Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

Java aplikace – teorie a praxe

Vojtěch Dudáš

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Vojtěch Dudáš

Informatika

Název práce

Java aplikace – teorie a praxe

Název anglicky

Java applications – theory and practice

Cíle práce

Bakalářská práce je tematicky zaměřena na problematiku vývoje aplikací v jazyce Java. Hlavním cílem práce je charakterizovat základní aspekty, které jazyk Java programátorům nabízí.
Dílčí cíle bakalářské práce jsou:
-analyzovat obecné požadavky na samotný programovací jazyk,
- charakterizovat různé problémy, které s sebou vývoj aplikací přináší a ukázat možná řešení v praxi,
- navrhnout nejschůdnější možné řešení určitých problémů při návrhu a vývoji aplikací.

Metodika

Metodika řešené problematiky bakalářské práce je založena na studiu a analýze odborných informačních zdrojů. Dále také na vlastních zkušenostech autora BP získaných při tvorbě aplikací.

Vlastní řešení je realizováno formou návrhů různých metod, kterými lze řešit různé situace a problémy.

Na základě syntézy teoretických poznatků, praktických zkušeností a výsledků vlastního řešení budou formulovány závěry bakalářské práce.

Doporučený rozsah práce

30-40 stran

Klíčová slova

Java aplikace, příkazy, vzory, modely, situace, možnosti, hlediska, plánování, návrh

---

Doporučené zdroje informací

DARWIN, I., F. Java – Kuchařka programátora, Computer Press, 2006, 800 s. ISBN: 80-251-0944-5
HEROUT, P. Učebnice Jazyka Java. 2. vyd. České Budějovice : KOPP, 2006. 349 s. ISBN 80-7232-115-3.
KEEGH, J., GIANNINI, M. OOP Objektově orientované programování bez předchozích znalostí – Průvodce pro samouky, Computer Press, 2006, 224 s. ISBN: 80-251-0973-9
RISCHPATER, R. Beginning Java ME Platform, 2008, 600 s. ISBN:9781430210610

Předběžný termín obhajoby

2016/17 LS – PEF

Vedoucí práce

Ing. Čestmír Halbich, CSc.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 18. 10. 2016

Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 24. 10. 2016

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 14. 03. 2017

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Java aplikace - teorie a praxe" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne _____

Poděkování

Rád bych touto cestou poděkoval vedoucímu bakalářské práce panu Ing. Čestmíru Halbichovi, Csc., za odborné rady a konzultace během vedení této bakalářské práce.

Souhrn:

Tématem této bakalářské práce je Java aplikace – teorie a praxe. Bakalářská práce se v první části zabývá historií programovacího jazyka Java a jejich vývojem. Dále je Java rozdělena do verzí programovacího jazyka. V další části práce je charakteristika Javy s popisem jednotlivých vlastností programovacího jazyka. V poslední řadě je v práci ukázka vybraných prvků jazyka.

V praktické části je popsaná aplikace vytvořená v Javě, která je přílohou této bakalářské práce. Je zde seznámení s kódem samotné aplikace a popsání důvodu proč autor zvolil právě tuto implementaci jednotlivých částí aplikace. Vytvořená desktopová aplikace představuje návrh zpracování aplikace, která pracuje s lokální databází a je zaměřena na sledování přijmu kalorií a změnu tělesné hmotnosti pro fitness nadšence.

Summary:

The theme of this thesis is a Java application - theory and practice. The thesis deals in the first part of the history of the Java programming language and its development. Furthermore, it is divided into a Java version of the programming language. The next part is characteristic of Java, with descriptions of the various features of programming languages. Finally it is working demonstration of selected elements of language.

The practical part describes the application created in Java, which is attached to this thesis. There is a familiarity with the application itself, and describe the reason why the author chose this implementation of the individual parts of the application. Created a desktop application is a proposal processing application that works with local databases and is focused on tracking calorie intake and body weight change for fitness enthusiasts.

Klíčová slova:

Java aplikace, příkazy, problém, řešení, návrh, třída, metoda, rozhraní

Keywords:

Java applications, commands, problem, solution, design, class, method, interface

Obsah

1 Úvod

Výpočetní technika je pro dnešní svět velice důležitá. V mnohém usnadňuje život a její uplatnění bylo rozšířeno do všech odvětví moderního světa. Za posledních několik let, vývoj výpočetní techniky postoupil o tolik, že se stala skoro nepostradatelná. V mnoha ohledech nám zjednodušuje život a práci, ale pomalu se na ní stáváme závislými. Využití výpočetní techniky můžeme najít skoro ve všech přístrojích, v čele s počítači a mobilními telefony. Dále jsou to například Smart, TV, průmyslové stroje i domácí spotřebiče. V poslední době se zvětšuje počet odpůrců výpočetní techniky a to hlavně z toho důvodu, že z velké části ovlivňuje moderní svět a má na něj veliký dopad.

V úzkém spojení s výpočetní technikou se vyvíjejí i programovací jazyky, bez kterých by svět výpočetní techniky nemohl existovat. Už na počátku vývoje programovacích jazyků se vědělo, že budou nepostradatelnou součástí výpočetní techniky. Jeden z nejvíce rozšířených a oblíbených programovacích jazyků je jazyk Java. Z tohoto důvodu je právě jazyk Java tématem této bakalářské práce.

2 Cíl práce a metodika

2.1 Cíl práce

Bakalářská práce je tematicky zaměřena na problematiku vývoje aplikací v jazyce Java. Hlavním cílem práce je charakterizovat základní aspekty, které jazyk Java programátorům nabízí.

Dílčí cíle bakalářské práce jsou:

- analyzovat obecné požadavky na samotný programovací jazyk,
- charakterizovat různé problémy, které s sebou vývoj aplikací přináší a ukázat možná řešení v praxi,
- navrhnout nejschůdnější možné řešení určitých problémů při návrhu a vývoji aplikací.

2.2 Metodika

Metodika řešené problematiky bakalářské práce je založena na studiu a analýze odborných informačních zdrojů. Dále také na vlastních zkušenostech autora BP získaných při tvorbě aplikací.

Vlastní řešení je realizováno formou návrhů různých metod,
kterými lze řešit různé situace a problémy. Na základě syntézy teoretických poznatků, praktických zkušeností a výsledků vlastního řešení budou formulovány závěry bakalářské práce.

3 Charakteristika Javy

3.1 Historie Javy

3.1.1 Stručná historie programovacího jazyka C

C je univerzální programovací jazyk, který je úzce spojen s operačním systémem, pro který byl vyvinut. Většina systémů a programů, které jsou v UNIX provozovány jsou napsány v jazyce C. (2)

Programovací jazyk C je strukturovaný programovací jazyk, který byl vyvinutý v Bellových laboratořích v roce 1972 Dennisem Ritchiem.
Jeho funkce byly odvozeny z dřívějšího jazyka s názvem "B" (Basic Combined Programming Language - BCPL). (2) (3)

V roce 1978 Dennis Ritchie a Brian Kernighan publikovali první vydání "The C Programming Language", běžně známé jako K & R C. Programovací jazyk C byl standardizován institutem ANSI (American National Standards Institute) v roce 1988. Tato verze je známá jako ANSI C". (2) (3)

3.1.1.1 Strukturovaný programovací jazyk

U tohoto typu jazyků jsou velké programy rozděleny do malých programů, které se nazývají funkce. Primární důraz je kladen na data, s kterými funkce operují a ne na funkce samotné. Data u strukturovaně orientovaných jazyků se mohou volně pohybovat mezi funkcemi. Struktura takovýchto programů využívá přístup "shora dolů". (2) (3)

