci třídy dopravní prostředek jsou třídy vozidlo, letadlo a plavidlo. Všechny tři třídy dědí stavy dopravního prostředku a metody pro práci s nimi, a navíc přidávají vlastní. Například vozidlo může obsahovat stavy jako počet kol nebo typ tažného zařízení. Tyto třídy mají další potomky. Na obrázku jsou uvedeny potomci jen pro třídu vozidlo. I potomků třídy automobil bychom dokázali najít více, ale pro znázornění budou stačit třídy automobil a motocykl. Tyto dvě třídy dědí stavy a metody dopravního prostředku a vozidla. Dále přidávají vlastní stavy a chování. U automobilu to může být značka, typ nebo obsah motoru.

2.3 Grafické uživatelské rozhraní

Pro tento pojem se často používá zkratka GUI z anglického Graphical User Interface a označuje vzhled a rozhraní programu pomocí kterého vypisujeme výstupy a načítáme vstupy od uživatele. Program je zobrazen jako okno, ve kterém jsou poskládány komponenty. Takový program je ovládán z větší části myší, na rozdíl od programů ovládaných z příkazové řádky pomocí klávesnice.

V jazyce Java je možné využít dvě grafická uživatelská prostředí. Starší z nich je AWT (Abstract Windowing Toolkit), novější JFC Swing (Java Foundation Classes). Swing je součástí Java Core API od JDK 1.2. Swing je prostředí sice novější a přišel s několika vylepšeními oproti AWT, ovšem je na AWT závislý. Swing používá stejný model událostí jako AWT a je postaven na základech AWT. Některé balíky ze staršího AWT dokonce Swing beze změn přebírá. Swing ale obsahuje více balíků a tříd a nabízí programátorovi výrazně více možností.

Největší rozdíl mezi AWT a Swingem je že AWT si nechává všechny komponenty vykreslovat od operačního systému, to sice zajistí vyšší rychlost, ale knihovna AWT může obsahovat jen ty komponenty, které jsou na všech platformách, proto je výrazně menší knihovnou než Swing.

Swing si komponenty vykresluje sám, takže jich obsahuje více a jejich vzhled se na různých operačních systémech neliší. AWT je sice také přenositelné, funkce komponent je na všech operačních systémech stejná, ovšem vzhled se může lišit, jelikož závisí na operačním systému, jak danou komponentu vykreslí.

Jak již bylo zmíněno, Swing vychází z AWT. Všechny komponenty Swingu dědí od třídy JComponent, která je zase potomkem třídy Container obsažené v knihovně AWT. Závislost mezi těmito dvěma prostředími je tedy naprosto zřejmá.

Knihovna JFC Swing obsahuje velké množství tříd. V následujících odstavcích je popsána jen několik nejdůležitějších z nich pro tvorbu programu s tímto uživatelským rozhraním.

Kontejnery jsou objekty, do kterých je možné vkládat další objekty. Nejpoužívanější jsou JFrame a JPanel. JFrame je komponenta, od které programátor oddědí své výstupní okno. Takové okno lze minimalizovat, maximalizovat, měnit jeho velikost nebo ho zavřít. Třída JFrame má předdefinované metody pro práci s tímto oknem, jako jsou

setSize() pro nastavení velikosti, setVisible() pro vykreslení okna nebo setDafaultCloseOperation() pro nastavení akce, která se rozběhne po zavření okna křížkem v jeho pravém horním rohu.

JPanel je nejjednodušší komponentou pro organizování ostatních komponent. Používá se k návrhu vzhledu. JPanelu lze totiž přiřadit layout manager, který zajistí rozložení komponent po JPanelu.

Layout manažerů, nebo také správců rozvržení, je několik různých druhů. Nejjednodušší je FlowLayout. Ten je defaultně nastaven každému JPanelu, proto se nemusí v kódu nastavovat. Tento správce rozvržení umisťuje komponenty do řádky vedle sebe a při dosažení konce řádky přejde na řádku další. Při použití tohoto správce nemá programátor příliš velkou kontrolu, jelikož komponenty se přeskládají při každé změně velikosti obrazovky. Komponenty lze pomocí FlowLayoutu zarovnat na střed, doprava nebo doleva.

Trochu složitější jsou správci rozvržení GridLayout a GridBagLayout. Jedná se o tabulkové manažery, kteří rozdělí JPanel (nebo jiný kontejner) na řádky a sloupce. Rozdíl mezi těmito správci je ten, že GrigLayout neumožňuje upravovat buňku, lze jen nastavit počet řádků a sloupců a následně přidávat komponenty, které se vkládají zleva doprava ve vkládaném pořadí. Všechny buňky mají stejnou velikost. GrigBagLayout umožňuje pomocí instance třídy GridBagConstraints definovat buňku, do které se má komponenta vložit, pak také definovat kolik buněk má komponenta zabírat nebo zda má být zarovnána vpravo, vlevo či na střed.

Nejpoužívanějším správcem rozvržení je BorderLayout. Ten rozděluje kontejner na pět částí. Jsou jimi sever, jih, východ, západ a střed. Každá z těchto částí má svoji specifickou vlastnost, co se reakcí na změny velikosti týká. Severní a jižní části si neustále udržují konstantní výšku a mění pouze šířku. Naopak východní a západní části mění svoji výšku a šířka zůstává stále stejná. Středová část vyplňuje prostor mezi ostatními částmi, a proto mění výšku i šířku. Tento správce rozvržení odpovídá modernímu rozvržení aplikací, u kterého se menu nachází v horní (severní) části, v levé nebo pravé (západ, východ) se nacházejí navigační prvky, další menu, obsah a podobné. Dole (jižní část) se objevují informativní výpisy. A v centrální části (střed) je samotný hlavní obsah stránky (okna).

Do kontejnerů se podle správců rozvržení vkládají komponenty pro výpisy a obsluhu programu. Takovými komponentami jsou například pro výpis JLabel nebo JTextArea a pro ovládání tlačítka (JButton) nebo JMenu. Komponentám pro výpis se text, který se má zobrazit, přiřadí v konstruktoru nebo pomocí metody setText. Kromě metody setText obě třídy obsahují mnoho dalších metod pro práci s textem, jejich vzhledem a funkčností. Aby komponenty pro ovládání programu byly funkční, musejí jim být přiřazené události. Java má velmi propracovaný koncept událostí, který Swing převzal ze staršího AWT. Událost vzniká uživatelskou akcí a posluchač, který je komponentě přiřazen provede akci definovanou ve svém těle. (Herout, 2007)

2.4 Vývojové prostředí

Pro psaní a překládání programů v jazyce Java je nutné mít nejprve nainstalované vývojové prostředí systému Java. Standartním systémem pro překlad je JDK (Java Development Kit), který je distribuován společností Oracle. JDK obsahuje dva důležité programy. Pro překlad slouží javac.exe a pro interpretaci java.exe.

Pro napsání jednoduššího programu stačí mít k dispozici jakýkoli textový editor, jako Poznámkový blok. Kód se po napsání uloží s příponou .java a přeloží z příkazové řádky pomocí překladače obsaženého v JDK. Kód v těchto jednodušších editorech ovšem není příliš přehledný. Existují proto vhodnější editory, které obsahují řadu doplňků usnadňujících programátorovi psaní. Takovým editorem může být SciTE (Scintilla Text Editor). Takový editor zvýrazňuje syntaxi, umožní kompilaci a spuštění programu na integrované konzoli.

Další možností v čem vyvíjet kód jsou vývojová prostředí nebo spíše platformy IDE (Integrated Development Environment). V současné době je nejpopulárnějším z těchto platforem Eclipse sponzorovaný společností IBM nebo prostředí NetBeans. (Brůha, 2006)

3 Návrh funkcionality

Funkcionality jsou rozděleny na dva typy. První budou klasické funkcionality programu, neboli činnosti, které uživatel provádí přímo v programu po jeho spuštění. Druhým typem budou funkcionality, které se budou provádět v okolí programu a po spuštění programu ovlivní jeho obsah.

3.1 Programové funkcionality

Po spuštění programu má žák možnost zvolit, zda chce studovat teorii nebo se procvičovat na příkladech.

V části teorie žák může zvolit jednu z kapitol, které jsou Kombinatorika, Pravděpodobnost, Grafy a Charakteristiky znaku. Tyto kapitoly obsahově odpovídají předmětu statistika na Sportovní a podnikatelské střední škole. Poté co žák rozklikne kapitolu se zobrazí podkapitola úvod do vybraného učiva a dále může žák vybrat podkapitolu dle svých potřeb.

V této části programu může žák studovat teorii, vrátit se zpět do hlavního menu, přepnout do části řešené příklady, přepnout do části neřešené příklady nebo ukončit program.

Další dvě části programu jsou podobné. Jedná se o části řešené příklady a neřešené příklady. Společné pro tyto části je, že může vybrat kapitolu pro procvičování, případně v ní typ příkladu k procvičování, je možné zadat výsledek příkladu a ověřit jeho správnost nebo vygenerovat nové zadání příkladu. Dále stejně jako v části teorie vrátit se zpět do hlavního menu, přepnout do části teorie, přepnout do části neřešené (řešené) příklady nebo ukončit program.

Část řešené příklady je navíc rozšířena o možnost zobrazení postupu řešení daného příkladu.

3.2 Další funkcionality

Program svá data čerpá ze složek umístěných v balíku jar, ve kterém se exportuje. Jar (neboli **J**ava **Ar**chive) je klasický archív, který slouží k distribuci .class souborů. Díky tomu, že obsahuje manifest s cestou ke třídě s metodou main, se stává tento archív

spustitelným. Tento archív jde, stejně jako jiné archívy, otevřít pomocí WinRaru, 7-Zipu a jiných programů. Pokud takto otevřeme tento archív, dostaneme se ke zdrojovým datům, meta-datům a souborům s daty tvořící náplň programu. Zde můžeme tato data mazat, přidávat nebo měnit.

Právě v tomto archívu je možné přidat, smazat nebo upravit zadání příkladů. Stejně tak můžeme přidávat, mazat nebo měnit obrázky s teorií a řešením příkladů. Jak tato data upravovat bude vysvětleno níže.

4 Popis struktury programu

V rozsáhlejších aplikacích se programátor snaží dodržet třívrstvou architekturu, tedy oddělit třídy pracující s daty, třídy pro prezentaci a samotnou logiku programu. V aplikaci, která dodržuje třívrstvou architekturu, se snadněji provádějí změny. Existují v ní totiž třídy specializované na určitou činnost, a pokud chceme tuto činnost upravit, například změnit v datové vrstvě formát načítaných dat, stačí upravit jednu třídu, ve které načítání probíhá. Pokud by třívrstvá architektura neexistovala, programátor upravující kód by musel vyhledávat všechna místa v programu, kde daná činnost probíhá a změnit ji. Odchylky od třívrstvé aplikace se v aplikaci, vytvářené v rámci této práce, vyskytují. Například při vkládání a tvorbě grafů se architektonické vrstvy aplikace prolínají.

UML diagram tříd je k dispozici v příloze, stejně jako celý kód programu.

4.1 Vrstva rozhraní

Vrstva rozhraní, nebo také bývá označována jako vrstva prezentační, je tou kterou uživatel vidí na monitoru svého počítače. Je to tedy vrstva sloužící k zobrazení výstupů programu a k zadání vstupů uživatelem.

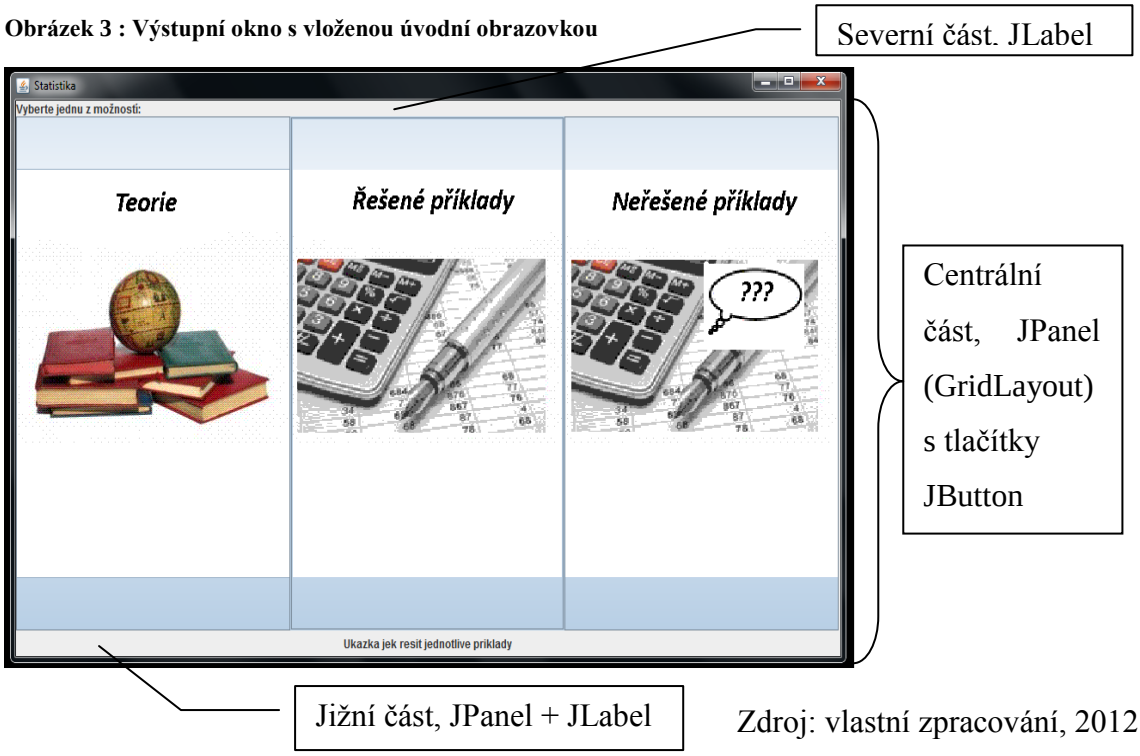
V této aplikaci byla použita knihovna JFC Swing.

Pro zobrazování výstupů programu jsem vytvořil okno (výstupní okno) dědící od JFrame. Velikost tohoto okna je defaultně nastavena na 975 px šířky a 600 px výšky. Taková velikost byla zvolena na základě testování na monitorech o rozlišení 1366 x 768 a 1024 x 600. Do toho okna se vkládají další komponenty.

Po spuštění programu se do výstupního okna vloží komponenta úvodní obrazovka. Ta dědí od komponenty JPanel a obsahuje několik dalších komponent. Úvodní obrazovce je přidělen layout. Layout, neboli rozložení (plán), vymezuje sektory, do kterých je možné vkládat další prvky. V Javě, respektive Swingu, je možné využít několik předdefinovaných layoutů. Nejjednodušší je FlowLayout, který umisťuje prvky vedle sebe, GridLayout a GridBagLayout umisťují prvky do mřížky. GridBagLayout umožňuje upravovat mřížku. Asi nejpoužívanější layout manager je BorderLayout. Ten umožní vložení maximálně pěti komponent do oblastí sever, jih, východ, západ a střed.

A právě BorderLayout je přidělen úvodní obrazovce. V severní a jižní části jsou informativní výpisy a ve středu tři tlačítka umístěná na panelu (JPanel) s GridLayoutem. Západní a východní části jsou prázdné.