3.1.1.2 Vlastnosti programovacího jazyka C

Jazyk C je jedním z „mocných“ jazyků. Níže jsou uvedeny některé z jeho vlastností:

- spolehlivost
- přenosnost
- flexibilita
- interaktivita
- modularita

- účinnost a efektivnost

3.1.2 Stručná historie C++

V roce 1979 Bjarne Stroustrup, dánský počítačový vědec, začal pracovat na programovacím jazyku „C with Classes“ („C s třídami“). V roce 1983, byl „C with Classes“ přejmenován na „C ++“ ("++" vyjadřuje operátor pro inkrementaci C). C++ tedy je rozšíření původního jazyka C o prvky objektově orientovaného programování. Se změnou jména také přišlo přidání nových funkcí jako například virtuální funkce, reference, lepší kontrola typu a podle BCPL typu také jednořádkové komentáře se dvěma lomítky („//“). Důležitou součástí byl také vývoj samostatného kompilátoru pro C++, Cfront. (1) (2) (3)

C++ byl jakási „nástavba“ („superset“) k jazyku C. Očekávalo se, že C++ bude takzvaný „dokonalý jazyk“, pomocí kterého lze vytvořit všechny programy. Hlavní nevýhodou jazyka C++ je však jeho nepřenositelnost na jiné platformy, to tedy vede k tomu, že je nutný kompilátor pro každou platformu, což může být technicky i finančně náročné. (1) (2) (3)

3.1.3 Přínos jazyků C a C++ pro Javu

Programovací jazyk Java je odvozený od jazyka C a C++.
Návrháři Javy věděli, že pomocí známé syntaxe C a zopakováním objektově orientovaných rysů C ++, se Java stane přitažlivou pro spoustu zkušených C a C ++programátorů. Navíc k povrchové podobnosti, Java sdílí některé z dalších atributů, které pomohly C a C ++ být úspěšné. Java byla navržena, testována a upravená již zkušenými a pracujícími programátory. Je to jazyk vycházející z potřeb a zkušeností lidí, kteří ho vymysleli. To znamená, že je Java jazyk programátorský. Za druhé je Java soudržná a logicky konzistentní. Za třetí Java poskytuje plnou kontrolu programátorovi programu. To znamená, že Java přímo odráží zkušenosti a dovednosti programátora. Je to jazyk pro profesionální programátory. (1) (2) (3)

3.1.4 Historie Javy

Programovací jazyk Java byl vytvořen v roce 1991 týmem, který tvořili James Gosling, Patrick Naughton, Chris Warth, Ed Frank a Mike Sheridan. Tým pracoval ve společnosti s názvem Sun Microsystems, Inc. Trvalo jim 18 měsíců vyvinout první pracovní verzi. Tento programovací jazyk byl zpočátku nazvaný "Oak", ale byl přejmenován na "Java" v roce 1995. (2) (4)

Poněkud překvapivě nebyl originální impuls pro Javu rozvoj internetu. Místo toho, primární motivací bylo potřeba vytvořit jazyk nezávislý na platformě (to znamená, že je jeho architektura neutrální).

Jazyk Java měl být použit hlavně k vytvoření softwaru, který měl být použitý ve spotřební elektronice, jako je například mikrovlnná trouba nebo dálkové ovládání.
V elektronice je použito mnoho různých typů procesorů. Potíž s C a C ++ (a většinou jiných jazyků) je, že jsou navrženy tak, že jsou kompilovány pro konkrétní cíl. I když je možné sestavit program v jazyce C ++ pro téměř jakýkoliv typ procesoru. V tomto případě je ale nutné, aby existoval kompilátor pro daný typ procesoru. Problémem je, že kompilátory jsou drahé a časově náročné na vytvoření. Bylo zapotřebí vymyslet a vytvořit snadnější a nákladově efektivnější řešení. Ve snaze najít toto řešení, Gosling a jiní začali pracovat na přenosném, platformě nezávislém jazyku, který by mohl být použit k výrobě kódu, který by běžel na různých procesorech v rozdílných prostředích. Toto úsilí nakonec vedlo k vytvoření programovacího jazyka Java. (2) (4)

Přibližně v době, kdy se podrobnosti týkající Javy začaly zpracovávat se ukázal druhý a nakonec další velmi důležitý faktor. Tento faktor hrál klíčovou roli v budoucnosti Javy. Tímto faktorem byl velmi rychle se rozšiřující internet. Se vznikem World Wide Web byla Java poháněna do popředí počítačového designu programovacího jazyka, protože na webu byly také požadovány přenosné programy. (2) (4)
V roce 1993 bylo pro členy projekčního týmu Javy zřejmé, že se stejnými problémy s přenosností, jako například při vytváření kódu pro dálkové ovládání, se často setkávali při pokusu o vytvoření kódu pro internet. Ve skutečnosti, že stejný problém s přenositelností u řešení v malém měřítku, pro který Java byla původně navržena, může být také aplikován na internet. Toto uvědomění způsobilo změnu zaměření Javy od spo-

třební elektroniky na internetové programování. Právě toto rozhodnutí vedlo k velkoplošnému úspěchu jazyka Java. (2) (4)

3.2 Základní rysy Javy

Ačkoli jsou základními důvody, které podpořily vývoj Javy, přenositelnost a bezpečnost, také další faktory hrají důležitou roli při formování konečné podoby jazyka.
Klíčové faktory nebo „hesla“, které byly uvedeny týmem Javy, si uvedeme níže:

- jednoduchá
- přenosná
- objektově orientovaná
- robustní
- vícevláknová
- neutrální architektura
- vysoký výkon
- distribuovaná
- dynamická
- bezpečená

Toto jsou základní vlastnosti programovacího jazyka Java, u kterých si uvedeme podrobnější informace. (2) (4)

3.2.1 Jednoduchá

Java byla navržena pro zkušené programátory tak, aby bylo snadné se jí naučit a efektivně používat. Pokud už má uživatel nějaké zkušenosti s programováním, neměl by mít žádný problém naučit se tento jazyk. Největší výhodou pro profesionální programátory bylo, že pokud uměl kvalitně programovat v jazyku C ++, tak učení Javy už mu nezabralo skoro žádný čas. Vzhledem k tomu, že Java vychází z jazyků C a C ++, které byly a stále

jsou velice populární, tak většina i méně zkušených programátorů neměla potíže se programování v Javě naučit. (2) (4)

3.2.2 Přenosná

Přenositelnost v programovacím jazyce znamená Write Once and Run Anywhere (Wora) tedy „Jednou napiš a spusť kdekoli“.
Přenosnost dovoluje aplikaci spuštění na libovolné platformě nebo operačním systému beze změny kódu. Ve srovnání s jazykem C a C ++, Java kód není přímo dělaný na určitý stroj nebo na určitý objekt, ale když se Java program kompiluje, vygeneruje se .class soubor. Tento soubor se nazývá byte kód. Tam, kde chceme spustit tento byte kód, musíme mít Java překladač. Java překladač je k dispozici pro všechny dostupné operační systémy a to dělá Javu přenosnou. (2) (4)