Obrázek 3 : Výstupní okno s vloženou úvodní obrazovkou



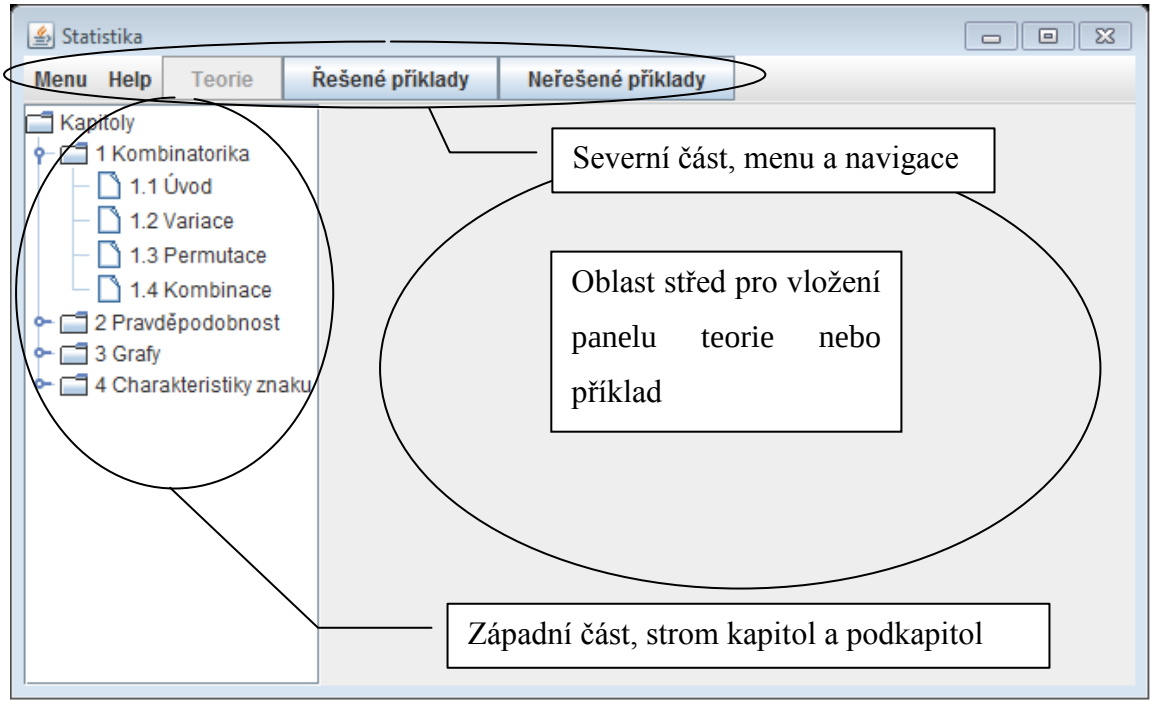
Zdroj: vlastní zpracování, 2012

Kliknutím na jedno z tlačítek úvodní obrazovky přecházíme na pracovní obrazovku. Pracovní obrazovka je také potomkem JPanel a stejně jako úvodní obrazovka využívá rozvržení podle BorderLayout manageru a také využívá jen jeho tři části. K Borderlayoutu je potřeba ještě zmínit jednu vlastnost. Při měnění velikosti okna si severní a jižní část zachovává konstantní výšku, východní a západní část konstantní šířku a středová část je proměnlivá. Což je vlastnost velmi praktická. Ale zpátky k pracovní obrazovce. V severní části je umístěna komponenta JMenuBar. Ta obsahuje položky pro navigaci v rámci programu, možnost zobrazení nápovědy a ukončení programu. V západní části pracovní obrazovky se nachází strom kapitol, ten je vypsán pomocí komponenty JTree, která definuje jeho vzhled a reakce na události. Jeho vlastnosti byly mírně upraveny pro větší uživatelské pohodlí. Při defaultní chování komponenty musí uživatel kapitolu nejprve dvojklikem rozbalit a následně si vybrat podkapitolu, po úpravě stačí na kapitolu kliknout jednou, ta se rozbalí a „přeskočí“ na první podkapitolu, kterou je většinou úvod do daného učiva. Třetí částí je část středová. Ta je sice při prvním načtení pracovní obrazovky zdánlivě prázdná, ale je připravena

metoda pro vložení dalšího panelu do tohoto prostoru, ve který obsahuje JScrollPane. Komponenta JScrollPane je komponenta podobná JPanelu, ovšem má předdefinovaný horizontální a vertikální posuvník, který se zobrazí ve chvíli, kdy obsah JScrollPane je větší, než komponenta samotná. Na komponentě JScrollPanel je položen i strom kapitol.

Další dvě komponenty, ze kterých se skládá uživatelské rozhraní programu, jsou také potomci nejpoužívanějšího kontejneru, který slouží pro seskupení jiných komponent a řízení jejich rozmístění, tedy JPanelu. Tyto dvě komponenty jsou panel teorie a panel příklad. Oba se vkládají po příslušné události do pracovní obrazovky do oblasti střed.

Obrázek 4 : Pracovní obrazovka programu s oblastí pro vložení panelů teorie a příklad



Zdroj: vlastní zpracování, 2012

Panel teorie sdružuje komponenty pro vykreslení slajdu s teorií a tlačítek pro přepínání mezi nimi. Komponenty na tomto panelu jsou opět rozmístěny s pomocí BorderLayout manageru. Severní část obsahuje výpis nadpisu aktuálně vybrané podkapitoly. Je vypsán na prvku JLabel, který umožňuje krátký, needitovatelný výpis textu. V jižní části se nachází další informativní výpis s číslem aktuálně zobrazeného slajdu a celkový počet slajdů v kapitole. V centrální části je samotný obrázek s teorií a tlačítka pro zobrazení následujícího a předcházejícího slajdu.

Panel příklad řídi rozložení komponent pomocí GridBagLayoutu. Ve výsledku je panel rozdělen na tři základní části. V horní třetině je připraven prostor pro výpis zadání, konkrétně JTextArea se zakázanou možností editování, která se hodí pro výpis delších textů. Ve středu panelu jsou seskupeny komponenty pro vložení výsledku, potvrzení výsledku, vygenerování nového zadání a pro zobrazení řešení příkladu. Tlačítko pro zobrazení řešení je funkční pouze v případě sekce řešené příklady. Prostor pro vykreslení řešení, které se zobrazuje formou obrázku, je v dolní třetině panelu příklad.

V prezentační vrstvě můžeme najít ještě jednoho zajímavého potomka třídy JPanel. Jedná se o třídu panel kreslení. Tato třída slouží k vykreslování všech obrázku, ať už se jedná o slajdy teorie nebo řešení příkladů. K zobrazení obrázku máme k dispozici předdefinovanou metodu paintComponent() a v ní použijeme metodu drawImage(). V metodě drawImage jsou obrázku přiděleny souřadnice a velikost.

Pro vykreslení grafů je připraven další potomek objektu JPanel, a to třída graf. Tato třída využívá externí knihovnu JFreeChart, která je napsána v Javě, distribuována ve dvou souborech JAR a umožňuje vykreslení velkého množství grafů. Knihovna je zdarma ke stažení na http://www.jfree.org/jfreechart (6. 4. 2012), včetně dokumentace. Ve třídě graf jsou implementované metody pro vykreslení grafu výsečového a sloupkového, další přibydou s dalším rozšiřováním aplikace.

Poslední třídou v prezentační vrstvě je třída nápověda. Spouští se po zvolení možností nápověda v menu. Tato třída je velmi jednoduchá. Jejím rodičem je třída JFrame, takže vytváříme nové okno. Do tohoto okna se na komponentu JTextArea vypíší texty s pokyny pro uživatele, jak postupovat v modu, ve kterém se aktuálně nachází.

4.2 Vrstva logiky a výpočtů

Vrstva logiky je kostrou programu. V této vrstvě jsou definované reakce na události přicházející z prezentační vrstvy, prováděny výpočty a odesílány požadavky do datové vrstvy na objekty a dále jsou zde tyto objekty zpracovávány. Vrstva logiky nebo také aplikační vrstva či Business logic v této aplikaci obsahuje jen tři třídy. Jsou to třídy příklad, řešitel a řídící třída.

Třída příklad reprezentuje objekt příklad. Obsahuje proměnné pro zaznamenání hodnot potřebných k identifikaci příkladu. Tyto hodnoty jsou číslo zadání, číslo kapitoly, číslo řešení, číslo postupu řešení a výsledek příkladu. Číslo řešení je číslo, pod jakým je v souboru uložen obrázek s ukázkou řešení. Číslo postupu řešení je číslo algoritmu jakým se daný příklad bude v aplikaci řešit. Dále tato třída obsahuje už jen metody get a set pro nastavování hodnot proměnných a získávání těchto hodnot.

Jednou z nejdůležitějších tříd programu je řídící třída. Ta je zároveň třídou spouštěcí, jelikož obsahuje metodu main. Kromě metody main, ve které se pouze vytvoří nová instance řídící třídy, obsahuje tato třídy další metody. Metoda provedAkci() je metoda, do které přicházejí veškeré požadavky z událostí zachycených uživatelským rozhraním. Každá z těchto událostí má své číslo a v metodě proveAkci() je rozhodováno, kam program zamíří dále.

Reakcí je v této metodě implementováno celkem sedm:

1 - vložení panelu pracovní obrazovka a natavení statutu na hodnotu 0 (teorie)

2 - vložení panelu pracovní obrazovka a natavení statutu na hodnotu 1 (řešené příklady)

3 - vložení panelu pracovní obrazovka a natavení statutu na hodnotu 2 (neřešené příklady)

4 - vložení panelu teorie nebo příklad (podle statusu)

5 - výpis řešení

6 - ověření správnosti výsledku zadaného uživatelem

7 - výpis nápovědy

Do každého zadání příkladu se hodnoty, se kterými se dále počítá, generují dynamicky. Hodnoty ovšem musejí být v určitém intervalu tak, aby v zadáních příkladů dávaly smysl. O vygenerování těchto hodnot se starají metody dogenerujCisla() a getArgumenty(). Samotné generování hodnot probíhá v metodě getArgumenty(), metoda dogenerujCisla() přetypuje čísla na objekty a pomocí třídy MessageFormat tyto objekty vloží do zadání.

V programu je možné vybrat si ze dvou variant příkladů. Příklady řešené mají pevně popsaný postup řešení, které je možné zobrazit, a proto i hodnoty vkládané do zadání musejí být vždy stejné, jinak by řešení neodpovídala zadání. Příklady neřešené nejsou omezené nutností zobrazit postup řešení, stačí jen, aby program vypočetl správný výsledek z vygenerovaných hodnot, a tak mohou hodnoty do těchto zadání být generovány náhodně.

Vygenerované hodnoty se tedy vloží do zadání a také do třídy příklad, konkrétně do její proměnné nazvané „parametry“. Hodnoty jsou uloženy v jednorozměrném poli typu double. Tento datový typ je 64 bitové číslo s pohyblivou desetinou čárkou. Pole těchto čísel má dvacet míst indexovaných od 0 do 19.

První pětice čísel v tomto poli (indexy 0 až 4) jsou čísla od hodnoty 0 do 1 s maximálně dvěma desetinnými místy. Toho je docíleno tím, že se nejprve vygeneruje náhodní číslo od velikosti 0 až 100 a následně vyděleno stem. Výsledná hodnota je uložena do pole hodnot. Tyto hodnoty jsou dobře využitelné u příkladů o pravděpodobnosti.

Na indexech 5 až 9 můžeme najít hodnoty od 10 do 100. Opět je vygenerováno přirozené číslo menší než sto, dále se ověří, zda je menší než 10 a pokud ano, tak vygenerovanou hodnotu zvětšíme o 20. O dvacet proto, že pro čísla menší než právě 20 je vyhrazeno speciální místo, tak šlo o to docílit větší četnosti výskytu na hodnotách větších než 20. Tato čísla lze použít u příkladů na charakteristiky znaku pro počítání průměrů nebo variačního rozpětí, tedy v příkladech, kde je žádoucí větší rozptyl generovaných hodnot.

Třetí část pole (indexy 10 až 14) a poslední čtvrtá (15 až 19) část jsou podobné, jedná se o hodnoty 3 až v prvním případě 10 a ve druhém 20. Pokud je vygenerované číslo menší, než 3 pak je zvětšeno o právě 3. Tyto hodnoty jsou v aplikaci nejpoužívanější. Vyskytují se ve většině typů příkladů.

Třetí, poslední, třídou logické vrstvy je třída řešitel. V ní probíhají výpočty výsledků příkladů. Obsahuje konstruktor, ve kterém je inicializována proměnná typu příklad. Dále jednu veřejnou metodu getVysledek() v níž dojde, na základě hodnoty „postup řešení“ získané z objektu příklad, k volání jedné z privátních metod, ve kterých je příklad řešen.

Tyto privátní metody jsou popsány v následujících odstavcích.

Metoda pro výpočet aritmetického průměru má dva parametry. Jeden z nich je nazvaný *odkud* a druhý *n*. Oba jsou typu integer (číslo). Metoda vypočte aritmetický průměr z *n* hodnot, které jdou v poli parametrů za sebou od indexu *odkud*. Metoda nejprve vypočte součet hodnot ve for-cyklu, výslednou hodnotu vydělí hodnotou *n* a tuto hodnotu vrátí.

```
private double aritmetickyPrumer(int odkud, int n) {
	double v = 0;
	for (int i = odkud; i < n + odkud; i++) {
		v = v + priklad.getParam()[i];
	}
	v = v / n;

	return v;
}
```

Metoda pro výpočet variačního rozpětí zjistí největší a nejmenší hodnotu z *n* hodnot jdoucí v poli parametrů za sebou od indexu *odkud*. Do proměnné max je nejprve uložena nejmenší hodnota datového typu integer, která se získá z proměnné MIN_VALUE objektu Interger. Následně jsou for-cyklem procházeny hodnoty pole a porovnávány s hodnotou uloženou v proměnné max. Toto porovnávání probíhá pomocí objektu Math v metodě max, která vrací menší ze dvou hodnot. Získání nejmenší hodnoty probíhá analogicky. Do proměnné min je uložena největší hodnota Integer.MAX_VALUE, porovnání ve for-cyklech probíhá metodou Math.min. Na závěr jsou hodnoty v proměnných max a min od sebe odečteny a výsledek metoda vrací jako variační rozpětí.

```
private double variacniRozpeti(int odkud, int n) {
	double min = Integer.MAX_VALUE;
	for (int i = odkud; i < n + odkud; i++) {
		min = Math.min(min, priklad.getParam()[i]);
	}
	double max = Integer.MIN_VALUE;
	for (int i = odkud; i < n + odkud; i++) {
		max = Math.max(max, priklad.getParam()[i]);
	}
	double reseni = max - min;

	return reseni;
}
```

Další metoda slouží pro výpočet rozptylu. Stejně jako předchozí metody má dva parametry *odkud* a *n*. Tentokrát jsou ovšem původní hodnoty zvětšeny tisíckrát a uloženy do nově vytvořeného pole. K výpočtu rozptylu musíme znát průměr. K tomu využijeme již známou metodu pro výpočet aritmetického průměru a výslednou hodnotu zvětšíme opět tisíckrát, protože v tomto řádu se pohybujeme (využíváno pro příklady se

mzdami a podobně). Dále je hodnota rozptylu nastavena na nulu a v cyklu je k této hodnotě vždy přičtena druhá mocnina příslušné hodnoty zmenšené o aritmetický průměr. Nakonec je takto získaná hodnota dělena počtem hodnot a vrácena metodou jako rozptyl.

```
private double rozptyl(int odkud, int n) {
	double[] hodnoty = new double[n];
	for (int i = 0; i < hodnoty.length; i++) {
		hodnoty[i] = priklad.getParam()[i + odkud] * 1000;
	}
	double prumer = aritmetickyPrumer(odkud, n) * 1000;
	double rozptyl = 0;
	for (int i = 0; i < hodnoty.length; i++) {
		rozptyl += ((hodnoty[i] - prumer) * (hodnoty[i] -
		prumer));
	}
	double reseni = rozptyl / n;
	return reseni;
}
```

Definována je i metoda pro výpočet modu. Modus je hodnota s nejčastějším výskytem, zjišťuje se proto, kolikrát se každá hodnota v souboru vyskytuje a tato četnost se porovná s doposud nejvyšší četností a v případě, že je větší uloží se místo ní a zaznamená se i hodnota tohoto nejčetnějšího znaku. Rozdělení může být bimodální, to ovšem v aplikaci lze realizovat složitě, proto v případě shody četností výskytu, považujeme za výsledek vyšší hodnotu znaku.

```
private double modus(int odkud, int n) {
	double reseni = priklad.getParam()[odkud];
	int pocetMax = 1;
	int pocet= 1;
	for (int i = odkud; i < (odkud+n); i++) {
		double a = priklad.getParam()[i];
		if(a == priklad.getParam()[12]) {
			++pocet;
		}
		for (int j = i+1; j < (odkud+n); j++) {
			if(a == priklad.getParam()[j]) {
				++pocet;
			}
		}
		if(pocet > pocetMax) {
			pocetMax = pocet;
			reseni = a;
		}
		else if(pocet == pocetMax) {
			reseni = Math.max(reseni, a);
		}
		pocet = 1;
	}
	return reseni;
}
```

Další metody lze použít k řešení kombinatorických příkladů.