3.2.3 Robustní

Více platformové prostředí internetu klade mimořádné nároky na program, protože program musí fungovat spolehlivě na různých systémech. To znamená, že na schopnost vytvářet robustní programy byla kladena velká priorita ve vývoji Javy. Vzhledem k tomu, že je Java přísně typový jazyk, zkontroluje váš kód při kompilaci. Java také kontroluje kód programu i při jeho běhu. Díky tomu si programátor nemusí hlídat spoustu nejčastějších chyb v kódu. Ve skutečnosti je skoro nemožné, udělat v programu takovou chybu, která by se špatně dohledávala nebo nešla dohledat vůbec.
Klíčem v Javě je, aby bylo snadné předpovědět budoucí chování programu. Například správa paměti může být v ostatních jazycích obtížná. V C nebo C++ musí programátor manuálně přidělit dynamickou paměť a to může vést k chybám. Například programátor bude uvolňovat paměť, kterou ještě program aktivně používá nebo může programátor zapomenout uvolnit paměť celkově. Java odstraňuje tyto problémy tím, že řídí alokace a dealokace za vás. (2) (4)

3.2.4 Víceúlohová

Java podporuje vícevláknové programování, to znamená, že program může dělat spoustu věcí najednou ve stejný čas. (2) (4)

3.2.5 Neutrální architektura

Jedním z hlavních problémů u programovacích jazyků je, že neexistuje záruka, že pokud napíšeme program dnes, poběží i zítra a to dokonce i na stejném stroji. Změna v základních systémových prostředcích, aktualizace procesoru a aktualizace operačního systému, jsou některé z důvodů, proč můžou vzniknout chyby v programu. Java návrháři dělali několik těžkých rozhodnutí v jazyce Java a Java Virtual Machine ve snaze tuto situaci změnit. Cílem bylo, udělat program řídící se heslem "write once; run anywhere, any time, forever." Do značné míry se to vývojářům Javy i povedlo. (2) (4)

3.2.6 Vysoký výkon

Jak bylo popsáno výše, Java umožňuje vytváření multi-platformní programy tím, že kompiluje programy do souboru s názvem Java bytecode. Tento kód může být vykonán na jakémkoli systému, který má Java Virtual Machine. Většina předchozích pokusů o řešení napříč platformami mělo za následek snížení výkonu výkonu. Java bytecode byl navržen tak, aby bylo snadné převést ho přímo do nativního strojového kódu pro velmi vysoký výkon pomocí kompilátoru just-in-time. Java run-time systémy zprostředkovávají tuto funkci bez jakékoli ztráty výhod přenositelnosti mezi platformami. (2) (4)

3.2.7 Distribuovaná

Java je vytvořena do prostředí internetu, protože dokáže zpracovávat TCP / IP protokoly. Java dokáže přistupovat k prostředku pomocí adresy URL a způsob jakým to dělá, se příliš neliší od přístupu k souboru. Java také podporuje „Remote Method Invocation" (RMI). Toto umožňuje programu volání metod v celé síti. (2) (4)

3.2.8 Dynamická

Tato vlastnost je klíčem k robustnosti appletu v prostředí, ve kterém mohou být malé fragmenty bytecódu dynamicky aktualizovány na běžícím systému. (2) (4)

3.2.9 Bezpečná

Pokaždé, když si stáhnete "normální" program riskujete, že spolu s ním, si stáhnete počítačový virus. Tento typ programu může shromažďovat osobní informace, jako jsou čísla kreditních karet, zůstatky bankovních účtů a hesla. Java brání této situaci tak, že poskytuje "firewall" mezi síťovou aplikací a vaším počítačem.

Při používání Java-kompatibilního webového prohlížeče, můžete bezpečně stáhnout Java applety bez obav z viru nebo jiného zlého úmyslu. Java dosahuje této ochrany tím, že omezí Java program pro spuštění pouze v Java prostředí a neumožní mu přístup do ostatních částí počítače. Možnost stahovat applety s důvěrou, že se uživatel nevystaví žádnému riziku, je mnohými považován za jeden z nejdůležitějších inovativních aspektů Javy. (2) (4)

3.3 Objektově orientovaný jazyk

Objektově orientovaného programování (OOP) je programovací paradigma, které používá "objekty" návrh aplikací a počítačových programů. Je založeno na několika technikách, včetně dědičnosti, modularity, polymorfismu a zapouzdření. To nebylo běžně používané při vývoji softwarových aplikací až do začátku roku 1990. Mnoho moderních programovacích jazyků nyní podporují OOP.

Programovací jazyk Simula byl první, kdo zavedl základní pojmy objektově orientovaného programování (objekty, třídy, podtřídy, virtuální metody, …). Smalltalk byl první programovací jazyk, který bude nazýván "objektově orientovaný".

Objektově orientované programování lze chápat jako soubor spolupracujících objektů, na rozdíl od tradičního pohledu, ve kterém může být program viděn jako seznam instrukcí zadávané počítači. V OOP je každý objekt schopen přijímat zprávy, zpracovávat data a posílat zprávy na jiné objekty. Na každý objekt může být nahlíženo jako na nezávislý malý stroj s výraznou rolí nebo odpovědností.

Objektově orientované programování má podpořit větší flexibilitu a udržovatelnost v programování, a je široce populární ve velkém měřítku softwarového inženýrství. OOP klade silný důraz na modularitu. Objektově orientovaný kód má být jednodušší na vývoj, má být více srozumitelnější a tím zjednodušit analýzu programu, kódování a pochopení složitých situací a postupů než menší modulární programovací metody.

Ačkoli Javu ovlivnili její předchůdci, nebyla navržena tak, aby měla zdrojový kód kompatibilní s jiným jazykem. Toto rozhodnutí dalo vývojovému týmu Java svobodu navrhnout programovací jazyk úplně od začátku. (1) (2) (4)

3.4 Verze Javy

3.4.1 JDK Alfa a Beta (1995)

Vydání alfy a bety Javy pro veřejnost, měly velmi nestabilní API a ABI. Java webový prohlížeč se jmenoval WebRunner. (3)

3.4.2 JDK 1.0 (23.Ledna, 1996)

Původní název pro tuto verzi byl „Oak“. První vydání stabilní verze JDK 1.0.2, se nazývá Java 1.

Ve verzích Javy a JDK až 1.0.1 mohly být ještě použity soukromá a chráněné klíčová slova společně pro vytvoření ještě další formy ochrany, který by omezil přístup k metodám nebo proměnným, výhradně na podtřídy dané třídy. Po vydání verze 1.0.2 byla tato funkčnost z jazyka odebrána. (3) (8) (9)

3.4.3 JDK 1.1 (19.Února, 1997)

K největším změnám patří přidání následujících funkčností:

- AWT - vybavení novými nástroji
- vnitřní třídy přidány k jazyku
- JavaBeans
- JDBC – propojení z databázovými systémy
- RMI - vzdálené volání metod
- reflexe - sebepozorování pouze, žádná úprava za běhu nebyla možná.

- JIT (Just In Time) kompilátor na platformě Microsoft Windows, vyráběný pro JavaSoft společností Symantec

(3) (8) (9) (10)

3.4.4 J2SE 1.2 (8.Prosince, 1998)

Kódové označení „**Playground“**. Tento a všechny následující verze až do J2SE 5.0 byly zpětně přejmenovány na Java 2 a název verze "J2SE" (Java 2 Platform Standard Edition) nahradil JDK pro rozlišení základní platformy od J2EE (Java 2 Platform, Enterprise Edition) a J2ME (Java 2 Platform , Micro Edition).

Tato verze byla velmi významná ve vývoji Java, protože ztrojnásobila velikost platformy Java na 1520 tříd v 59 balíčcích.

K největším změnám patří přidání následujících funkčností:

- strictfp – klíčové slovo
- Swing grafické API byly integrovány do základních tříd
- Sun JVM byl poprvé vybaven o JIT překladač
- Java plug-in
- Java IDL, implementace IDL pro CORBA
- collections framework

(3) (8) (9) (10)

3.4.5 J2SE 1.3 (8.Května, 2000)

Kódové označení „Kestrel“.

K největším změnám patří přidání následujících funkčností:

- HotSpot JVM (HotSpot JVM byl poprvé vydán v dubnu 1999 pro J2SE 1.2 JVM)
- RMI – upraven tak, aby podporovalo volitelnou kompatibilitu s CORBA
- Java Naming and Directory Interface (JNDI) – nově zahrnut v základních knihovnách (dříve k dispozici pouze jako rozšíření)
- Java Platform Debugger Architecture (JPDA)
- JavaSound
- syntetické třídy proxy serveru

(3) (8) (9)

3.4.6 J2SE 1.4 (6.Února, 2002)

Kódové označení „Merlin“. Jednalo se o první vydání platformy Java vytvořené v rámci „Java Community Process“ jako JSR 59.
K největším změnám patří přidání následujících funkčností:

změny jazyka

- assert – klíčové slovo (uvedeno v JSR 41.)

knihovny

- regulární výrazy
- výjimka řetězení umožňuje zapouzdření původní výjimky nižší úrovně
- podpora Internet Protocol version 6 (IPv6)
- I/O8 subsystém založený na kanálech (NIO9)
- záznam API (Uvedeno v JSR 47.)
- mage I / O rozhraní API pro čtení a psaní obrázků ve formátech jako je JPEG a PNG
- zpracování XML
- integrované zabezpečovací a kryptografické rozšíření (JCE, JSSE, JAAS)
- Java Web Start (Java Web Start byl poprvé vydán v březnu 2001 pro J2SE 1.3)
- nastavení API (java.util.prefs)

Podpora a bezpečnostní aktualizace pro Javu 1.4 skončila v říjnu 2008.
(3) (8) (9) (10)

3.4.7 J2SE 5.0 (30.Září, 2004)

Kódové označení „Tiger“. Původně měl označení 1.5, v tuto chvíli používano jako interní označení pro tuto verzi. J2SE 5.0 vstoupil do svého „end-of-public-updates“ období dne 8. dubna 2008. Aktualizace již nejsou přístupné veřejnosti od 3. listopadu 2009. Aktualizace byly pro Oracle zákazníky k dispozici až do května 2014. [13]
K největším změnám patří přidání následujících funkčností:

- generické typy
- metadata
- autoboxing / autounboxing

- enum – klíčové slovo, výčtové typy
- varargs: argumenty s proměnou délkou
- rozšíření cyklu „for each“- prochází celou kolekcí
- opravení dříve rozbité sémantiky Java paměťového modelu
- statické importy

Zlepšení v oblasti standardních knihoven:

- Swing: Nový přizpůsobivý vzhled a dojem nazvaný „synth“.
- formátovaný vstup a výstup

Java 5 je poslední verze Javy, která oficiálně podporovala Microsoft Windows 9x (Windows 95, Windows 98, Windows ME).
(3) (8) (9) (10)

3.4.8 JAVA SE 6 (11.Prosince, 2006)

Kódové označení „Mustang“. K této verzi, Sun nahradil název "J2SE" názvem Java SE a přestal používat označení, které na konci mělo ".0". Interní číslování pro vývojáře i nadále zůstávalo 1.6.0.

Během vývojové fáze, nové buildy aplikace včetně vylepšení a oprav chyb byly vydávány přibližně jednou týdně. Beta verze byla vydána v únoru a červnu 2006 a finální verze byla vydána dne 11. prosince 2006.
K největším změnám patří přidání následujících funkčností:

- podpora pro starší verze Win9x oficiálně zkončila
- dramatické zvýšení výkonu pro jádro platformy a Swing.
- vylepšená podpora webových služeb pomocí JAX-WS
- JDBC podpora 4.0
- Java kompilátory API: API umožňující programu výběr kompilátoru programově.
- modernizace JAXB na verzi 2.0
- mnoho vylepšení GUI, jako je například integrace SwingWorker v API, tabulkové třídění a filtrování, a Swing double-buffering (eliminuje gray-area efekt).
- vylepšení JVM: Synchronizace a výkonnostní optimalizace kompilátoru, nové algoritmy a modernizace stávajících algoritmů

(3) (8) (9) (10)

3.4.9 JAVA SE 7 (28.Července, 2011)

Kódové označení „Dolphin“. Tato verze je jedna z hlavních aktualizací, která byla zahájena dne 7. července 2011 a byla zpřístupněna pro vývojáře dne 28. července 2011.
K největším změnám patří přidání následujících funkčností:

- podpora JVM pro dynamické jazyky, s novým „invokedynamic“ bytekódem pod JSR-292
- malé změny jazyka (seskupeny v rámci projektu s názvem „Coin“):
    - řetězce v příkazu „switch“
    - automatické řízení zdrojů v try-statement
    - operátor diamant <>
    - zjednodušená metoda varargs
    - binární celočíselné literály
    - umožnit použití podtržítka v numerických literálech
    - *multi-catch* - Chytání více typů výjimek jediným příkazem *catch*
- nové balíčky java.nio.file a java.nio.file.attribute
- přidání frameworku *Fork/Join* – zajišťuje podporu pro paralelní programování
- upstream aktualizace XML a Unicode

Od dubna 2012, byla Java 7 jako výchozí verze ke stažení na java.com.
(3) (8) (9) (10)

3.4.10 JAVA SE 8 (18.Března, 2014)

Java 8 byla vydána 18. března 2014 a zahrnovala některé funkce, které byly plánovány ještě pro Java 7, ale byly odloženy.
K největším změnám patří přidání následujících funkčností:

- projekt „Nashorn“, JavaScript runtime, který umožňuje vývojářům vložit JavaScript kód do aplikací
- anotace na typy Java
- opakované poznámky
- datum a čas API

- statické propojení JNI knihoven
- spuštění aplikace JavaFX (přímé spouštění souborů JAR, aplikace JavaFX)
- odstranění trvalé generace

(3) (8) (9) (10)

3.4.10.1 Lambda výrazy

Lambda výrazy představují bezejmenné funkce, které lze použít i jako parametry metod, umožňující filtrovat vstupní hodnoty, na jejichž základě je posléze určena hodnota výstupu.

Java 8 není podporována na systému Windows XP. Od října 2014 byla Java 8 jako výchozí verze ke stažení na java.com. V době psaní této bakalářské práce je Java SE 8 aktuální oficiální verze Javy.
(3) (8) (9) (10)

3.4.11 JAVA SE 9

Java 9 má oficiální datum vydání plánované na červenec roku 2017.
K hlavním cílům této verze patří:

- ddělat Javu ještě více flexibilní
- zlepšit bezpečnost a udržitelnost
- přístup k lepšímu výkonu
- zjednodušení používaní Javy u rozsáhlých aplikací

(3) (8) (9) (10)

3.5 Základní struktury jazyka Java

3.5.1 Třída

Třída je šablona, která popisuje data a chování spojené s instancí této třídy. Třída je definována podle klíčového slova class a musí začínat velkým písmenem. Tělo třídy je obklopeno {}.