Metoda pro výpočet faktoriálu s parametrem *n* reprezentujícím hodnotu, jejíž faktoriál se snažíme zjistit. Proměnnou řešení nastavíme na hodnotu jedna a dále ji ve for-cyklu násobíme hodnotou v každém cyklu zmenšovanou o jedna, dokud je tato hodnota větší než nula. Pokud bychom chtěli pomocí této metody zjistit faktoriál čísla nula, pak by cyklus neproběhl ani jednou, proto je zpočátku řešení nastaven na hodnotu 1, což také výsledek faktoriálu nuly.

```
private double faktorial(double n) {
	double reseni = 1;
	for (double i = n; i > 0; i = i - 1) {
		reseni *= i;
	}
	return reseni;
}
```

Metodu faktoriál ke svým výpočtům využívají další metody třídy řešitel. První z nich je metoda pro výpočet kombinací bez opakování. Tato metoda má dva parametry. Parametr *k* je počet vybíraných prvků, parametr *n* reprezentuje celkový počet prvků. Nejprve pomocí metody faktoriál spočteme čitatele, první část jmenovatele a druhou část jmenovatele. Následně řešení snadno získáme jako podíl čitatele a součinu jmenovatelů.

```
private double kombinaceBezOpakovani(double k, double n) {
	double citatel = faktorial(n);
	double jmenovatel1 = faktorial(n-k);
	double jmenovatel2 = faktorial(k);

	double reseni = citatel / (jmenovatel1 * jmenovatel2);
	return reseni;
}
```

Kombinace s opakováním se počítají stejným způsobem. Odlišnost je pouze v parametrech posílaných do metody faktoriál.

```
private double kombinaceSOpakovanim(double k, double n) {
	double citatel = faktorial(k + n - 1);

	double jmenovatel1 = faktorial(n-1);
	double jmenovatel2 = faktorial(k);

	double reseni = citatel / (jmenovatel1 * jmenovatel2);
	return reseni;
}
```

Stejně tak je tomu i metody řešící variace bez opakování. Postup je stejný, rozdíl je opět v parametrech metod faktoriál a tentokrát také ve jmenovateli.

```
private double variaceBezOpakovani(double k, double n) {
	double citatel = faktorial(n);
	double jmenovatel = faktorial(n-k);

	return (citatel / jmenovatel);
}
```

Metoda pro výpočet variací s opakováním se od předchozích odlišuje. Parametry zůstávají stejné, ovšem postup výpočtu variací s opakováním je n^k. O umocnění celkového počtu prvků počtem vybíraných prvků se stará metoda pow() třídy Math.

```
private double variaceSOpakovanim(double k, double n) {

	double reseni = Math.pow(n, k);
	return reseni;
}
```

Další výpočty se snadno zapíší do jednoho řádku, proto nebylo potřeba vytvářet pro ně speciální metody a proběhnou přímo v metodě getVysledek() v jednotlivých case switche, který rozhoduje podle čísla postupu řešejí z objektu příklad.

```
public double getVysledek() {
	switch (priklad.getPostupReseni()) {
	case 0:
		break;
	case 1:
		vysledek = aritmetickyPrumer(5, 5);
		break;

	.

	.

	.

	default:
		break;
	}

}
```

Postupem číslo jedna zjistíme aritmetické průměr z hodnot na pozicích 5, 6, 7, 8, 9 pole parametrů (generované pole viz výše). Postup využívá již zmíněnou metodu pro výpočet aritmetického průměru.

Číslo 3 je postup pro výpočet variačního rozpětí na indexech 10, 11, 12, 13, 14. Výsledek z metody pro výpočet variačního rozpětí je vynásoben číslem 1000. Využívá se především pro příklady se mzdami, proto se pohybuje v řádu tisíců.

Case 4 vypočte rozptyl z pěti hodnot od indexu 10 včetně, metodou pro rozptyl.

Postup číslo 5 zjistí počet kombinací bez opakování druhé třídy z počtu prvků na indexu 17 v poli argumentů.

Postup 6 jsou také kombinace bez opakování. Ovšem o trochu složitější. Výsledek je součin kombinace bez opakování třetí třídy z počtu hodnot na indexu 12, kombinace bez opakování druhé třídy z počtu hodnot na indexu 17 a kombinace bez opakování první třídy ze dvou (tato poslední kombinace se rovná 2).

Case 7 uloží do proměnné výsledek počet kombinací s opakování, kde celkový počet prvků je index 15 a vybíráme tolik prvků, kolik je na indexu 10.

Číslo 8 zavolá metodu faktoriál s parametrem, který je uložen na indexu 17 v poli parametrů.

Pro výpočet kombinačních čísel slouží postup číslo 9. Kombinační číslo vypočte z indexů 8 a 17. Na indexu osm je vždy číslo větší nebo rovno deset a na indexu sedmnáct je číslo menší než deset, výraz tak bude vždy dávat smysl. Algoritmus výpočtu přesně kopíruje vzorec pro kombinační čísla, s využitím metody faktoriál.

Postupy 10 a 11 jsou prakticky stejné. Jedná se o součin faktoriálů, který lze využít i pro součin permutací. Výsledek je roven součinu faktoriálů z indexů 15, 16 a 17. U postupu 11 je tento výsledek navíc vynásoben ještě faktoriálem čísla tři, tedy číslem 6.

Postup číslo 12 je vhodný pro počítání pravděpodobností. Výsledek je totiž roven součinu hodnoty na indexu 0, hodnoty 1 – hodnota na indexu 1 (pravděpodobnost opačného jevu) a hodnoty 1 – hodnota na indexu 2 (pravděpodobnost opačného jevu).

Číslo 13 je obdobné číslu 12, také vhodný pro počítání pravděpodobností. Takto vypadá kód postupu 13:

```
Výsledek = 1 - (1 - priklad.getParam()[0]) * (1 - priklad.getParam()[1]) * (1
- priklad.getParam()[2]);
```

Zjistí opačnou pravděpodobnost z indexů 1, 2 a 3 a vypočte pravděpodobnost, že nenastane jejich průnik. Takový postup se hodí pro příklady typu: „Pravděpodobnost, že

výrobek obstoji v první zkoušce je 0,4, ve druhé 0,8 a ve třetí 0,1. Jaká je pravděpodobnost, že výrobek obstoji alespoň v jedné zkoušce?“.

Postup 14 využije metodu pro výpočet variací bez opakování a zjistí počet kombinací třetí třídy z počtu prvků uvedených v poli parametrů na pozici 15.

Stejné hodnoty použije i postup číslo 15, ovšem posílá je do metody pro řešení variací s opakováním.

Postup 16 se hodí k výpočtu směrodatné odchylky. Do výsledku tento postup uloží odmocninu z rozptylu hodnot z pole parametrů na indexech 15, 16, 17, 18 a 19.

Postupem číslo 17 je možné zjistit aritmetický průměr z osmi hodnot na pozicích 8, 9, 10, 11, 12, 13, 14 a 10 v poli parametrů. K výpočtu je využita metoda pro výpočet aritmetického průměru. Ta vrátí aritmetický průměr ze sedmi hodnot. Tento výsledek je vynásoben počtem prvků, ze kterého byl vypočten (7) a je připočtena hodnota z indexu 10. Výsledný součet je vydělen počtem prvků (osmi) a výsledek uložen do proměnné.

Case 18 vypočet modus podle výše popsané metody s parametry *odkud = 11* a *n = 9*.

Postup 19 zjistí počet variací bez opakování druhé třídy z počtu prvků na indexu 15 v poli parametrů.

Postup číslo 20 slouží k výpočtu rozdílu mezi variací čtvrté třídy z pěti prvků a variací třetí třídy ze čtyř prvků, obě variace jsou bez opakování a využívána je metoda pro jejich výpočet. Příklad, pro který je toto řešení vhodné, může vypadat následovně: „Kolik čtyřciferných čísel lze sestavit s různými číslicemi 0; 4; 6; 5; 9?“.

4.3 Vrstva datová

Datová vrstva, jak již název napovídá, slouží k práci s daty. V případě tohoto programu je využívána k načítání textových dat a obrázků z příslušných souborů. Obsahuje pouze jedinou třídu nazvanou DAO, z anglického Data Access Object. Tento výraz se používá pro třídy webových aplikací, které pracují s daty uloženými v databázi. Tato desktopová aplikace ovšem není napojena na databázový server s databází, a tak třída DAO pracuje se složkami a soubory obsaženými v JAR balíku.

Třída DAO obsahuje několik metod pro zpracování požadavků z logické vrstvy. Tyto metody vracejí do logické vrstvy data v požadovaném formátu. Implementovány jsou metody nactiTeorii, nactiZadani, getReseni, getPocetPrikladuVKapitole a dále pomocné metody nactiPriklad, zjistiPocetPrikladu a prevodNazvu.

Pro práci s datovými proudy jsou v Java knihovnách k dispozici třídy obsažené v balíku java.io. Při načítání je nejprve nastavena cesta a vytvořena instance abstraktní třídy InputStream, dále je vytvořen BufferedReader, při jehož vytváření se do konstruktoru přidává parametr instanci třídy InputStreamReader, která má zase jako parametr již vytvořenou abstraktní třídu InputStream, ve které je nastavena cesta k požadovanému souboru. Pomocí metod třídy BufferedReader pak můžeme číst z textového souboru. To aplikace využívá u načítání teorie a načítání příkladů.

Načítání obrázků probíhá pomocí třídy ImageIcon, které je jako parametr konstruktoru přidělena cesta k požadovanému souboru. Ze které výsledný obrázek typu Image získáme metodou getImage.

Metoda nactiTeorii má jako parametr název kapitoly v textové podobě. Metoda vrací pole obrázku. Pole obrázku je datového typu Image a je vytvořeno pomocí kolekce ArrayList, což je pole, ovšem není omezené velikostí. Prvky do tohoto pole vkládáme a třída ArrayList se sama stará, aby velikost pole byla dostatečná. Parametr název kapitoly je převeden na číslo kapitoly a je k němu přidáno písmeno „T“. Výsledná hodnota je názvem složky, ve které jsou hledané obrázky uloženy.

V každé takové složce jsou uloženy obrázky s teorií ve formátu GIF, který byl zvolen díky velmi nízké náročnosti na paměť, přestože není tak kvalitní jako třeba JPG. Tyto obrázky jsou nazvány čísly od nuly, číslo označuje, v jakém pořadí budou uživateli zobrazovány. V této složce je navíc uložen textový soubor obsahující jedinou hodnotu, a to počet obrázků ve složce.

Metoda nactiTeorii zjistí počet obrázku z textového souboru pomocí metody readLine() třídy BufferedReader, a tuto hodnotu využije pro definování počtu průběhů cyklu, ve kterém se načítají samotné obrázky teorie a ukládají se do pole (ArrayList). Celý kód metody je zabalen do try-catch bloku, pro případ, kdy načítaná složka nebo soubor neexistují.

```
public ArrayList<Image> nactiTeorii(String vybranaKapitola) {

	int[] p = prevodNazvu(vybranaKapitola);
	String cisloKapitoly = p[0] + ""+ p[1];


	ArrayList<Image> pole = new ArrayList<Image>();
	try {
		InputStream inS = getClass()
				.getResourceAsStream(
					"/teorie/" + cisloKapitoly
					+ "T/pocetListu.txt");
		BufferedReader bfr = new BufferedReader(new
		InputStreamReader(inS));
		int pocetListu = Integer.parseInt(bfr.readLine());

		for (int i = 0; i < pocetListu; i++) {
			ImageIcon imI = new
			ImageIcon(getClass().getResource("/teorie/"     +
			cisloKapitoly + "T/" + i + ".gif"));

			Image obr = imI.getImage();
			pole.add(obr);
		}
	}catch(Exception e) {
		e.getStackTrace();
	}
	return pole;
}
```

Zadání příkladů jsou uložena v textovém souboru priklady.txt. V prvním řádku souboru je číslo označující nejvyšší počet příkladů v kapitole, tedy počet příkladů v kapitole, která jich obsahuje nejvíce. Druhá řádka je informativní a uvádí v jakém formátu zadávat příklady. Každý příklad je na jedné řádce a ta má následující podobu. První je číslo označující kapitolu, následuje mezera a za ní číslo identifikující příklad, opět mezera a další číslo je číslo řešení, tedy hodnota, kterou je nazván obrázek s řešením příkladu. Za další mezerou se nachází číslo označující postup řešení, o tom byla řeč v logické vrstvě u třídy řešitel. Za následující mezerou je samotný text příkladu.

Text příkladu je v jedné řádce, což není úplně vhodné. K odřádkování textu slouží znak /n. Hodnoty, které má program dogenerovat sám, se zapisují do složených závorek, ve kterých je číslo indexu vstupující do parametrů (viz logická vrstva, řídící třída), ze kterého chceme číslo vložit.

Samotné načtení pak probíhá pomocí třídy BufferedReader, jejíž metodou readLine získá program všechny informace o příkladu v jedné řádce. Řádek vybere podle čísla kapitoly a čísla příkladu. Číslo příkladu je buď zadáno uživatelem (lze vybrat konkrétní příklad) nebo generováno náhodně. Další rozhodovacím kritériem je, zda příklad má

nebo nemá mít řešení ve formě obrázku. Aplikace k výběru správného příkladu používá metodu nactiPriklad, která prochází textový soubor po řádcích, dokud nenarazí na odpovídající zadání. Toto zadání vrátí metodě nactiZadani a ta ho převede na pole s prvky text příkladu, číslo řešení, číslo příkladu, postup řešení.

Další metodou třídy DAO je getReseni, ta vrací obrázek s řešení k příkladu, podobně jako metoda nactiTeorii. Metoda getReseni je jednodušší, jelikož obrázek s řešením je jen jeden, který načte pomocí třídy ImageIcon a vrátí.

Metoda getPocetPrikladuVKapitole projde soubor priklady.txt a vrátí počet příkladů v dané kapitole, toho je využíváno pro správné vygenerování stromu s kapitolami a příklady v uživatelském rozhraní.

Posledními dvěma třídami, které napomáhají činnostem ostatních metod této třídy, jsou metody prevodNazvu a metoda zjistiPocetPrikladu. Druhá jmenovaná vrací hodnotu uvedenou na prvním řádku souboru priklady.txt, neboli počet příkladů v nejpočetnější kapitole. Podle této hodnoty se následně generuje číslo příkladu. Vygeneruje se hodnota menší než toto číslo (žádná jiná kapitola více příkladů neobsahuje) a pokud vygenerované číslo je vyšší než počet příkladů vybrané kapitoly, pak se hodnota půlí dokud příklad s takovým číslem v kapitole není objeven.

Názvy kapitola jsou ve tvaru „X kapitola“, kde X je číslo kapitoly, každá kapitola obsahuje příklady značené „X.Y Příklad“, kde X je číslo kapitoly do které spadá a Y je číslo příkladu. Metoda převod názvu z tohoto značení separuje číslo kapitoly a případně číslo příkladu a vrací tyto dvě hodnoty v poli. Pokud se jedná o kapitolu (není obsaženo číslo příkladu), pak dosadí hodnotu nula a číslo se dogeneruje později.

5 Testování programu

Testování vytvořeného programu probíhalo ve dvou fázích. V první fázi byl program testován ve spolupráci s vedoucím této práce. Ve druhé fázi byl program nasazen do vyučování na Sportovní a podnikatelské střední škole a testován žáky.

5.1 Alfa testování

Prvotní testování bylo provedeno metodou white-box. Metoda white-box je způsob testování při kterém tester zná algoritmy programu a testuje, zda pracují správně.

V této fázi testování byly objeveny nesrovnalosti při generování hodnot doplňovaných do zadání. Generování hodnot tedy následně bylo navrženo do své konečné podoby uvedené výše, tak aby zadání příkladů dávala smysl.

Dále byly zjednodušeny metody ve třídě Resitel.java a také optimalizováno jejich využití tak, aby nedocházelo k duplicitám v kódu.

5.2 Beta testování

Druhé testování proběhlo na Sportovní a podnikatelské střední školy ve spolupráci s žáky třetího ročník. Toto testování probíhalo metodou black-box, žáci neznali algoritmy, ale testovali, zda výstupy programu odpovídají realitě. Také testovali uživatelskou příjemnost rozhraní.

Po tomto testování bylo nutné v programu provést několik změn. Byly objeveny nejednoznačnosti v zadáních příkladů, ty byly opraveny tak, aby byly správně pochopitelné. Dále byly po tomto testování provedeny úpravy ve vertikálním menu (strom kapitol a podkapitol), tak aby bylo uživatelsky příjemnější a akce bylo možné provádět jedním kliknutím.

Dále bylo přidáno více slajdů s teorií, jelikož pro žáky střední školy byla teorie příliš stručná a žáci nebyli na jejím základě schopni řešit obsažené příklady. I tento nedostatek byl odstraněn.