Data spojená se třídou jsou uložena v proměnné. Chování spojené s určitou třídou nebo objektem je realizováno metodami.
(2) (3) (5)

3.5.2 Balíček

Java skupiny tříd seskupeny do funkčních balíčků „Packages“.
Packages se obvykle používají k seskupení class do logických celků. Například, všechna grafická zobrazení aplikace mohou být umístěna ve stejném balíčku.
Je běžnou praxí používat reverzní doménové jméno společnosti jako balíček na nejvyšší úrovni. Například spolecnost.cz a název balíčku by začínal cz.spolecnost.
Dalším hlavním důvodem pro využití balíčků je zabránění kolize názvů jednotlivých tříd. Tato kolize nastane, když dva programátoři dávají stejné „*fully qualified name*“ třídě. Tento kvalifikovaný název třídy se v jazyce Java skládá z názvu balíčku následovaný tečkou (.) a jménem třídy.
(2) (5) (6)

3.5.3 Dědičnost

Třída může být odvozena z jiné třídy. V tomto případě se tato třída nazývá „podtřída“.
Dědičnost umožňuje třídě zdědit chování a data jiné třídy.
(2) (3)

3.5.4 Exception handling v Javě

V Javě je výjimkou taková událost, která oznámí chybu při běhu aplikace. To narušuje obvyklý tok instrukcí aplikace.
(3)

3.5.4.1 Checked Exceptions

„Checked Exceptions“ jsou explicitně vyvolány metodami, které by mohly způsobit výjimku nebo jí znovu vyvolat v případě, že vyvolána výjimka není zachycena.

Takže při volání metody, která může vyvolat „Checked Exception“, musí být výjimky buď zachyceny, nebo znovu vyvolány.
(3) (6)

„Checked Exceptions“ jsou používány v takové části programu, kdy může být chyba předvídatelná za určitých okolností, například soubor, který nemůže být nalezen.

3.5.4.2 Runtime Exceptions

„Runtime Exceptions“ jsou výjimky, které nejsou výslovně uvedené v metodě, a proto také nemusejí být explicitně chycené.
Nejznámější „Runtime Exception“ je NullPointerException, ke kterému dochází při běhu, kdy je metoda vyvolána na objekt, který je ve skutečnosti null.
(2) (3) (5)

3.5.5 Proměnné

3.5.5.1 Proměnná

Proměnné umožňují Java programu ukládání hodnot při běhu.
Jsou dva typy proměnných: primitivní a referenční.
Primitivní proměnná obsahuje hodnotu. Referenční proměnná obsahuje odkaz (ukazatel) k objektu.
(3) (5) (6)

3.5.5.2 Proměnná instance

Proměnná instance je spojena s instancí třídy (nazývané také objekt). Přístup k proměnné funguje přes tyto objekty.
Proměnné instance mohou mít jakoukoli kontrolu přístupu a mohou být označeny jako konečné nebo přechodné.
(3) (5) (6)

3.5.5.3 Lokální proměnná

Místní (stack) deklarace proměnných, nemůže mít modifikátory přístupu. Lokální proměnné nedostanou výchozí hodnoty, a proto musí být inicializován před tím, než mohou být použity.
„final“ je jediný modifikátor k dispozici pro lokální proměnné. Tento modifikátor definuje, že proměnná nemůže být změněna po prvním úkolu.
(2) (3) (5)

3.5.6 Metody

Metoda je blok kódu s parametry a návratovou hodnotou. Může být také nazývána objekt.
Metody mohou být deklarovány s argumenty. V tomto případě metoda deklaruje parametr, který vstupuje jako proměnná do metody. To umožní mnohonásobné použití metody, pro nekonečně mnoho vstupních hodnot.
(3)

3.5.7 Import statements

3.5.7.1 Použití import statements

Máte přístup k Java třídě vždy prostřednictvím plně kvalifikovaného jména „*fully qualified name“* , to znamená název balíčku plus (.), po níž následuje jméno třídy.
Můžete přidat příkaz „import“ do souboru třídy. Ty umožňují použít příslušné třídy v kódu bez její package kvalifikace.
(2) (3) (6)

3.5.7.2 Statické importy

Statický import umožňuje `public static` členy (pole a metody) jiné třídy, použít v Java kódu bez ohledu na referencování třídy.
Tato funkce zlepšuje čitelnost kódu a umožňuje Java API návrhářům napsat stručnou API.
(4) (5) (6)

3.5.8 Java podmínky

3.5.8.1 if-then a if-then-else

„if-then“ je tvrzení řízení toku dat. Blok kódu se provádí pouze tehdy, pokud část specifikována v podmínce „if“ je splněna. Volitelný „else“ blok je vykonán, když podmínka v „if“ není splněna.
(2) (3) (5)

3.5.8.2 Switch

Příkaz switch může být použit ke zpracování několika podmínek, pokud jsou založeny na stejné konstantní hodnotě.
(2) (3) (5)

3.5.9 Smyčky v Javě

3.5.9.1 The for loop

„For“ smyčka je ovládací opakování do které je zapsán blok kódu, který je spuštěn pro určitý počet opakování. Syntaxe je následující.
(4) (5) (6)

3.5.9.2 The while loop

Smyčka while je ovládací opakování do které je zapsán blok kódu, který je opakovaně proveden až do té doby, než je podmínka vyhodnocena jako false. Syntaxe je následující.
(2) (3)

3.5.10 Pole

3.5.10.1 Arrays in Java

Pole je objekt typu kontejner, který drží pevně stanovený počet hodnot jednoho typu. Položka v poli se nazývá prvek. Ke každému prvku lze přistupovat pomocí indexu. První prvek v poli má index 0. (3) (5)

3.5.11 Řetězce

3.5.11.1 Strings in Java

Třída „String“ představuje řetězce znaků. Všechny řetězcové literály, například "ahoj", jsou implementovány jako instance této třídy. Řetězce jsou neměnitelné, například přiřazení nové hodnoty na objekt „String“ vytvoří nový objekt.
Pokud používáte proměnné různých typů, Java vyžaduje pro určité typy explicitní konverze.
(3) (5)

3.5.12 Statement

3.5.12.1 Statement

Statement rozhraní slouží k provádění běžných SQL dotazů. Při používání statement nemůžete předat parametry dotazu SQL v době spuštění programu. Toto rozhraní je výhodnější než PreparedStatement v případě, že budete chtít provést příslušné SQL dotazy pouze jednou. Výkon tohoto rozhraní je velmi malý ve srovnání s dalšími rozhraními, které umožní provádět SQL dotazy.
(2) (3) (5) (6)

3.5.12.2 PreparedStatement

PreparedStatement se používá k provádění dynamických nebo parametrizovaných SQL dotazů. PreparedStatement rozšiřuje rozhraní statement. U tohoto rozhraní můžeme předat parametry SQL dotazu v době spuštění programu. PreparedStatement se doporučuje používat, pokud chcete provádět určitý SQL dotaz vícekrát. PreparedStatement má také lepší výkon než Statement rozhraní vzhledem k tomu, že PreparedStatement je předkompilováván a dotaz je vytvořen pouze jednou bez ohledu na to, kolikrát jste použili tento dotaz.
(2) (3) (5) (6)

3.5.13 Swing

Swing je primární GUI widget toolkid Javy. Je součástí Oracle Java Foundation Classes (JFC) - API pro poskytování grafického uživatelského rozhraní (GUI) pro programy v jazyce Java.