6 Obsluha programu

Následující kapitoly nahrazují uživatelskou dokumentaci. Bude zde vysvětleno, jak program vlastně naistalovat, spustit, a nejen jak ovládat program při jeho běhu, ale také jak upravovat data obsažená v programu.

6.1 Instalace a spuštění

Aplikace je implementována v jazyce Java, proto je nutné mít nainstalované běhové prostředí (JRE – Java Runtime Environment). JRE umožňuje programy v jazyce Java spouštět, pro úpravy kódu nebo psaní kódu vlastního je nutné mít nainstalované JDK (viz výše). JRE je možné stáhnout na webových stránkách společnosti Oracle, na adrese http://www.oracle.com/technetwork/java/javase/downloads/jre-6u31-download-1501637.html. Stačí vybrat správnou instalaci pro Váš operační systém.

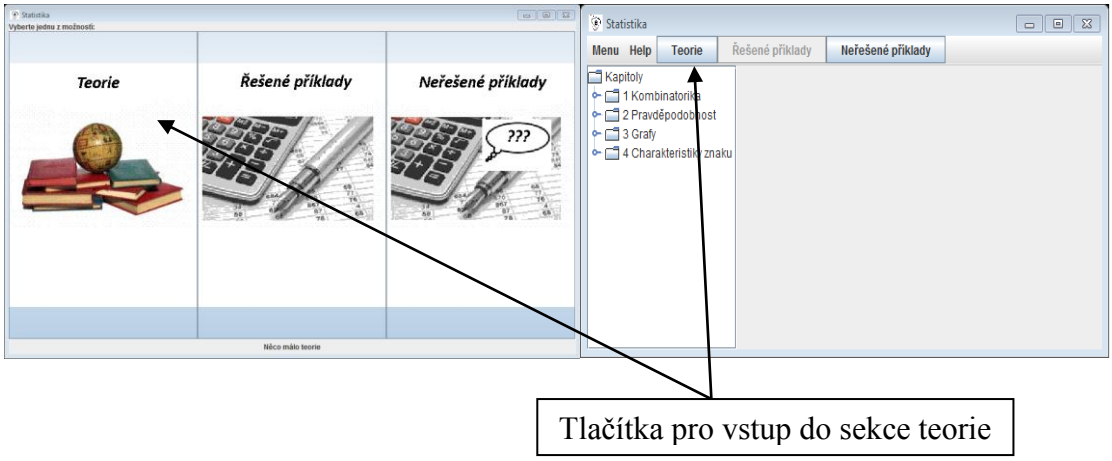
Samotný výukový program je distribuován ve formátu JAR spolu s balíky podporující grafy (JFreeChart), celkem tedy tři JAR balíky jsou k dispozici ke stažení na http://home.zcu.cz/~harmady nebo na CD u této práce. Tyto souboru musí být umístěny v jedné složce.

Pokud je nainstalované JRE a soubory aplikace nahrány v příslušné složce, pak program lze spustit otevřením souboru Statistika.jar.

6.2 Sekce teorie

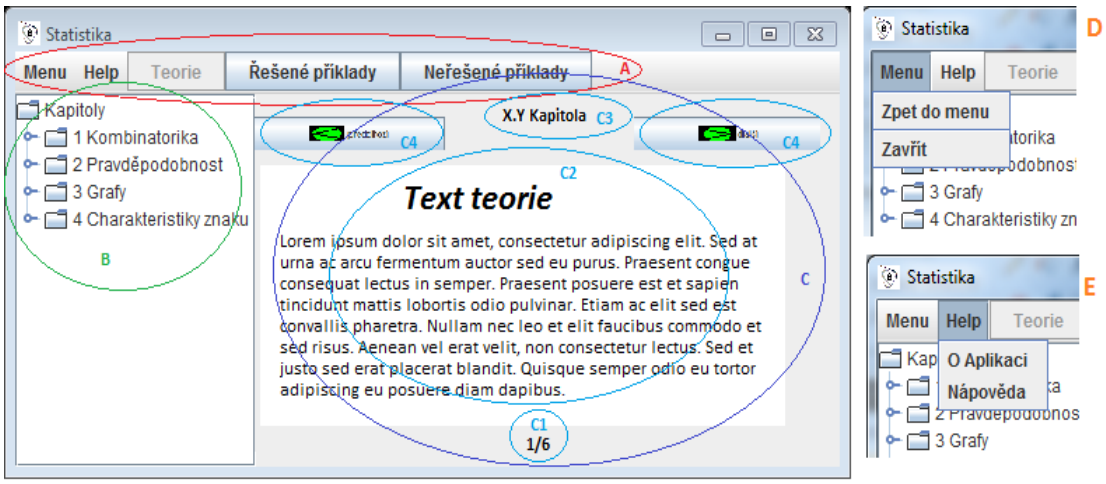
Do této sekce je možné vstoupit dvěma způsoby. První způsob je z úvodního menu, kliknutím na tlačítko s nápisem teorie (Obr. 5 levá polovina). Druhý je ze sekce příkladů, kliknutí na položku teorie v menu (Obr. 5 pravá polovina).

Obrázek 5 : Způsoby vstupu do sekce teorie



V samotné sekci teorie máme možnost v menu na horní liště vrátit se zpět do úvodního menu, ukončit program, vypsat si nápovědu nebo informace o aplikaci a také přepnout do jiných sekci (Obr. 6).

Obrázek 6 : vzhled a komponenty sekce teorie



Zdroj: vlastní zpracování, 2012

Horizontální menu (Obr. 6 písmeno A) nabízí uživateli několik možností. Slouží jako navigační menu. Při kliknutí na položku „menu“ se zobrazí dvě možnosti (Obr. 6 D). První je „Zpět do menu“ pro návrat do hlavní nabídky na úvodní obrazovku. Druhou možností je položka „Zavřít“, která ukončí program. Další možností horizontálního menu je „Help“ (Obr. 6 E). Kliknutím na „O Aplikaci“ se ve zvláštním okně zobrazí informace o aplikaci. Zvolením možnosti „Nápověda“ se také ve zvláštním okně objeví

informace, které by měly pomoci uživateli orientovat se v sekci teorie. Další tlačítka na obrázku 6 v oblasti A slouží k přepnutí aplikace do jiné sekce.

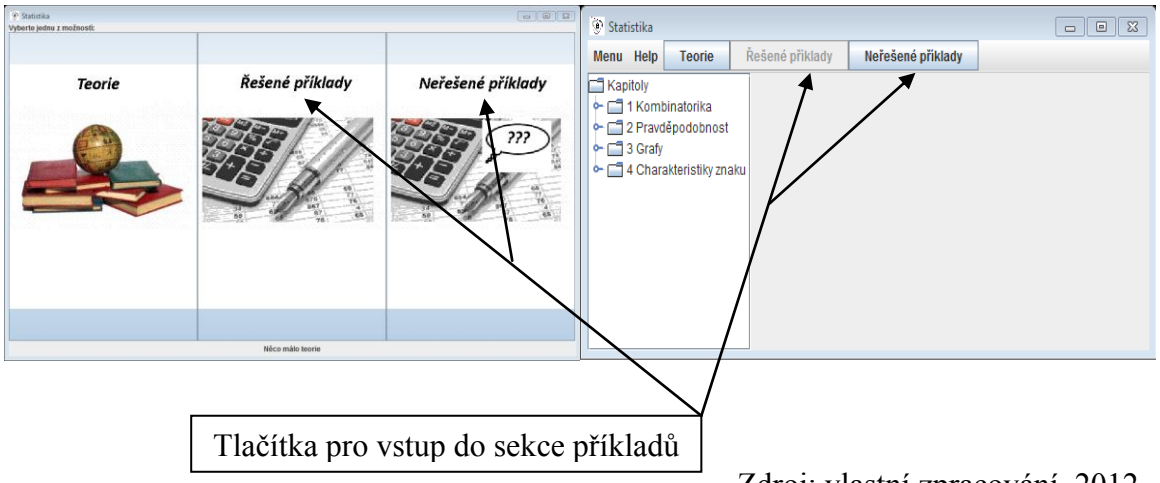
Sekce B na obrázku číslo 6 je strom kapitol, vertikální menu. Zde je možné vybrat si kapitolu, kterou chce uživatel studovat. Po kliknutí na jednu ze čtyř kapitol se vybraná kapitola rozbalí, otevře se první podkapitola, která je „úvod“. Následně může uživatel vybrat jinou podkapitolu nebo změnit kapitolu.

V sekci C (Obr. 6) se zobrazuje samotný text teorie (Obr. 6 C2) a ovládací prvky k tomuto textu. K přepnutí na další slajd vybrané kapitoly slouží tlačítka (C4) umístěná nad textem. Na jakém slajdu a kolik slajdů podkapitola celkem obsahuje, se uživatel dozví z výpisu označeného C1. Další výpis informativního charakteru se nachází mezi ovládacími tlačítky (označen C3). Výpis obsahuje název vybrané podkapitoly.

6.3 Sekce příklady

V aplikaci jsou tyto sekce dvě. Jedna obsahuje řešené příklady a druhá neřešené. Vstoupit do těchto sekcí lze, stejně jako do sekce teorie, dvěma způsoby (Obrázek 7).

Obrázek 7 : Způsoby vstupu do sekcí příkladů



Zdroj: vlastní zpracování, 2012

Po vstupu do sekce příkladů má uživatel podobné možnosti jako v sekci teorie. Horizontální menu zůstává stejné, liší se pouze obsah textu, který se vypíše po kliknutí na možnost „Help -> Nápověda“. Také vertikální menu je na stejné pozici jako u teorie. Kapitoly v něm jsou také stejné, ovšem odlišnost je v listech tohoto stromu (Obr. 8).

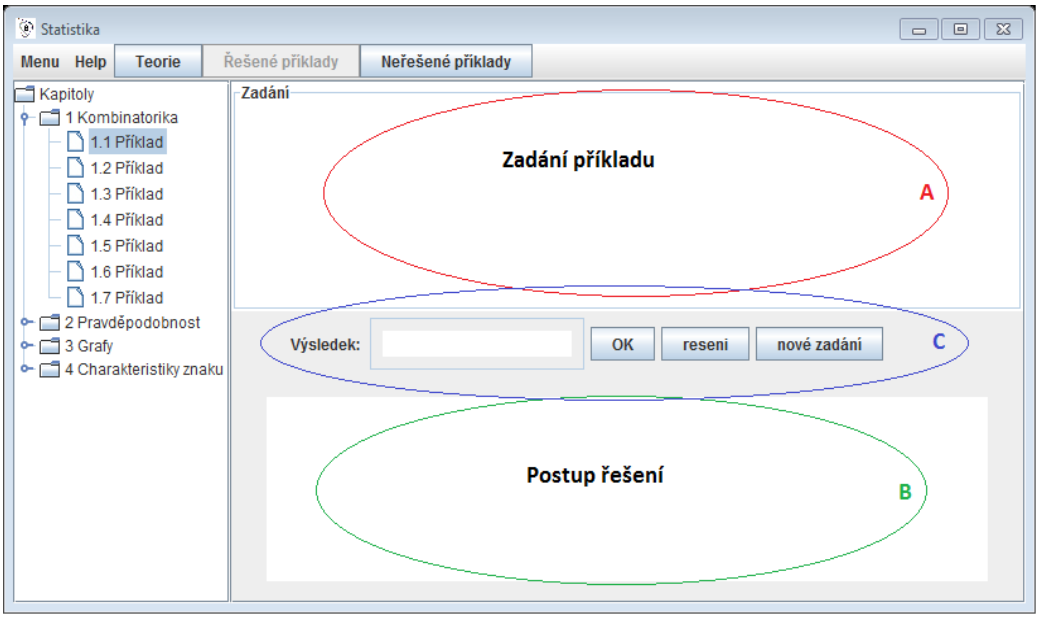
V sekci teorie se po rozkliknutí kapitoly zobrazily podkapitoly a zobrazila se úvodní podkapitola, v případě sekce příkladů se místo podkapitol zobrazí jednotlivé příklady a v hlavní části okna napravo od stromu se zobrazí náhodně vygenerovaný příklad z vybrané kapitoly.

Obrázek 8 : Rozdíly mezi vertikálním menu v sekci teorie a sekci příklady

Zdroj: vlastní zpracování, 2012

Tím nejdůležitějším je část okna pro zobrazení zadání, řešení a ovládacích prvků.

Obrázek 9 : Zobracení příkladu a jeho ovládacích komponent



Zdroj: vlastní zpracování, 2012

Na obrázku číslo 9 v oblasti A je vypsáno zadání příkladu, které se zobrazí po vybrání kapitoly nebo konkrétního příkladu. V oblasti C na stejném obrázku se nacházejí ovládací prvky. Nalevo je pole pro vložení výsledku příkladu. Tento výsledek se odešle tlačítkem s označením „OK“. Dalším tlačítkem je „řešení“, po jeho klinutí se v oblasti B zobrazí postup řešení. V sekci neřešených příkladů je toto tlačítko nefunkční. Posledním prvkem je tlačítko „nové zadání“, po jehož stisku se vygeneruje nové zadání příkladu.

6.4 Přidání a editace slajdů do sekce teorie

Aby obrázek s teorií byl správně přidán, je nejprve nutné jen správně vytvořit. Obrázek musí být ve formátu GIF, a nazván číslem od hodnoty nula. Název obrázku také určuje pořadí, v jakém budou obrázky zobrazeny. Doporučená velikost obrázku 722 x 397 px, pokud by jeho velikost byla jiná, program by ji upravil a obsah se stal rozmazaným.

Pokud je obrázek vytvořen, je nutné vložit ho do JAR souboru. Do správného umístění se dostaneme následujícím způsobem. Pravým tlačítkem myši kliknout na soubor Statistika.jar. Zvolit možnost „otevřít v programu“ a vybrat program pro práci s archívy (WinRar, 7zip). V tomto programu otevřít složku „teorie“ a následně správnou složku podkapitoly. Složky jsou identifikovány následujícími čísly a písmenem „T“: 11T – Úvod do kombinatoriky, 12T - Variace, 13T – Permutace, 14T – Kombinace, 21T – Úvod do pravděpodobnosti, 22T – Pravidla počítání pravděpodobností, 31T – Úvod do grafů, 32T – Typy grafů, 41T – Úvod do charakteristiky znaku, 42T – Charakteristiky polohy, 43T – Charakteristiky variability. Do jedné příslušné složky obrázek vkopírovat a potvrdit modifikaci souboru. Nakonec je ještě nutné upravit ve složce, do které byl obrázek přidán, soubor pocetListu.txt na korektní hodnotu. Soubor pocetListu.txt otevřít v textovém editoru, hodnotu přepsat a změny uložit. Při dalším spuštění programu bude již slajd v dané podkapitole přidán.

Pro editaci slajdů je nutné obrázek extrahovat do Vašeho PC, tam ho upravit a vložit ho zpět do příslušné složky.

6.5 Přidání a editace příkladů

Přidávat a upravovat zadání příkladů je možné v souboru priklady.txt, který je umístěn ve složce „priklady“ v archívu Statistika.jar. Tento soubor stačí otevřít v textovém editoru a je možné provádět úpravy. Každý příklad je na jedné řádce souboru. První číslo označuje číslo kapitoly. Kapitola 1 je kombinatorika, číslo 2 označuje kapitolu pravděpodobnost, číslo 3 grafy a číslo 4 charakteristiky znaku. Druhé číslo je identifikační číslo příkladu. Toto číslo je nutné zadávat tak, aby bylo v dané kapitole jedinečné. Třetí hodnota je číslo řešení neboli název obrázku s řešením, pokud řešení nemá, pak hodnotu nastavte na nulu. A čtvrté číslo označuje postup řešení. Podle tohoto čísla bude program zadaný příklad počítat. Postupy, které jsou zatím implementovány, jsou popsány v kapitole 4.2 Vrstva logiky a výpočtů. Za tímto posledním identifikátorem následuje mezera a za ní text příkladu. Hodnoty, které mají být dogenerovány, je nutné zapsat ve složených závorkách (viz 4.2 Vrstva logiky a výpočtů) a odřádkování je možné pomocí /n.

Po úpravě souboru je potřeba změny uložit a potvrdit modifikaci archívu. Při dalším spuštění programu již bude program pracovat i s novým nebo změněným příkladem.

6.6 Přidání řešení k příkladům

V souboru priklady.txt je u každého příkladu atribut číslo řešení. Pokud se pak v archívu Statistika.jar ve složce „reseni“ nachází obrázek s názvem shodným s číslem řešení, pak si uživatel může toto řešení zobrazit u daného příkladu.

Obrázek s řešení by měl mít velikost 700 x 270 px, jinak bude rozmazaný. Tento obrázek po vložení do výše zmíněné složky bude při dalším spuštění k dispozici.

Postup editace je stejný jako v případě teorie.

7 Návrh dalšího vývoje programu

Další vývoj programu by se mohl ubírat dvěma směry. První by znamenal pokračovat ve vylepšování programu jako desktopové aplikace. Druhým směrem by bylo vytvoření webové aplikace pro podporu výuky statistiky.

7.1 Desktopová aplikace

Současnou aplikaci by bylo možné rozšířit o další funkcionality. Jednou takovou by mohlo být generování testu. Program by vygeneroval určitý počet příkladů a po odeslání výsledků je vyhodnotil. Dalším vylepšením by mohlo být vytvoření třídy reprezentující tabulku jako alternativu ke grafům. Také určitě přidat další příklady a jejich postupy řešení.

7.2 Webová aplikace

Vytvoření webové aplikace by znamenalo kompletní předělení uživatelského rozhraní a také úpravy tříd přistupujících k datům. Webová aplikace totiž často spolupracuje s databázovým serverem, ukládání dat do souboru není příliš vhodné. Pokud by se aplikace vyvíjela tímto směrem, bylo by nutné změnit platformu. Java 6 SE (Standard Edition) by již byla nedostačující a muselo by se přejít na platformu Java EE (Enterprise Edition), která je pro takové účely vytvořena.

Applet je již „mrtvá“ technologie, takže třídy současné logické vrstvy by bylo nutné předělat na servlety. Rozhraní zase vytvořit pomocí například Java Server Pages (JSP), což je technologie pro dynamické generování webových stránek.

Webovou aplikace s databází by bylo možné rozšířit o evidenci uživatelů, tedy žáků a učitelů. Výsledky žáků by se uchovávaly v databázi a učitel by si je mohl zobrazit. Také by mohlo být připojeno fórum, kde by si žáci vyměňovali své zkušenosti se statistikou.

8 Závěr

V této bakalářské práci, zejména v její první části, je popsáno, co vlastně je výukový program a co byl měl výukový program na podporu výuky statistiky na střední škole obsahovat. Obsah tohoto výukové programu vychází ze školského vzdělávacího programu a obsahu učiva na Sportovní a podnikatelské škole v Plzni. Dále jsou v této části popsány technologie, které byly použity pro tvorbu tohoto programu. Jako programovací jazyk byl zvolen jazyk Java, hlavní důvody této volby byla přenositelnost, jednoduchost a možnost dalšího vývoje programu směrem k webové aplikaci.

Druhá, větší část práce, obsahuje popis samotné aplikace. Nejprve je popsána struktura aplikace, která je doprovázena UML diagramem v příloze této práce. Dále je ukázán vzhled a fukce jednotlivých komponent v prezentační vrstvě aplikace. Podobný způsobem jako prezentační vrstva, jsou popsány i vrstvy datová a logické. V těchto částech práce jsou také ukázky kódu jednotlivých algoritmů pro výpočty příkladů, nebo pro práci s daty. Kód je také podrobně komentován.

Další důležitou částí práce je zjednodušená uživatelská dokumentace, díky níž by uživatel měl být schopen program instalovat, spustit a následně obsluhovat. Také jsou zde obsaženy návody pro přidání teorie, příkladů a řešení k těmto příkladům.

Výstupem této práce je výukový program na podporu výuky statistiky na středních školách. Pro jeho implementaci byl zvolen jazyk Java, a to na základě přenositelnosti programů v něm napsaných a objektové orientaci.

Program je distribuován jako desktopová aplikace v archívu JAR. Je proto možné aby žáci program využívali bez ohledu na připojení k síti. Nezávislost na připojení je zcela jistě jednou z předností aplikace, ovšem při dalším vývoji aplikace (desktopové) by bylo na místě uvažovat nad umožnění aktualizace dat uvnitř programu. Tedy off-line aplikaci, jakou je program nyní, přeprogramovat na off-line aplikaci s on-line podporou.

Pokud se podíváme na aplikaci z pohledu koncového uživatele, tedy žáka, a ne programátora, pak má tato aplikace oproti knižní podobě řadu výhod. V dnešní době jsou počítače fenoménem, který je pro žáka střední školy lákavější než kniha. Pro žáka je snažší program stáhnout než sehnat knihu a následná práce s ním je pro něj zajímavější. Příklady totiž nejsou statické, ale hodnoty jsou do jednotlivých zadání

dynamicky dogenerovávané. Také orienatce v programu je snažší než v knize, není nutné neustále přelistovávat, zobrazí se vždy vybraná kapitola nebo příklad.

Výhodou také je snadná aktualizace programu. A to i přes absenci on-line aktualizací. Do aplikace učitel (nebo jiný uživatel) může přidávat libovolné množství teorie nebo příkladů. Takto sice může dojít k verzování programu, ovšem ani to rozhodně není na škodu. Každý učitel si může defaultní verzi upravit tak, aby obsah ještě lépe vyhovoval probírané látce na konkrétní škole, v konkrétní třídě. Takovou „novou“ verzi je vhodné v jejím názvu označit, aby nenastali problémy v identifikaci jednotlivých verzí.

V současné verzi je program možno implementovat do výuky nebo jej doporučit žákům k samostudiu, ovšem program čeká na dopnění dalších algoritmů pro řešení příkladů a také na přidání dalších zadání příkladů.

9 Seznam obrázků

10 Seznam použité literatury

BRŮHA, L. *Java: hotová řešení*. 1. vyd. Brno: Computer Press, 2006. ISBN 80-251-0072-3.

CALDA, E., DUPAČ, V. *Matematika pro gymnázia.* 1. vyd. Praha: Jednota českých matematiků a fyziků, 1993. ISBN 80-7015-444-6.

DOSTÁL, J. Výukový software a didaktické hry – nástroje moderního vzdělávání. [online elektronický časopis] *Journal of Technology and Information Education,* 2009, roč. 1, č. 1, s. 24 – 28, ISSN 1803-6805. [cit. 2012-23-04] Dostupné na www: <http://www.jtie.upol.cz/clanky_1_2009/dostal.pdf>

HEROUT, P. *Java: grafické uživatelské prostředí a čeština*. 1. vyd. České Budějovice: Kopp, 2006. ISBN 80-85931-35-4.

HEROUT, P. *Učebnice jazyka Java*. 3. vyd. České Budějovice: Kopp, 2007. ISBN 978-80-7232-323-4.

HINDLS, R., HRONOVÁ, S., SEGER, J., FISCHER, J. *Statistika pro ekonomy*. 8. vyd. Praha: Professional Publishing, 2007. ISBN 97-80-86946-43-6.

Oracle: Java 6 SE API specification. [online] Redwood city: Oracle Corporation, 2012, Aktualizace 30.11.2011, [cit. 2012-01-16] Dostupné na www:< http://docs.oracle.com/javase/6/docs/api/>

Oracle: The Java tutotial. [online] Redwood city: Oracle Corporation, 2012, Aktualizace 3.2.2012, [cit. 2012-04-16] Dostupné na www:<http://docs.oracle.com/javase/tutorial/>

PECINOVSKÝ, R. *Myslíme objektově v jazyku Java*. 1. vyd. Praha: Grada Publishing, 2004. ISBN 80-247-0941-4.

Sportovní a podnikatelská střední škola Plzeň. [online] Plzeň: Sportovní a podnikatelská střední škola Plzeň, 2012, Aktualizace 11.4.2012, [cit. 2012-04-16] Dostupné na www:<http://www.sapss-plzen.cz/>

ZAKHOUR, S., a kol. *Java 6 Výukový kurz*. 1. vyd. Brno: Computer Press, 2007. ISBN 978-80-251-1575-6.

11 Seznam příloh

Příloha A : Zdrojové kódy aplikace

Příloha B : UML diagram tříd

Příloha C : Abstrakt

Příloha D : Abstract

Příloha A : **Zdrojové kódy aplikace**

Datová vrstva:

Třída DAO.java:

```
package data;

import java.awt.Image;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Random;

import javax.swing.ImageIcon;

public class DAO {
	public DAO() {
	}
	public ArrayList<Image> nactiTeorii(String vybranaKapitola) {

		int[] p = prevodNazvu(vybranaKapitola);
		String cisloKapitoly = p[0] + ""+ p[1];

		ArrayList<Image> pole = new ArrayList<Image>();
		try {
			InputStream inS = getClass()
					.getResourceAsStream(
							"/teorie/" + cisloKapitoly
							+ "T/pocetListu.txt");
			BufferedReader bfr = new BufferedReader(new
			InputStreamReader(inS));
			int pocetListu = Integer.parseInt(bfr.readLine());
			for (int i = 0; i < pocetListu; i++) {
				ImageIcon imI = new
				ImageIcon(getClass().getResource("/teorie/" +
				cisloKapitoly + "T/" + i + ".gif"));
				Image obr = imI.getImage();

				pole.add(obr);
			}
		}catch(Exception e) {
			e.getStackTrace();
		}
		return pole;
	}
	public String[] nactiZadani(String vybranaKapitola,int sResenim) {

		String[] pole = new String[4];
```

```
	try {
		InputStream inS =
		getClass().getResourceAsStream("/priklady/priklady.txt");
		BufferedReader bfr = new BufferedReader(new
		InputStreamReader(inS));
		int maximalniPocetPrikladu = zjistiPocetPrikladu(bfr);
		int[] p = prevodNazvu(vybranaKapitola);
		String[] polePriklad = nactiPriklad(p[0],p[1],
		maximalniPocetPrikladu, bfr, sResenim);


		pole[0] = "";
		for (int i = 4; i < polePriklad.length; i++) {
			pole[0] = pole[0] + polePriklad[i] + " ";
		}
		pole[1] = polePriklad[2];
		pole[2] = polePriklad[1];
		pole[3] = polePriklad[3];
		pole[0] = pole[0].replace("/n", "\n");

	}catch (Exception e) {
		e.getStackTrace();
	}
	return pole;
}
public int[] prevodNazvu(String nazev) {
	int[] pole = new int[2];
	int kapitola;

	String[] pom = nazev.split(" ");
	try {
		kapitola = Integer.parseInt(pom[0]);
		pole[1] = 0;
	} catch (Exception e) {
		pom = pom[0].split("\\.");
		kapitola = Integer.parseInt(pom[0]);
		pole [1] = Integer.parseInt(pom[1]);
	}
	pole[0] = kapitola;
	return pole;
}

private String[] nactiPriklad(int kapitola, int priklad, int max,
BufferedReader bfr, int sResenim) {
	try {
		int cisloZadani;
		if(priklad == 0) {
			Random r = new Random();
			cisloZadani = r.nextInt(max) +1;
		}
		else {
```

```
                        cisloZadani = priklad;
                }
                while(true) {

                        InputStream inS = getClass()
                        .getResourceAsStream("/priklady/priklady.txt");
                        bfr = new BufferedReader(neInputStreamReader(inS));

                        bfr.readLine();
                        bfr.readLine();

                        int i = cisloZadani;
                        while (i > 0) {

                                try {
                                        String radka = bfr.readLine();
                                        String[] pole = radka.split(" ");

                                        int nactenaKapitola =
                                        Integer.parseInt(pole[0]);
                                        int reseni =
                                        Integer.parseInt(pole[2]);
                                        if(nactenaKapitola ==kapitola &&
                                        ((reseni >= sResenim) || (reseni == -
                                        1))) {
                                                --i;
                                        }
                                        if(i == 0) {
                                                return pole;
                                        }
                                }catch(Exception e) {
                                        cisloZadani = cisloZadani /2;
                                        break;
                                }
                        }
                }
        }catch (Exception e) {
                e.getStackTrace();
        }
        return null;
}

private int zjistiPocetPrikladu(BufferedReader bfr) {
        String radka = "";
        try {
                radka = bfr.readLine();
        } catch (IOException e) {
                e.printStackTrace();
        }
        String[] pole = radka.split(" ");
        int pocet = Integer.parseInt(pole[2]);
```

```
		return pocet;
	}

	public Image getReseni(String typ, String cisloReseni) {
		Image obr = null;
		try {

		ImageIcon imI = new ImageIcon(getClass().getResource("/"+typ+"/"
		+ cisloReseni + ".gif"));
		obr = imI.getImage();

		}catch (Exception e) {
			e.getStackTrace();
		}

		return obr;
	}
	public int getPocetPrikladuVKapitole(int kapitola, int status) {
		BufferedReader bfr;
		InputStream inS = getClass()
				.getResourceAsStream(
					"/priklady/priklady.txt");
		bfr = new BufferedReader(new InputStreamReader(inS));

		try {
			bfr.readLine();
			bfr.readLine();
		} catch (IOException e1) {
			e1.printStackTrace();
		}

		int pocetPrikladu = 0;
		while(true) {
			try {
				String radka = bfr.readLine();
				String[] pole = radka.split(" ");

				int nactenaKapitola = Integer.parseInt(pole[0]);
				int reseni = Integer.parseInt(pole[2]);
				if(kapitola == nactenaKapitola && status == 2) {
					++pocetPrikladu;
				}
				else if(kapitola == nactenaKapitola && reseni != 0)
{
					++pocetPrikladu;
				}
			}catch(Exception e) {
				break;
			}
		}
```

```
	return pocetPrikladu;
	}
}
```

Logická vrstva:

Třída Ridici.java

```
package logika;

import java.awt.Image;
import java.text.MessageFormat;
import java.util.ArrayList;
import java.util.Random;
import data.*;
import rozhrani.*;

public class Ridici {
	VystupniOkno okno;
	PracovniObrazovka prObr;
	public DAO priklad;
	public int aktualniStatus; //0-teorie, 1- resene, 2-neresene
	public AktualniPriklad prikladAktualni;
	public int aktualniSlajd;

	Ridici() {
		okno = new VystupniOkno(this);
		aktualniSlajd = 0;
		priklad = new DAO();
	}

	public void provedAkci(int cisloAkce, String vybranaKapitola) {
		switch (cisloAkce) {
		case 1:
			aktualniStatus = 0;
			vlozPracovniObrazovku();
			break;
		case 2
			aktualniStatus = 1;
			vlozPracovniObrazovku();
			break;
		case 3:
			aktualniStatus = 2;
			vlozPracovniObrazovku();
			break;
		case 4:
			switch (aktualniStatus) {
			case 0:
				vlozTeorii(vybranaKapitola);
```

```
                        break;
                    case 1:
                        vlozResenePriklady(vybranaKapitola);
                        break;

                    case 2:
                        vlozNeresenePriklady(vybranaKapitola);
                        break;
                    default:
                        break;
                    }
                    break;

            case 5:
                    if(prikladAktualni.getCisloReseni() == -1)  {
                        ((PrikladyPanel)
prObr.getObsah()).vlozGraf(prikladAktualni.getCisloZadani(),
prikladAktualni.getParam());
                        aktualizujVystup();

                    }
                    else if(prikladAktualni.getCisloReseni() > 0 &&
aktualniStatus == 1) {                        ((PrikladyPanel)
prObr.getObsah()).vlozReseni(priklad.getReseni("reseni","" +
prikladAktualni.getCisloReseni()));
                        okno.vlozPracovniObrazovku(prObr);
                    }
                    else {
                        //pokud nema obrazek s resenim
                    }
                    break;
            case 6:
                    if(prikladAktualni.getCisloReseni() == -1) {
                        provedAkci(5, vybranaKapitola);
                        break;
                    }
                    Resitel r = new Resitel(prikladAktualni);

                    double vysledek = r.getVysledek();
                    double vysledekUzivatele =
Double.parseDouble(vybranaKapitola);



                    if(vysledek == vysledekUzivatele) {
                        ((PrikladyPanel)
prObr.getObsah()).setVypisOSpravnosti("spravne");
                    }
                    else {
                        ((PrikladyPanel)
prObr.getObsah()).setVypisOSpravnosti("spatne");
```

```
                }
                break;
            case 7: //napoveda
                new Napoveda(aktualniStatus);

            default:
                break;
            }
        }
        private void vlozTeorii(String vybranaKapitola) {
            aktualniSlajd = 0;
            ArrayList<Image> poleTeorie = new ArrayList<Image>();
            poleTeorie = priklad.nactiTeorii(vybranaKapitola);
            TeoriePanel t;
            try {
                prObr.smazObsah();
                t = new TeoriePanel(poleTeorie, vybranaKapitola, this);
            }catch (Exception e) {
                t = new TeoriePanel(poleTeorie, vybranaKapitola, this);
            }
            prObr.vlozPanel(t);
            okno.vlozPracovniObrazovku(prObr);
        }

        private void vlozResenePriklady(String vybranaKapitola) {
            String[] polePriklad = priklad.nactiZadani(vybranaKapitola, 1);
        prikladAktualni = new
AktualniPriklad(Integer.parseInt(polePriklad[2]),
priklad.prevodNazvu(vybranaKapitola)[0], Integer.parseInt(polePriklad[1]));

        prikladAktualni.setPostupReseni(Integer.parseInt(polePriklad[3]));
            try {
                prObr.smazObsah();
                boolean graf = false;
                if(prikladAktualni.getCisloReseni() == -1) {
                    graf = true;
                }
                PrikladyPanel res = new
PrikladyPanel(dogenerujCisla(polePriklad[0], false), this, aktualniStatus,
graf);
                res.vytvorTlacitkoNovy(vybranaKapitola, true); //true =
resene
                prObr.vlozPanel(res);

                okno.vlozPracovniObrazovku(prObr);
            }catch (Exception e) {
                boolean graf = false;
                if(prikladAktualni.getCisloReseni() == -1) {
                    graf = true;
                }
```

```
                  PrikladyPanel res = new
PrikladyPanel(dogenerujCisla(polePriklad[0], false), this, aktualniStatus,
graf);
                  res.vytvorTlacitkoNovy(vybranaKapitola, true); //true =
resene
                  prObr.vlozPanel(res);
                  okno.vlozPracovniObrazovku(prObr);
            }
      }
      private void vlozNeresenePriklady(String vybranaKapitola) {
            String[] polePriklad = priklad.nactiZadani(vybranaKapitola, 0);
                        prikladAktualni = new
      AktualniPriklad(Integer.parseInt(polePriklad[2]),
      priklad.prevodNazvu(vybranaKapitola)[0],
      Integer.parseInt(polePriklad[1]));

      prikladAktualni.setPostupReseni(Integer.parseInt(polePriklad[3]));
            try {
                  prObr.smazObsah();
                  boolean graf = false;
                  if(prikladAktualni.getCisloReseni() == -1) {
                        graf = true;
                  }
                  PrikladyPanel res = new
PrikladyPanel(dogenerujCisla(polePriklad[0], true),                    this,
aktualniStatus, graf);
                  res.vytvorTlacitkoNovy(vybranaKapitola, false); //true =
resene
                  prObr.vlozPanel(res);

                  okno.vlozPracovniObrazovku(prObr);
            }catch (Exception e) {
                  boolean graf = false;
                  if(prikladAktualni.getCisloReseni() == -1) {
                        graf = true;
                  }
                  PrikladyPanel res = new
                  PrikladyPanel(dogenerujCisla(polePriklad[0], true), this,
                  aktualniStatus, graf);
                  res.vytvorTlacitkoNovy(vybranaKapitola, false); //true =
resene
                  prObr.vlozPanel(res);
                  okno.vlozPracovniObrazovku(prObr);
            }
      }
      private void vlozPracovniObrazovku() {
            prObr = new PracovniObrazovka(okno);
            okno.vlozPracovniObrazovku(prObr);

      }
      public void aktualizujVystup() {
```

```
			okno.validate();
	}
	public VystupniOkno getVystOkno() {
		return okno;
	}
	private String dogenerujCisla(String puvText, boolean nahodost) {
		Double[] params = getArgumenty(nahodost);
		prikladAktualni.setParam(params);
		Object[] o = params;
		String message = MessageFormat.format(puvText, o);

		return message;
	}
	private Double[] getArgumenty(boolean nahodne) {
		Double[] argsDouble;
		if (nahodne == true) {
			argsDouble = new Double[20];
			Random r = new Random();
			for (int i = 0; i < 5; i++) {
				double p = r.nextInt(100);
				p = p / 100;
				argsDouble[i] = p;
			}
			for (int i = 5; i < 10; i++) {
				double p = r.nextInt(100);

				if (p < 10) {
					p = p + 20;
				}
				argsDouble[i] = p;
			}
			for (int i = 10; i < 15; i++) {
				double p = r.nextInt(20);
				if (p < 3) {
					p = p + 3;
				}
				argsDouble[i] = p;
			}
			for (int i = 15; i < 20; i++) {
				double p = r.nextInt(10);
				if (p < 3) {
					p = p + 3;
				}
				argsDouble[i] = p;
			}
		} else {
			argsDouble = new Double[20];
			argsDouble[0] = 0.1;
			argsDouble[1] = 0.2;
			argsDouble[2] = 0.3;
			argsDouble[3] = 0.4;
```

```
                argsDouble[4] = 0.8;
                argsDouble[5] = 15.0;
                argsDouble[6] = 38.0;
                argsDouble[7] = 46.0;
                argsDouble[8] = 62.0;
                argsDouble[9] = 93.0;
                argsDouble[10] = 4.0;
                argsDouble[11] = 8.0;
                argsDouble[12] = 15.0;
                argsDouble[13] = 17.0;
                argsDouble[14] = 18.0;
                argsDouble[15] = 4.0;
                argsDouble[16] = 5.0;
                argsDouble[17] = 6.0;
                argsDouble[18] = 7.0;
                argsDouble[19] = 9.0;
        }
        return argsDouble;
    }

    public static void main(String[] args) {
        new Ridici();
    }
}
```

Resitel.java:

```
package logika;

public class Resitel {
    double vysledek;
    AktualniPriklad priklad;

    public Resitel(AktualniPriklad p) {
        this.priklad = p;

    }



    private double aritmetickyPrumer(int odkud, int n) {
        double v = 0;
        for (int i = odkud; i < n + odkud; i++) {
            v = v + priklad.getParam()[i];
        }
        v = v / n;

        return v;
    }


    private double variacniRozpeti(int odkud, int n) {
        double min = Integer.MAX_VALUE;
        for (int i = odkud; i < n + odkud; i++) {
            min = Math.min(min, priklad.getParam()[i]);
```

```
		}
		double max = Integer.MIN_VALUE;
		for (int i = odkud; i < n + odkud; i++) {
			max = Math.max(max, priklad.getParam()[i]);
		}
		double reseni = max - min;

		return reseni;
	}

	private double rozptyl(int odkud, int n) {
		double[] hodnoty = new double[n];
		for (int i = 0; i < hodnoty.length; i++) {
			hodnoty[i] = priklad.getParam()[i + odkud] * 1000;
		}
		double prumer = aritmetickyPrumer(odkud, n) * 1000;

		double rozptyl = 0;
		for (int i = 0; i < hodnoty.length; i++) {
			rozptyl += ((hodnoty[i] - prumer) * (hodnoty[i] -
prumer));
		}
		double reseni = rozptyl / n;
		return reseni;
	}

	private double modus(int odkud, int n) {
		double reseni = priklad.getParam()[odkud];
		int pocetMax = 1;
		int pocet= 1;

		for (int i = odkud; i < (odkud+n); i++) {
			double a = priklad.getParam()[i];
			if(a == priklad.getParam()[12]) {
				++pocet;
			}
			for (int j = i+1; j < (odkud+n); j++) {
				if(a == priklad.getParam()[j]) {
					++pocet;
				}
			}
			if(pocet > pocetMax) {
				pocetMax = pocet;
				reseni = a;
			}
			else if(pocet == pocetMax) {
				reseni = Math.max(reseni, a);
			}
			pocet = 1;
		}
		return reseni;
	}



	private double kombinaceBezOpakovani(double k, double n) {
		double citatel = faktorial(n);
		double jmenovatel1 = faktorial(n-k);
		double jmenovatel2 = faktorial(k);
```

```
		double reseni = citatel / (jmenovatel1 * jmenovatel2);
		return reseni;
	}
	private double kombinaceSOpakovanim(double k, double n) {
		double citatel = faktorial(k + n - 1);

		double jmenovatel1 = faktorial(n-1);
		double jmenovatel2 = faktorial(k);

		double reseni = citatel / (jmenovatel1 * jmenovatel2);
		return reseni;
	}

	private double faktorial(double n) {
		double reseni = 1;
		for (double i = n; i > 0; i = i - 1) {
			reseni *= i;
		}
		return reseni;
	}
	private double variaceBezOpakovani(double k, double n) {
		double citatel = faktorial(n);
		double jmenovatel = faktorial(n-k);

		return (citatel / jmenovatel);
	}
	private double variaceSOpakovanim(double k, double n) {

		double reseni = Math.pow(n, k);
		return reseni;
	}

	private double odmocnina(double x) {
		return Math.sqrt(x);
	}
	public double getVysledek() {
		switch (priklad.getPostupReseni()) {
		case 0:
			break;
		case 1:
			vysledek = aritmetickyPrumer(5, 5);
			break;
		case 2:

			break;
		case 3:
			vysledek = variacniRozpeti(10, 5) * 1000;
			break;
		case 4:
			vysledek = rozptyl(10, 5);
			break;
		case 5:
			vysledek = kombinaceBezOpakovani(2,
priklad.getParam()[17]);
			break;
		case 6:
```

```
                vysledek = kombinaceBezOpakovani(3,
                priklad.getParam()[12]) * kombinaceBezOpakovani(2,
                priklad.getParam()[17]) * 2;
                break;
            case 7:
                vysledek = kombinaceSOpakovanim(priklad.getParam()[10],
                priklad.getParam()[15]);
                break;
            case 8:
                vysledek = faktorial(priklad.getParam()[17]);
                break;
            case 9:
                vysledek = faktorial(priklad.getParam()[8]) /
                (faktorial(priklad.getParam()[8] - priklad.getParam()[17])
                * faktorial(priklad.getParam()[17]));
                break;
            case 10:
                vysledek = faktorial(priklad.getParam()[15]) *
                faktorial(priklad.getParam()[16])
                *faktorial(priklad.getParam()[17]);
                break;
            case 11:
                vysledek = faktorial(priklad.getParam()[15]) *
                faktorial(priklad.getParam()[16])
                *faktorial(priklad.getParam()[17]) * faktorial(3) ;
                break;
            case 12:
                vysledek = priklad.getParam()[0] * (1 -
                priklad.getParam()[1]) * (1 - priklad.getParam()[2]);
                break;
            case 13:
                vysledek = 1 - (1 - priklad.getParam()[0]) * (1 -
                priklad.getParam()[1]) * (1 - priklad.getParam()[2]);
                break;
            case 14:
                vysledek = variaceBezOpakovani(3, priklad.getParam()[15]);
                break;
            case 15:
                vysledek = variaceSOpakovanim(3, priklad.getParam()[15]);
                break;
            case 16:
                vysledek = odmocnina(rozptyl(15,5))/1000;
                break;
            case 17:
                vysledek = ((aritmetickyPrumer(8, 7) * 7) +
priklad.getParam()[10]) / 8 ;
                break;
            case 18:
                vysledek = modus(11, 9);
                break;
            case 19:
                vysledek = variaceBezOpakovani(2, priklad.getParam()[15]);
                break;
            case 20:
                vysledek = variaceBezOpakovani(4, 5) -
variaceBezOpakovani(3, 4);
                break;

            default:
```

```
                    break;
                }
                vysledek = vysledek * 100;
                vysledek = Math.round(vysledek);
                vysledek = vysledek / 100;
                return vysledek;

        }
}
```

Třída AktualniPrikalad.java:

```
package logika;

public class AktualniPriklad {
        int cisloZadani;
        int cisloKapitoly;
        int cisloReseni;
        double vysledek;
        int postupReseni;
        Double[] parametry;

        public AktualniPriklad(int cisloZadani, int cisloKapitoly, int
cisloReseni) {
                this.cisloZadani = cisloZadani;
                this.cisloKapitoly = cisloKapitoly;
                this.cisloReseni = cisloReseni;
        }
        public int getCisloZadani() {
                return cisloZadani;
        }
        public int getCisloKapitoly() {
                return cisloKapitoly;
        }
        public int getCisloReseni() {
                return cisloReseni;
        }
        public Double[] getParam() {
                return parametry;
        }
        public void setVysledek(double v) {
                this.vysledek = v;
        }
        public void setParam(Double[] p) {
                this.parametry = p;
        }
        public void setPostupReseni(int x) {
                this.postupReseni = x;
        }
        public int getPostupReseni() {
                return postupReseni;
        }
}
```

Vrstva uživatelského rozhraní:

VystupniOkno.java:

```
package rozhrani;

import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JPanel;

import logika.Ridici;

public class VystupniOkno extends JFrame {
	Ridici ridici;

	private static final long serialVersionUID = 1L;

	public VystupniOkno(Ridici ridici) {
		this.ridici = ridici;
		vytvorOkno();
		vlozUvodniData();
	}
	public void akce(int cisloAkce, String vybranaKapitola) {
		ridici.provedAkci(cisloAkce, vybranaKapitola);

	}
	public void vlozPracovniObrazovku(PracovniObrazovka p) {
		aktualizuj(p);
	}
	public void vlozUvodniData() {
		this.getContentPane().add(new UvodniObrazovka(this));
		validate();
		setVisible(true);
	}

	private void vytvorOkno() {
		setSize(975, 600);
		setDefaultCloseOperation(EXIT_ON_CLOSE);
		setTitle("Statistika");
		ImageIcon i = new ImageIcon(getClass().getResource(
	"/img/ikona.gif"));
		setIconImage(i.getImage());
		setLocationRelativeTo(null);
	}
	public void aktualizuj(JPanel x) {
		this.getContentPane().removeAll();
		this.getContentPane().add(x);
		this.validate();
	}
}
```

Třída UvodniObrazovka.java:

```
package rozhrani;
```

```
import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;

import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class UvodniObrazovka extends JPanel {

	JPanel proTacitkaVolbyPN;
	JLabel vypis;
	VystupniOkno okno;

	private static final long serialVersionUID = 1L;

	public UvodniObrazovka(VystupniOkno okno) {
		this.okno = okno;

		this.setLayout(new BorderLayout());
		proTacitkaVolbyPN = new JPanel();
		proTacitkaVolbyPN.setLayout(new GridLayout(1, 3));

		JPanel infoPN = new JPanel();
		vypis = new JLabel();
		vypis.setText(" ");

		JLabel vypis1 = new JLabel();
		vypis1.setText("Vyberte jednu z možností:");
		infoPN.add(vypis);
		this.add(vypis1, BorderLayout.NORTH);
		this.add(infoPN, BorderLayout.SOUTH);
		vytvorTlacitka();
		this.add(proTacitkaVolbyPN, BorderLayout.CENTER);
	}

	private void vytvorTlacitka() {
		JButton teorie = new JButton(new
ImageIcon(getClass().getResource(
				"/img/teorieIk.gif"))); // z
		JButton resene = new JButton(new
ImageIcon(getClass().getResource(
				"/img/ResPrIk.gif")));
		JButton neresene = new JButton(new
ImageIcon(getClass().getResource(
				"/img/NeResPrIk.gif")));
```

```
            teorie.addMouseListener(new TeorieList());
            teorie.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        okno.akce(1, "");
                  }
            });
            resene.addMouseListener(new ReseneList());
            resene.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        okno.akce(2, "");
                  }
            });
            neresene.addMouseListener(new NereseneList());
            neresene.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        okno.akce(3, "");
                  }
            });
            proTacitkaVolbyPN.add(teorie);
            proTacitkaVolbyPN.add(resene);
            proTacitkaVolbyPN.add(neresene);
      }
      private class TeorieList extends MouseAdapter implements MouseListener
{
            public void mouseEntered(MouseEvent e) {
                  vypis.setText("Něco málo teorie");
            }
      }
      private class ReseneList extends MouseAdapter implements MouseListener
{
            public void mouseEntered(MouseEvent e) {
                  vypis.setText("Ukazka jek resit jednotlive priklady");
            }
      }
      private class NereseneList extends MouseAdapter implements
MouseListener {
            public void mouseEntered(MouseEvent e) {
                  vypis.setText("Vyzkoušejte si zda dokážete správně řešit
příklady");
            }
      }
}
```