Swing obsahuje sofistikovanější sadu GUI komponent než předchozí Abstract Window Toolkit (AWT). Swing poskytuje nativní vzhled a pocit, že napodobuje vzhled a dojem z několika platforem. Na rozdíl od AWT komponent, Swing komponenty nejsou implementovány kódem specifickým pro platformu. Místo toho jsou zcela napsané v Javě a to jim dodává nezávislost na platformě.
(3) (5) (7)

4 PRAKTICKA ČÁST

4.1 Cíle praktické části

Součástí bakalářské práce je i praktická část, která popisuje desktopovou aplikaci s názvem

„Fitness kalkulačka". Primárním cílem aplikace je, demonstrace způsobu možného řešení, jak lze zpracovat aplikaci, která přímo pracuje z databází. Aplikace má ukázat, jak lze pracovat s údaji uživatelů a zpravovaní jejich přístupu k nim. Mezi dílčí cíle aplikace patří vývoj aplikace, která je zaměřena na sledování přijmu kalorií a změnu tělesné hmotnosti pro zájemce o fitness. V neposlední řadě aplikace názorně ukazuje praktické použití vybraných prvků Javy, které byly popsány v teoretické části této práce. Aplikace byla zpracována ve vývojovém prostředí Eclipse Neon.1.

Aplikaci je možné spustit přímo přes vývojové prostředí nebo pomocí vygenerovaného JAR souboru.

4.2 Popis kódu aplikace

4.2.1 Třída sqliteConnection

Účelem této třídy je pouze propojení desktopové aplikace s databází. Pro spojení s databází je použito JDBC. Třída obsahuje metodu „dbConnector()“, pomocí které se vytvoří propojení s SQLite databázi. Důležité pro tuto třídu jsou SQL knihovny „java.sql.*“. Toto označení je použito pro import všech knihoven s SQL.

Ukázka kódu:

```
import java.sql.*;
import javax.swing.*;
public class sqliteConnection {
Connection conn = null;
public static Connection dbConnector() {
     try {
          Class.forName("org.sqlite.JDBC");
          Connection conn=DriverManager.getConnection
               ("jdbc:sqlite::resource:FitCalcDB.sqlite"
          );
          return conn;
     } catch (Exception e) {
               JOptionPane.showMessageDialog(null,
               "Přihlášení k DB se nepovedlo");
          return null;
     }
}
}
```

Pomocí metody *getConnection*("jdbc:sqlite::resource:FitCalcDB.sqlite") se do proměnné „conn“ nastaví připojení k databázi. V tomto případě je SQLite databáze s názvem „FitCalcDB.sqlite“ umístěna do souboru „resource“. V případě nedohledání databáze se díky „try-catch“ bloku zobrazí hláška o neúspěšném připojení databáze.

4.2.2 Třída Login

Grafická třída, která pomocí JFrame zobrazuje přihlašovací formulář. Grafické uživatelské rozhraní je v programu řešeno pomocí knihovny swing. Jedná se o hlavní třídu aplikace, která obsahuje metodu main.

```
public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                Login window = new Login();
                window.loginFrame.setVisible(true);
            } catch (Exception e) {
                JOptionPane.showMessageDialog(null,
                e);
            }
        }
    });
}
```

Ve třídě login jsou vytvořeny pole pro jméno a heslo do kterých uživatel vyplní své přihlašovací údaje. Pro pole heslo je použit prvek JPasswordField(), který umožní skrytí hesla za pomocí jiných znaků, například „*“.

Ukázka pole pro heslo:

```
passwordHeslo = new JPasswordField();
passwordHeslo.setBounds(239, 110, 86, 20);
loginFrame.getContentPane().add(passwordHeslo);
```

Dále se na obrazovce nachází tlačítka „Přihlásit“ a „Registrace“. Tlačítko „Přihlásit“ zkontroluje platnost hesla a vyvolá úvodní obrazovku aplikace.

Ukázka tlačítko „Přihlásit“:

```
JButton btnLogin = new JButton("Přihlásit");
btnLogin.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        try {

            String querry = "select * from Uzivatele
            where jmeno = ? and heslo = ?";
            PreparedStatement pst =
            connection.prepareStatement(querry);
            pst.setString(1,
            textLoginJmeno.getText());
            pst.setString(2,
            passwordHeslo.getText());
```

```
ResultSet rs = pst.executeQuery();
int count = 0;
while (rs.next()) {
     count = count + 1;

}
if (count == 1) {
     JOptionPane.showMessageDialog(null,
     "Přihlášení proběhlo úspěšně");

     loginFrame.dispose();
     JmenoUzivatele();
     FitnessKalkulačka.Fit();

} else if (count > 1) {
     JOptionPane.showMessageDialog(null,
     "Duplicitní přihlašovací jméno a
     heslo");
} else {
     JOptionPane.showMessageDialog(null,
     "Kombinace přihlašovacího jména a
     hesla není správná");
}
  rs.close();
  pst.close();
```

```
                } catch (Exception e) {
                    JOptionPane.showMessageDialog(null,
                    e);
                }

            }

        });
        btnLogin.setBounds(140, 215, 109, 29);
        loginFrame.getContentPane().add(btnLogin);
```

Heslo uživatele se ověřuje oproti heslu, které je uložené v databázi u jeho jména. Připojení k databázi proběhlo přes výše popsanou třídu sqliteConnection.

```
connection = sqliteConnection.dbConnector();
```

Za použití PreparedStatement , se nad databází spustí select, který je uložen v proměné „querry“. Díky PreparedStatement můžeme použít parametrizovaný SQL příkaz, do kterého můžeme vložit hodnoty z polí, kam uživatel zadával své přihlašovací údaje. Toto zprostředkovává metoda „setString“ a „getText“. Metoda setString nastaví hodnoty místo znaku „?“ v SQL selectu. Pořadí „?“ určuje indexy v metodě setString.

Podmínka „while“ počítá výskyt přihlašovacích údajů v databázi. Pokud se v databázi údaje vyskytují právě jednou, tak je uživatel úspěšně přihlášen do aplikace. Pokud se údaje v databázi nevyskytují, zobrazí se pouze informativní hláška o tom, že údaje neodpovídají uloženým údajům. Je zde ošetřen i málo pravděpodobný scénář, kdy se

v databázi vyskytuje více stejných záznamů, ale tento stav by neměl nastat, protože při registraci se hlídá unikátnost přihlašovacího jména.

Po úspěšném přihlášení se pomocí funkce dispose() schová JFrame, který slouží jako přihlašovací obrazovka a vyvolá se třída FitnessKalkulačka.

Na konci „try“ bloku se musí uzavřít připojení k databázi pomocí object.close(), aby se uvolnilo připojení k databázi a mohlo být znovu použito dalšími metodami.

Tlačítko „Registrace“ otevře okno pro registrování nového uživatele do systému. Třída, která toto zajišťuje, bude popsána níže.

4.2.3 Třída Registration

Třída Registration je další grafická třída, sloužící pro vytvoření nových účtů pro uživatele. Pomocí této třídy si uživatel může zvolit své přihlašovací jméno a heslo a tím možnost přístupu do aplikace. Obrazovka pro registraci obsahuje pole pro jméno a heslo, kde pro heslo je použito znovu JPasswordField(), jako tomu bylo u obrazovky přihlášení. Dále jsou přístupná tlačítka „Registrovat“ a „Zpět“. Pomocí tlačítka „Registrovat“ proběhne zápis zadaných údajů uživateli do databáze. Tlačítko „Zpět“ slouží pro vypnutí obrazovky registrace.