PracovniObrazovka.java

```
package rozhrani;

import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.event.ActionEvent;
```

```
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;

public class PracovniObrazovka extends JPanel {
	VystupniOkno okno;
	JPanel obsah;
	JScrollPane jsp;

	private static final long serialVersionUID = 1L;

	public PracovniObrazovka(VystupniOkno o) {
		this.okno = o;
		setLayout(new BorderLayout());
		add(menu(), BorderLayout.NORTH);
		add(strom(), BorderLayout.WEST);

	}
	private JPanel strom() {

		return (new MenuVyberKapitoly(this));
	}
	private JMenuBar menu() {
		JMenuBar menuJMB = new JMenuBar();
		JMenu polozkaMenu = new JMenu("Menu");



		JMenuItem polozkaZpetDoMenu = new JMenuItem("Zpet do menu");
		polozkaZpetDoMenu.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				okno.aktualizuj(new UvodniObrazovka(okno));
			}
		});

		JMenuItem polozkaZavrit = new JMenuItem("Zavřít");
		polozkaZavrit.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				zavrit();
			}
		});

		JMenu polozkaHelp = new JMenu("Help");
		JMenuItem polozkaNapoveda = new JMenuItem("Nápověda");
		polozkaNapoveda.addActionListener(new ActionListener() {
```

```
	public void actionPerformed(ActionEvent e) {
		okno.akce(7, "");
	}
});
JMenuItem polozkaAbout = new JMenuItem("O Aplikaci");
polozkaAbout.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		vypisInfoOAplikaci();
	}
});

menuJMB.add(polozkaMenu);

polozkaMenu.add(polozkaZpetDoMenu);
polozkaMenu.addSeparator();
polozkaMenu.add(polozkaZavrit);

menuJMB.add(polozkaHelp);
polozkaHelp.add(polozkaAbout);
polozkaHelp.add(polozkaNapoveda);

JButton polozkaTeorie = new JButton("Teorie");
polozkaTeorie.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		okno.akce(1, "");
	}
});
if(okno.ridici.aktualniStatus == 0) {
	polozkaTeorie.setEnabled(false);
}
menuJMB.add(polozkaTeorie);
JButton polozkaResene = new JButton("Řešené příklady");
polozkaResene.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		okno.akce(2, "");
	}
});
menuJMB.add(polozkaResene);
if(okno.ridici.aktualniStatus == 1) {
	polozkaResene.setEnabled(false);
}
JButton polozkaNeresene = new JButton("Neřešené příklady");
polozkaNeresene.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		okno.akce(3, "");
	}
});
menuJMB.add(polozkaNeresene);
if(okno.ridici.aktualniStatus == 2) {
	polozkaNeresene.setEnabled(false);
}
```

```
		return menuJMB;
	}

	public void vybranaKapitola(String vybranaKapitola) {
		okno.akce(4, vybranaKapitola);
	}
	private void zavrit() {
		Object[] options = { "Ano", "Ne" };
		int n = JOptionPane.showOptionDialog(okno,
				"Opravdu chcete ukončit program", "Zavřít?",
				JOptionPane.YES_NO_OPTION,
JOptionPane.QUESTION_MESSAGE, null,
				options, options[0]);
		if (n == 0) {
			System.exit(0);
		}
		return;
	}

	private void vypisInfoOAplikaci() {
		String vypisInfoString;
		vypisInfoString = "Výukový program pro statistiku \n \n autor:
Harmady";

		JOptionPane.showMessageDialog(okno, vypisInfoString, "O
Aplikaci",
				JOptionPane.PLAIN_MESSAGE);
	}
	public void vlozPanel(JPanel panel) {
		obsah = panel;
		jsp = new JScrollPane(obsah);
		obsah.setPreferredSize(new Dimension(730, 515));
		add(jsp, BorderLayout.CENTER);
	}
	public void smazObsah() {
		remove(obsah);
		remove(jsp);
		validate();
	}
	public JPanel getObsah() {
		return obsah;
	} }
```

TeoriPanel.java:

```
package rozhrani;

import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.Image;
import java.awt.event.ActionEvent;
```

```
import java.awt.event.ActionListener;
import java.util.ArrayList;

import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JPanel;

import logika.Ridici;

public class TeoriePanel extends JPanel {

	JButton dalsi;
	JButton predchozi;
	String vybranaKapitola;
	ArrayList<Image> pole;
	MujPanelKresleni panelImg;
	Ridici r;
	JLabel pocitadlo;
	JPanel centralni;


	/**
	 *
	 */
	private static final long serialVersionUID = 1L;

	public TeoriePanel(ArrayList<Image> pole, String vybranaKapitola,
Ridici r) {
		this.r = r;
		this.vybranaKapitola = vybranaKapitola;
		this.pole = pole;
	    setLayout(new BorderLayout());
	    centralni = new JPanel();
	    centralni.setLayout(new BorderLayout());
		tlacitka();
		vlozObrazek();

	}


	private void tlacitka() {
		JPanel panelProTlacitka = new JPanel();
		panelProTlacitka.setLayout(new GridLayout(1,3));

		dalsi = new JButton("další");
		predchozi = new JButton("předchozí");
		predchozi.setIcon(new
ImageIcon(getClass().getResource("/img/predchozi.gif")));
		dalsi.setIcon(new
ImageIcon(getClass().getResource("/img/dalsi.gif")));
```

```
            dalsi.setPreferredSize(new Dimension(50,50));
            predchozi.setPreferredSize(new Dimension(50,50));

            JLabel nadpis = new JLabel(vybranaKapitola);
            JPanel horni = new JPanel();
            horni.add(nadpis);


            pocitadlo = new JLabel("");

            dalsi.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        if(r.aktualniSlajd < (pole.size() - 1)) {
                              r.aktualniSlajd = r.aktualniSlajd + 1;
                              String vypis = (r.aktualniSlajd+1) + "/" +
pole.size();
                              pocitadlo.setText(vypis);
                              vlozObrazek();
                        }

                  }
            });
            predchozi.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        if(r.aktualniSlajd > 0) {
                              r.aktualniSlajd = r.aktualniSlajd - 1;
                              String vypis = (r.aktualniSlajd+1) + "/" +
pole.size();
                              pocitadlo.setText(vypis);
                              vlozObrazek();
                        }
                  }
            });

            String vypis = (r.aktualniSlajd+1) + "/" + pole.size();
            pocitadlo.setText(vypis);

            JPanel jizni = new JPanel();
            jizni.add(pocitadlo);

            this.add(horni, BorderLayout.NORTH);
            this.add(jizni, BorderLayout.SOUTH);

            panelProTlacitka.add(predchozi);
            JPanel pom1 = new JPanel();
            panelProTlacitka.add(pom1);
            panelProTlacitka.add(dalsi);

            centralni.add(panelProTlacitka,BorderLayout.NORTH);

      }
```

```
	private void vlozObrazek() {
		try {
			panelImg.smazObrazek();
			MujPanelKresleni panelImg = new
			MujPanelKresleni(pole.get(r.aktualniSlajd), 900, 700,
			true);
			centralni.add(panelImg, BorderLayout.CENTER);
		}catch (Exception e) {
			MujPanelKresleni panelImg = new
			MujPanelKresleni(pole.get(r.aktualniSlajd), 900, 700,
			true);
			centralni.add(panelImg, BorderLayout.CENTER);
		}
		this.add(centralni, BorderLayout.CENTER);
		r.aktualizujVystup();
	}
}
```

PrikladyPanel.java:

```
package rozhrani;

import java.awt.Component;
import java.awt.Dimension;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Image;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.BorderFactory;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextArea;

import logika.Ridici;

public class PrikladyPanel extends JPanel {

	JPanel pracPlochaPN;

	GridBagLayout gbl;
	GridBagConstraints gbc;
	private static final int CEN = GridBagConstraints.CENTER;
	private static final int BOT = GridBagConstraints.BOTH;

	private JTextArea vypisZadani;
```

```
private MujPanelKresleni vypisReseni;
private JButton tlBReseni;
private JTextArea vysledek;
private JButton tlBpotvrzeno;
protected static JButton tlBdalsi;
protected JPanel pomocnyPN;
protected Ridici r;
private JLabel vypisSpravnostiJL;
int vyska;
private ImageIcon imgSpravnosti;
boolean graf;

String textPrikladu;

private static final long serialVersionUID = 1L;

public PrikladyPanel(String t,Ridici r, int status, boolean graf) {
	this.r = r;
	this.textPrikladu = t;
	this.graf = graf;
	vyska = r.getVystOkno().getHeight();
	vytvorKomponenty(status);
	vlozTextPrikladu();
}


private void vytvorKomponenty(int status) {
	vytvorGBL(status);


}
private void vytvorGBL(int status) {
	gbl = new GridBagLayout();
	this.setLayout(gbl);
	gbc = new GridBagConstraints();

	vypisZadani = new JTextArea("zadani");
	vypisZadani.setEditable(false);
	vypisZadani.setPreferredSize(new Dimension(10, vyska / 4));

vypisZadani.setBorder(BorderFactory.createTitledBorder("Zadání"));
	nastav(vypisZadani, 0, 0, 1, 1, 1.0, 1.0, BOT, CEN);

	vypisReseni = new MujPanelKresleni();
	vypisReseni.setPreferredSize(new Dimension(10, (vyska / 2) -
	20));
	nastav(vypisReseni, 0, 2, 1, 1, 1.0, 1.0, BOT, CEN);

	vysledek = new JTextArea();
	JPanel pom = new JPanel();
	vysledek.setPreferredSize(new Dimension(150, 20));
```

```
		pom.setBorder(BorderFactory.createTitledBorder(""));
		pom.add(vysledek);

		JLabel napisVysledek = new JLabel("Výsledek: ");

		tlBReseni = new JButton("reseni");
		tlBReseni.addActionListener(new ActionListener() {

			public void actionPerformed(ActionEvent e) {
				r.provedAkci(5, "");

			}
		});

		if(status == 2) {
			tlBReseni.setEnabled(false);
		}

		tlBpotvrzeno = new JButton("OK");
		tlBpotvrzeno.addActionListener(new ActionListener() {

			public void actionPerformed(ActionEvent e) {
				if (vypisSpravnostiJL != null) {
					vypisSpravnostiJL.setIcon(null);
				}
				if (vysledek.getText().equals("") && graf == false)
{

	JOptionPane.showMessageDialog(r.getVystOkno(),
							"Zadejte hodnotu vašeho
výsledku.",
							"Chyba vstupních dat",
JOptionPane.ERROR_MESSAGE);
				}
				else {
					try {
						String x = "";
						if(graf == false) {
							x = vysledek.getText();
							x = x.replace(',', '.');
							x = x.replace(',', '.');
							Double.parseDouble(x);
						}
						r.provedAkci(6, x);

					} catch (Exception e1) {
						String chybovaHlaska = "Chybny format
						vysledku \n (zkontrolujte, zda
						neobsahuje pismena "
								+ "nebo jine znaky.";
```

```
                                    JOptionPane.showMessageDialog(r.getVys
                                    tOkno(), chybovaHlaska,
                                                "Chyba vstupních dat",

      JOptionPane.ERROR_MESSAGE);
                                }
                          }
                    }
              });

              pomocnyPN = new JPanel();
              pomocnyPN.setPreferredSize(new Dimension(10, 20));
              nastav(pomocnyPN, 0, 1, 1, 1, 1.0, 1.0, BOT, CEN);

              pomocnyPN.add(napisVysledek);
              pomocnyPN.add(pom);
              pomocnyPN.add(tlBpotvrzeno);
              pomocnyPN.add(tlBReseni);


      protected void nastav(Component c, int x, int y, int s, int v, double
rs,
                    double rv, int vyp, int k) {

              gbc.gridx = x;
              gbc.gridy = y;
              gbc.gridwidth = s;
              gbc.gridheight = v;
              gbc.weightx = rs;
              gbc.weighty = rv;
              gbc.fill = vyp;
              gbc.anchor = k;

              gbl.setConstraints(c, gbc);
              add(c);
      }
      public void vlozTextPrikladu() {
              vypisZadani.removeAll();
              String pripis = "\n\npozor: vysledek zaokrouklete na 2 desetinna
cisla!";
              int k = r.prikladAktualni.getCisloKapitoly();
              if (k == 3 || k == 4 || k == 5 || k == 7 || k == 8) {
                    pripis = "";
              }
              vypisZadani.setText(textPrikladu + pripis);
      }
      public void vytvorTlacitkoNovy(final String vybranaKapitola, boolean
resene) {
              tlBdalsi = new JButton("nové zadání");

              tlBdalsi.addActionListener(new ActionListener() {
```

```
                public void actionPerformed(ActionEvent e) {
                        r.provedAkci(4, vybranaKapitola);
                }
        });
        pomocnyPN.add(tlBdalsi);
    }
    public void vlozReseni(Image img) {
        vypisReseni = new MujPanelKresleni(img, 270, 700, false);
        nastav(vypisReseni, 0, 2, 5, 5, 1.0, 1.0, BOT, CEN);
    }
    public void vlozGraf(int cisloZadani, Double[] p) {
        vypisReseni = new MujPanelKresleni();
        Graf g = new Graf(p, cisloZadani);
        vypisReseni.add(g);
        nastav(vypisReseni, 0, 2, 5, 5, 1.0, 1.0, BOT, CEN);
    }
    public void setVypisOSpravnosti(String text) {
        imgSpravnosti = new ImageIcon(getClass().getResource(
                "/img/" + text + ".gif"));
        vypisSpravnostiJL = new JLabel(imgSpravnosti);
        vypisSpravnostiJL.setToolTipText(text);
        pomocnyPN.add(vypisSpravnostiJL, 2);

        r.aktualizujVystup();
    }
}
```

Graf.java:

```
package rozhrani;

import javax.swing.JPanel;
import java.awt.*;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.CategoryPlot;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.DefaultCategoryDataset;
import org.jfree.data.general.DefaultPieDataset;

public class Graf extends JPanel {

    private static final long serialVersionUID = 1L;

    public Graf(Double[] args, int zadani) {

        switch (zadani) {
        case 1:
            sloupcovyGraf1(args);
            break;
```

```
		case 2:
			kolacovyGraf1(args);
			break;
		default:
			break;
		}
	}
	private void sloupcovyGraf1(Double[] args) {
		DefaultCategoryDataset data = new DefaultCategoryDataset();
		data.setValue(args[15] * 10, "zisk", "leden");
		data.setValue(args[17] * 10, "zisk", "unor");
		data.setValue(args[15] * 10, "zisk", "brezen");
		data.setValue(args[19] * 10, "zisk", "duben");
		data.setValue(args[18] * 10, "zisk", "kveten");
		data.setValue(args[19] * 10, "zisk", "cerven");
		data.setValue(args[14] * 10, "zisk", "cervenec");
		data.setValue(args[18] * 10, "zisk", "srpen");
		data.setValue(args[17] * 10, "zisk", "září");
		data.setValue(args[15] * 10, "zisk", "říjen");
		data.setValue(args[16] * 10, "zisk", "listopad");
		data.setValue(args[15] * 10, "zisk", "prosinec");

		JFreeChart chart = ChartFactory.createBarChart3D("Graf zisků",
"mesic",
				"zisk (v tis. kč)", data, PlotOrientation.VERTICAL,
true, false,
				true);
		chart.setBackgroundPaint(Color.WHITE);
		chart.getTitle().setPaint(Color.BLUE);
		CategoryPlot p = chart.getCategoryPlot();
		p.setBackgroundPaint(Color.WHITE);
		p.setRangeGridlinePaint(Color.RED);

		ChartPanel chartPanel = new ChartPanel(chart);
		 chartPanel.setPreferredSize(new Dimension(500, 220));
		 this.add(chartPanel);

	 }
	private void kolacovyGraf1(Double[] args) {
		DefaultPieDataset dataSet = new DefaultPieDataset();

		dataSet.setValue("výborně " + args[10] + "%", args[10]);
		dataSet.setValue("chvalitebně " + args[11] + "%", args[11]);
		dataSet.setValue("dobře 50%", 50);
		dataSet.setValue("dostatečně " + args[12] + "%", args[12]);

		double pocetPetek = 100 - (args[10] + args[11] + 50 + args[12]);
		dataSet.setValue("nedostatečně " + pocetPetek + "%",
pocetPetek);
```

```
            JFreeChart chart = ChartFactory.createPieChart("Výsledky
písemky",
                        dataSet, true, true, false);
            chart.setBackgroundPaint(Color.WHITE);
            chart.getTitle().setPaint(Color.BLUE);

            ChartPanel chartPanel = new ChartPanel(chart);
        chartPanel.setPreferredSize(new Dimension(500, 220));

        this.add(chartPanel);
      }
}
```

MenuVyberKapitoly.java:

```
package rozhrani;

import javax.swing.JPanel;
import javax.swing.JScrollPane;

import javax.swing.JTree;
import javax.swing.tree.DefaultMutableTreeNode;
import javax.swing.tree.TreeSelectionModel;
import javax.swing.event.TreeSelectionEvent;
import javax.swing.event.TreeSelectionListener;
import javax.swing.ToolTipManager;

import java.awt.GridLayout;

public class MenuVyberKapitoly extends JPanel implements
TreeSelectionListener {

      private static final long serialVersionUID = 1L;

      private JTree stromMenu;
      private DefaultMutableTreeNode uvodni;
      PracovniObrazovka o;
      int status;


      public MenuVyberKapitoly(PracovniObrazovka o) {
            super(new GridLayout(1, 0));
            this.o = o;
            this.status = o.okno.ridici.aktualniStatus;

            uvodni = new DefaultMutableTreeNode("Kapitoly");
            vytvorVetve();
            stromMenu = new JTree(uvodni);
```

```
            stromMenu.getSelectionModel().setSelectionMode(
                        TreeSelectionModel.SINGLE_TREE_SELECTION);
            ToolTipManager.sharedInstance().registerComponent(stromMenu);

            stromMenu.addTreeSelectionListener(this);
            JScrollPane stromJSP = new JScrollPane(stromMenu);

            add(stromJSP);
      }

      public void valueChanged(TreeSelectionEvent e) {
            DefaultMutableTreeNode vybrano = (DefaultMutableTreeNode)
stromMenu
                        .getLastSelectedPathComponent();

            if (vybrano == null)
                  return;

            Object infoOVybranem = vybrano.getUserObject();

            if (vybrano.isLeaf()) {
                  NastavNazev clanek = (NastavNazev) infoOVybranem;

                  o.vybranaKapitola(clanek.toString());
            }
            else if (!vybrano.isLeaf()) {
                  if(status == 0) {
                        stromMenu.expandPath(stromMenu.getSelectionPath());

                        int x = stromMenu.getSelectionRows()[0];

                        stromMenu.setSelectionRow(x+1);
                  }
                  else {
                        if(stromMenu.getSelectionRows()[0] == 0) {

                        }
                        else {


      stromMenu.expandPath(stromMenu.getSelectionPath());

      o.vybranaKapitola(stromMenu.getSelectionPath().getLastPathComponent().
toString());
                        }
                  }
            }
      }

      private class NastavNazev {
            public String nazevKapitoly;
```

```
		public NastavNazev(String kapitola) {
			nazevKapitoly = kapitola;

		}

		public String toString() {
			return nazevKapitoly;
		}
	}

	private void vytvorVetve() {
		DefaultMutableTreeNode kapitola = null;
		DefaultMutableTreeNode clanek = null;

		sekceKombinatorika(kapitola, clanek);

		sekcePravdepodobnost(kapitola, clanek);

		sekceGrafy(kapitola, clanek);

		sekceCharakteristikyZnaku(kapitola, clanek);
	}

	private void sekceKombinatorika(DefaultMutableTreeNode kapitola,
			DefaultMutableTreeNode clanek) {
		kapitola = new DefaultMutableTreeNode("1 Kombinatorika");
		uvodni.add(kapitola);

		if(status == 0) {
			clanek = new DefaultMutableTreeNode(new NastavNazev("1.1
Úvod"));
			kapitola.add(clanek);

			clanek = new DefaultMutableTreeNode(new NastavNazev("1.2
Variace"));
			kapitola.add(clanek);
			clanek = new DefaultMutableTreeNode(new NastavNazev("1.3
Permutace"));
			kapitola.add(clanek);

			clanek = new DefaultMutableTreeNode(new NastavNazev("1.4
Kombinace"));
			kapitola.add(clanek);
		}
		else {
			int pocetPrikladu =
			o.okno.ridici.priklad.getPocetPrikladuVKapitole(1,
			status);
			for (int i = 1; i <= pocetPrikladu; i++) {
```

```
                        clanek = new DefaultMutableTreeNode(new
                        NastavNazev("1." +i+" Příklad"));
                        kapitola.add(clanek);
                  }
            }
      }
      private void sekcePravdepodobnost(DefaultMutableTreeNode kapitola,
                  DefaultMutableTreeNode clanek) {

            kapitola = new DefaultMutableTreeNode("2 Pravděpodobnost");
            uvodni.add(kapitola);

            if(status == 0) {
            clanek = new DefaultMutableTreeNode(
                        new NastavNazev("2.1 Úvod"));
            kapitola.add(clanek);
            clanek = new DefaultMutableTreeNode(
                        new NastavNazev("2.2 Pravidla počítání"));
            kapitola.add(clanek);
            }
            else {
                  int pocetPrikladu =
                  o.okno.ridici.priklad.getPocetPrikladuVKapitole(2,
                  status);
                  for (int i = 1; i <= pocetPrikladu; i++) {
                        clanek = new DefaultMutableTreeNode(new
                        NastavNazev("2." +i+" Příklad"));
                        kapitola.add(clanek);
                  }
            }
      }

      private void sekceGrafy(DefaultMutableTreeNode kapitola,
                  DefaultMutableTreeNode clanek) {

            kapitola = new DefaultMutableTreeNode("3 Grafy");
            uvodni.add(kapitola);

            if(status == 0) {

            clanek = new DefaultMutableTreeNode(new NastavNazev("3.1
Úvod"));
            kapitola.add(clanek);
            clanek = new DefaultMutableTreeNode(new NastavNazev("3.2 Typy
grafů"));
            kapitola.add(clanek);
            }
            else {
                  int pocetPrikladu =
                  o.okno.ridici.priklad.getPocetPrikladuVKapitole(3,
                  status);
```

```
                  for (int i = 1; i <= pocetPrikladu; i++) {
                        clanek = new DefaultMutableTreeNode(new
                        NastavNazev("3." +i+" Příklad"));
                        kapitola.add(clanek);
                  }
            }
      }

      private void sekceCharakteristikyZnaku(DefaultMutableTreeNode
kapitola,
                  DefaultMutableTreeNode clanek) {

            kapitola = new DefaultMutableTreeNode("4 Charakteristiky
znaku");
            uvodni.add(kapitola);

            if(status == 0) {

            clanek = new DefaultMutableTreeNode(new NastavNazev("4.1
Úvod"));
            kapitola.add(clanek);
            clanek = new DefaultMutableTreeNode(new NastavNazev("4.2
            Charakteristiky polohy"));
            kapitola.add(clanek);
            clanek = new DefaultMutableTreeNode(new NastavNazev("4.3
            Charakteristiky variability"));
            kapitola.add(clanek);
            }
            else {
                  int pocetPrikladu =
                  o.okno.ridici.priklad.getPocetPrikladuVKapitole(4,
                  status);
                  for (int i = 1; i <= pocetPrikladu; i++) {
                        clanek = new DefaultMutableTreeNode(new
                        NastavNazev("4." +i+" Příklad"));
                        kapitola.add(clanek);
                  }
            }
      }
}
```

MujPanelKresleni.java:

```
package rozhrani;

import java.awt.Graphics;
import java.awt.Image;
import javax.swing.JPanel;

public class MujPanelKresleni extends JPanel {
```

```
	private static final long serialVersionUID = 1L;

	Image img;
	int v;
	int s;
	boolean isTeorie;

	public MujPanelKresleni() {
	}
	public MujPanelKresleni(Image obrazek, int vyska, int sirka, boolean
isTeorie) {
		this.img = obrazek;
		this.v = vyska;
		this.s = sirka;
		this.isTeorie = isTeorie;
	}
	public void paintComponent (Graphics g) {
		super.paintComponent(g);

		int realVyska = 397;
		int realSirka = 722;
		if (isTeorie) {
			g.drawImage(img, 10, 20, realSirka, realVyska, this);
		}
		else {
			g.drawImage(img, 10, 20, this.s, this.v, this);
		}
	}
	public void smazObrazek() {
		img = null;
		paintComponent(getGraphics());
	}
}
```

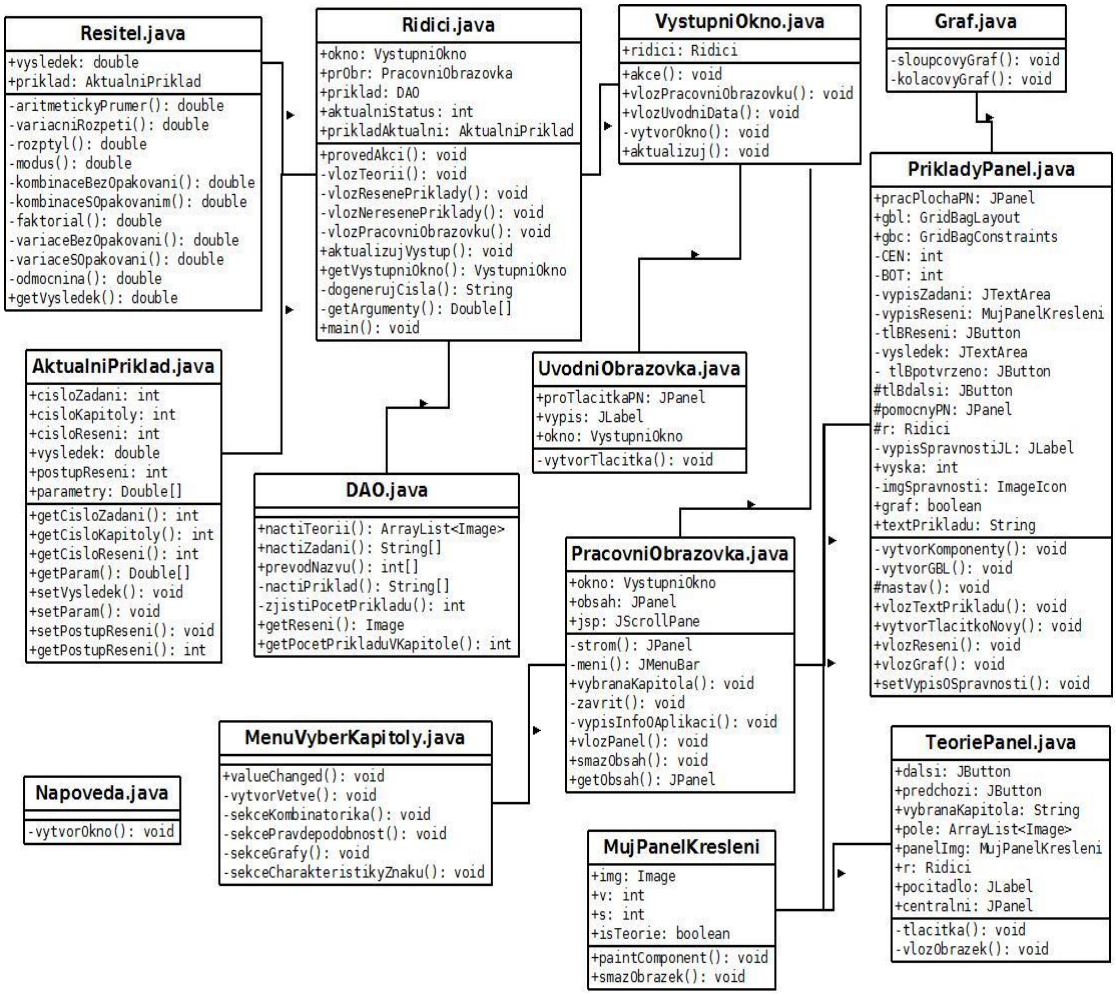
Napoveda.java:

```
package rozhrani;

import javax.swing.JFrame;
import javax.swing.JTextArea;

public class Napoveda extends JFrame {

	private static final long serialVersionUID = 1L;

	public Napoveda(int status) {
		vytvorOkno();

		JTextArea v = new JTextArea();
		v.setEditable(false);
		VypisyNapovedy vypisy = new VypisyNapovedy();
```

```
		switch (status) {
		case 0:
			setTitle("Nápověda- teorie");
			v.setText(vypisy.getNapovedaProTeorii());
			break;
		case 1:
			setTitle("Nápověda- řešené příklady");
			v.setText(vypisy.getProResene());
			break;
		case 2:
			setTitle("Nápověda- neřešené příklady");
			v.setText(vypisy.getProResene());
			break;

		default:
			break;
		}
		this.getContentPane().add(v);
		setVisible(true);
	}

	private void vytvorOkno() {
		setSize(500, 400);
		setTitle("Nápověda");
		setLocationRelativeTo(null);
	}
	class VypisyNapovedy {
		StringBuilder vypis;

		public VypisyNapovedy() {
			vypis = new StringBuilder();
		}

		public String getNapovedaProTeorii() {
			getVyberKapitoly();
			getVTeorii();
			navraty();

			return vypis.toString();
		}
		public String getProResene() {
			getVyberKapitoly();
			getProPriklady();
			navraty();

			return vypis.toString();
		}

		private void getVyberKapitoly() {
			vypis.append("Výběr Kapitoly \n\n");
```

```
                  vypis.append("V levém soupci vyberte kliknutím myši
kapitolu, \n");
                  vypis.append("kterou chcete procvičovat. Vámi vybraná
lekce se zobrazí \n");
                  vypis.append("na volném místě vpravo. Pokud chcete
kapitolu změnit,  \n");
                  vypis.append("vyberte kliknutím myši jinou kapitolu, ta se
                  zobrazí na místě \n");
                  vypis.append("předchozí kapitoly. \n");
            }
            private void getProPriklady() {
                  vypis.append("\n\n");
                  vypis.append("Váš výsledek zaokrouhlený na dvě desetinná
                  čísla zadejte do bílého \n");
                  vypis.append("pole za nápisem Výsledek a potvrďte
                  stisknutím tlačítka OK. \n");
                  vypis.append("V případě správné odpovědi se objeví zelená
                  značka, v opačném případě červený křížek. \n");

                  vypis.append("Stisknutím tlačítka řešení se v dolní části
                  programu objeví postup a výsledek příkladu \n");
                  vypis.append("(pokud se jedná o příklady s řešením) \n");

                  vypis.append("Tlačítko Nové zadaní (objeví se až po
                  zobrazení prvního příkladu) zobrazí jiný příklad \n");
                  vypis.append("aktuálně vybrané kapitoly \n");
            }
            private void getVTeorii() {
                  vypis.append("\n\n");
                  vypis.append("Kapitoly teorie mají více listů, jejich
počet \n");
                  vypis.append("a aktulaní list na kterém se nachíte lze
                  vidět mezi tlačítky \n");
                  vypis.append("předchozí a další. K listování v teorii
slouží \n");
                  vypis.append("právě tato tlačítka. \n");
            }
            private void navraty() {
                  vypis.append("\n\n");
                  vypis.append("Pro návrat na úvodní stránku vyberte v menu
                  položku Úvodní stránka \n");
                  vypis.append("Pro návrat do hlavního maenu vyberte v menu
                  položku Zpet do menu. \n");
                  vypis.append("Pro ukončení programu vyberte možnost
                  Zavřít, nebo stikněte křížek. \n");

            }
      }

}
```

Příloha B : UML diagram tříd



Příloha C : Abstrakt

HARMADY, J. *Tvorba výukového programu na podporu výuky statistiky na středních školách.* Bakalářská práce. Plzeň: Fakulta ekonomická ZČU v Plzni, 50 s., 2012

Klíčová slova: statistika, kombinatorika, pravděpodobnost, grafy, charakteristiky znaku, Java, výukový program, střední škola

Cílem této bakalářské práce je vytvoření výukového programu na podporu výuky statistiky na střední škole. Program je vytvořen v jazyce Java a distribuován v balíku JAR. Jedná se o desktopovou aplikaci umožňující získat teoretické poznatky ze statistiky, které se vyučuje na středních školách. Program obsahuje kapitoly kombinatorika, pravděpodobnost, grafy a charakteristiky znaku. V každé kapitole lze procvičovat příklady, do kterých se dynamicky generují hodnoty.

Program je sestaven tak, aby mohl nahradit nebo doplnit středoškolské učebnice statistiky, kterých se trhu příliš nenachází.

Příloha D : Abstract

HARMADY, J. *Creation tutorial to support the teaching of statistics in secondary schools.* Bachelor's thesis. Pilsen: Faculty of economics, University of West Bohemia, 50 p., 2012

Key words : statistics, combinatorics, probability, graphs, character features, Java, tutorial, high school, secondary school

The aim of this Bachelor´s thesis is creation tutorial to support the teaching statistics in secondary schools. Program is created in Java and distributed in JAR file. Tutorial is desctop aplication to acquire knowledge from secondary school statistics. Program includes these chapters combinatorics, probability, graphs and characteristics of the data. Every chapter includes examples to practice with dynamically generated values.

The application could replace or support statistics textbooks, which are not on the market too.