Pole pro přihlašovací jméno uživatele:

```
textRegJmeno = new JTextField();
textRegJmeno.addKeyListener(new KeyAdapter() {
    public void keyTyped(KeyEvent e) {
        if (textRegJmeno.getText().length() >= 15) {
            Toolkit.getDefaultToolkit().beep();
            e.consume();
        }
```

```
        }
    });
```

U pole pro přihlašovací jméno uživatele je použita metoda keyTyped() s podmínkou pro omezení počtu znaků pro jméno. Po patnácti znacích, což by mělo být plně dostačující pro kreativní přihlašovací jméno, se díky metodě beep(), ozve zvukové znamení, které značí, že už není možné vkládat další znaky a znaky se začnou pohlcovat metodou consume().

Ukázka funkce tlačítka „Registrovat“:

```
JButton btnReg = new JButton("Registrovat");
btnReg.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        try {

            String querry1 = "insert into Uzivatele
            (jmeno,heslo) values (?,?)";
            PreparedStatement pst =
            connection.prepareStatement(querry1);
            pst.setString(1, textRegJmeno.getText());
            pst.setString(2,
            passwordRegHeslo.getText());

            pst.execute();
```

```
                    JOptionPane.showMessageDialog(null,
                    "Registrace proběhla úspěšně");

                    pst.close();
                    registrationFrame.dispose();

                } catch (org.sqlite.SQLiteException e1) {
                    JOptionPane.showMessageDialog(null,
                    "Uživatelské jméno již existuje, zvolte
                    prosím jiné");
                } catch (Exception e) {
                    JOptionPane.showMessageDialog(null, e);
                }

            }
        });
        btnReg.setBounds(178, 220, 106, 28);
        registrationFrame.getContentPane().add(btnReg);
```

Po zmáčknutí tlačítka proběhne kontrola unikátnosti přihlašovacího jména. Kontrola probíhá pomocí try-catch bloku, kde se v bloku catch objevuje org.sqlite.SQLiteException. Kontrola je důležitá z toho důvodu, že atribut „přihlašovací jméno" má v databázi příznak „UNIQUE". Pro tuto aplikaci byl zvolen, kvůli jednodušší implementaci, jako unikátní klíč právě přihlašovací jméno uživatele. Tento unikátní klíč by bylo vhodné nahradit s ID uživatele v případě, že by aplikace používala globální databázi s více uživateli.

Vložení hodnot do tabulky je řešeno pomocí PreparedStatement a to ze stejného důvodu jako již bylo popsáno u třídy Login, tedy kvůli parametrizovanému SQL příkazu. Po úspěšném ověření přihlašovacího jména se uživateli zobrazí hláška o úspěšné registraci a registrační okno se zavře.

4.2.4 Třída Helpers

Třída Helpers je pouze pomocná třída, ve které jsou uloženy metody používané na více místech v aplikaci.

Ukázka metod:

```
public static void RefreshTable() {
    try {

        String querry = "select * from Macro";
        PreparedStatementpst= connecti-
        on.prepareStatement(querry);
        ResultSet rs = pst.executeQuery();
        FitnessKalkulačka.mainTable.setModel
        (DbUtils.resultSetToTableModel(rs));

        rs.close();
        pst.close();

    } catch (Exception e) {
```

```
            e.printStackTrace();
        }
    }
```

Metoda RefreshTable() je používaná pro zobrazení hodnot z databázové tabulky „Macro“ do tabulky „mainTable“ v aplikaci.

```
    public static void RefreshUserTable() {
        try {

            String querry = "select
            ID,Název,Bílkoviny,Sacharidy,Tuky,Kcal from
            UzivatelDen where
            UserName ='"+ FitnessKalkulačka.Jmeno + "'";
            PreparedStatement pst =
connection.prepareStatement(querry);
            ResultSet rs = pst.executeQuery();
            FitnessKalkulačka.UserTable.setModel
            (DbUtils.resultSetToTableModel(rs));

            rs.close();
            pst.close();

        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
}
```

Metoda RefreshUserTable() je stejně jako metoda RefreshTable() používaná pro zobrazení hodnot z databáze, tentokrát se jedná pouze o tabulku, která je pro každého uživatele zobrazena jinak. Pomocí části SQL dotazu where UserName ='"+ FitnessKalkulačka.*Jmeno* + "'jsou vybrány hodnoty z tabulky „UzivatelDen“, které patří právě přihlášenému uživateli. Jak už bylo popsáno u třídy „Registration“, přihlašovací jméno je zde použito jako jednoznačný identifikátor uživatele.

```
public static void RefreshDenikTable() {
    try {

        String querry = "select Váha,Datum from Denik
        where
        UserName ='" + FitnessKalkulačka.Jmeno + "'";
        PreparedStatement pst = connection.prepareStatement(querry);
        ResultSet rs = pst.executeQuery();
        FitnessKalkulačka.tableDenik.setModel
        (DbUtils.resultSetToTableModel(rs));

        rs.close();
        pst.close();

    } catch (Exception e) {
        e.printStackTrace();
    }
```

```
    }
```

Metoda RefreshDenikTable() je stejně jako metoda RefreshUserTable() používaná pro zobrazení hodnot z databáze, pouze se tentokrát jedná o tabulku „Denik“. Výběr hodnot probíhá také podle jména přihlášeného uživatele.

Tyto metody pro zobrazení hodnot z databáze by mohly být upraveny do jedné metody s přidanými parametry, ale zde byl zvolen způsob více podobných metod pro lepší přehlednost SQL příkazů.

```
    public static Double VypocetDen(int pocet) {
        int rowsCount = FitnessKalkulačka.UserTable.getRowCount();
        double mezisoucet = 0;

        for (int i = 0; i < rowsCount; i++) {
            mezisoucet = mezisoucet + Double.parseDouble(FitnessKalkulačka.UserTable
            .getValueAt
            (i, pocet).toString());
        }

        Double hodnota = Math.round(mezisoucet * 100) / 100.0;

        return hodnota;
    }
```

```
    public static void RefreshMacroZaDen() {


    FitnessKalkulačka.textBden.setText(Double.toString(Vypoc
etDen(2)));

    FitnessKalkulačka.textSden.setText(Double.toString(Vypoc
etDen(3)));

    FitnessKalkulačka.textTden.setText(Double.toString(Vypoc
etDen(4)));

    FitnessKalkulačka.textKden.setText(Double.toString(Vypoc
etDen(5)));
    }
```

Tato dvojice metod slouží pro výpočet součtu nutričních hodnot pro uživatele za daný den.

Metoda VypocetDen(**int** pocet) má jeden parametr, který do ní vstupuje a to „pocet“, který značí index atributu v tabulce „UserTable“ . Díky tomu se vybere právě počítaná nutriční hodnota. Cyklus for slouží pouze k tomu, aby se sečetly všechny řádky daného atributu. Pro výpočet je nutné převést String hodnotu, načtenou z tabulky, na číselnou hodnotu, v tomto případě Double. Dále se hodnota ukládá do proměnné, kterou pak metoda vrací.

Metoda RefreshMacroZaDen()poté pouze převede výsledek metody VypocetDen(**int** pocet) zpět na String, který se zobrazí v poli pro právě počítanou nutriční hodnotu.

```
public static void VytvorGrafDenik() {
    try {

        String querry = "select Datum,Váha from Denik
        where UserName ='" + FitnessKalkulačka.Jmeno+
        "' order by Datum";

        JDBCCategoryDataset dataset = new
        JDBCCategoryDataset(sqliteConnection.dbConnect
        or(), querry);
        JFreeChart GrafDenik =
        ChartFactory.createLineChart("Deník", "Datum",
        "Váha", dataset,
        PlotOrientation.VERTICAL, false, true, true);
        BarRenderer renderer = null;

        CategoryPlot plot =
GrafDenik.getCategoryPlot();
        CategoryAxis xAxis = (CategoryAxis)
plot.getDomainAxis();

        xAxis.setCategoryLabelPositions
        (CategoryLabelPositions.DOWN_90);

        renderer = new BarRenderer();

        ChartPanel graf = new ChartPanel(GrafDenik);
```

```
            FitnessKalkulačka.panelDenikGraf.removeAll();
            FitnessKalkulačka.panelDenikGraf.add(graf);
            FitnessKalkulačka.panelDenikGraf.validate();

        } catch (Exception e1) {
            e1.printStackTrace();
        }

    }
```

Metoda VytvorGrafDenik() vytváří graf pokroku váhy pro přihlášeného uživatele. Pro vykreslení spojnicového grafu je použita knihovna JFreeChart. Do grafu se vykreslí hodnoty z tabulky „Denik“.

```
    public static boolean validateJavaDate(String strDate) {
        if (strDate.trim().equals("")) {
            return false;
        } else {

            SimpleDateFormat sdfrmt = new
            SimpleDateFormat("yyyy-MM-dd");
            sdfrmt.setLenient(false);
            Date javaDate = null;
            try {
                javaDate = sdfrmt.parse(strDate);
```

```
            } catch (Exception e) {
                return false;
            }
            return true;
        }
    }
```

Poslední metodou je validateJavaDate(String strDate) . Tato metoda slouží k validaci data vyplněného uživatelem. První částí je podmínka „if-else“, která hlídá vyplnění pole. Pokud pole vyplněno nebylo vůbec, metoda vrací boolean hodnotu false. Pokud však uživatel datum vyplnil, kontroluje se správnost formátu. Tato kontrola je nutná z důvodu sjednocení formátu dat. Sjednocený formát zjednodušuje práci s daty. Druhá část podmínky pracuje s try-catch blokem, který porovnává formát data s předem nadefinovaným formátem. Pokud tedy datum nemá stejný formát jako je předem nadefinovaný, spustí se část kódu v catch bloku a funkce vrátí hodnotu false. Pokud je datum vyplněné a má správný formát, tak funkce vrací boolean hodnotu true.

4.2.5 Třída FitnessKalkulačka

Třída FitnessKalkulačka je nejrozsáhlejší grafickou třídou celé aplikace. Zde je definován vzhled a prvky hlavní části programu. Kvůli velké rozsáhlosti třídy si zde uvedeme pouze některé ze zajímavějších prvků, které se příliš nepodobají prvkům z ostatních probíraných tříd. Kód celého programu je přiložen jako příloha k bakalářské práci.

```
mainTable = new JTable();
mainTable.addMouseListener(new MouseAdapter() {
    public void mouseClicked(MouseEvent arg0) {
```

```
                try {

                    int row = mainTable.getSelectedRow();
                    String Název_ =
(mainTable.getModel().getValueAt
                    (row, 0).toString());

                    String querry = "select * from Macro
                    where Název = '" + Název_ + "'";
                    PreparedStatement pst =
                    connection.prepareStatement(querry);
                    ResultSet rs = pst.executeQuery();

                    while (rs.next()) {

      textNazev.setText(rs.getString("Název"));
                        textBilkoviny.setText
                        (rs.getString("Bílkoviny_100"));
                        textSacharidy.setText
                        (rs.getString("Sacharidy_100"));

      textTuky.setText(rs.getString("Tuky_100"));

      textKcal.setText(rs.getString("Kcal_100"));
                    }
```

```
                rs.close();
                pst.close();

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

        }
    });
    scrollPane.setViewportView(mainTable);
```

Zajímavou funkcí všech tabulek v aplikaci je, že po kliknutí do tabulky se hodnoty z tabulky zobrazí v polích JTextField, které jsou vedle nich a to umožňuje rychlou a jednoduchou manipulaci s daty v SQLite tabulkách. Tato funkčnost je zajištěna metodou mouseClicked(MouseEvent arg0). Po kliknutí do příslušného řádku se spustí SQL příkaz select a hodnoty z tabulky se postupně pomocí metody setText(), uloží do příslušných polí.

```
    textBilkoviny = new JTextField();
    textBilkoviny.addKeyListener(new KeyAdapter() {
        public void keyTyped(KeyEvent arg0) {
            char c = arg0.getKeyChar();
                if (!(Character.isDigit(c) || (c ==
            KeyEvent.VK_DELETE) ||
                    (c == KeyEvent.VK_PERIOD) || (c ==
            KeyEvent.VK_BACK_SPACE))) {
```

```
                    Toolkit.getDefaultToolkit().beep();
                    arg0.consume();
                }
                if (textBilkoviny.getText().length() >= 15) {
                    Toolkit.getDefaultToolkit().beep();
                    arg0.consume();
            }
            }
        });
        textBilkoviny.setBounds(92, 59, 86, 20);
        panelMacro.add(textBilkoviny);
        textBilkoviny.setColumns(10);
```

Toto je ukázka toho jak je v aplikaci řešená validace jednotlivých polí, které může uživatel vyplňovat. V poli jsou povoleny pouze některé znaky, které jsou vybrány do podmínky if. Dále je povoleno jen určité množství znaků. Při porušení některé z podmínek se ozve zvukový signál a všechny další znaky se nebudou přidávat do pole, ale jsou odstraňovány metodou consume().

```
        JButton btnMacro = new JButton("Macro");
        btnMacro.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                panelMacro.setVisible(true);
                panelMain.setVisible(false);
                Jmeno = Login.JmenoUzivatele();
```

```
                Helpers.RefreshTable();
                Helpers.RefreshUserTable();
                Helpers.RefreshMacroZaDen();
            }
        });
        btnMacro.setBounds(212, 88, 100, 100);
        panelMain.add(btnMacro);
```

Ukázka jednoho z tlačítek na hlavní obrazovce. Změna obrazovek v hlavní části aplikace je provedena za použití různých JPanel nikoliv více JFrame. Přepínání mezi panely aplikace funguje pomocí tlačítek na úvodní obrazovce. Panely se zobrazují a mizí za použití metody setVisible().

4.2.6 Databáze

Aplikace využívá čtyři tabulky.

Tabulka Denik
Jsou zde atributy, které zachycují pokrok uživatele. Jak název napovídá, jedná se o deník uživatele.
Tabulka Macro
Tabulka, ve které jsou uloženy kalorické hodnoty jídel, kterou mohou používat všichni uživatelé pro zrychlení zápisu hodnot do osobní tabulky.
Tabulka UzivatelDen
V této tabulce jsou uloženy osobní informace o jídlech, které uživatel snědl za daný den.
Tabulka Uzivatele
Zde jsou uloženy informace o uživatelích, které slouží k přihlášení do aplikace.

5 Závěr

Programovací jazyk Java nabízí široké spektrum využití. Java lze použít pro naprogramování jednoduchých i složitých aplikací, od konzolových až po rozsáhlé desktopové aplikace. Java je i základním stavebním kamenem operačního systému Android, což přispívá k dalšímu rozšíření jazyka a zvýšení jeho popularity. Java se v posledních letech drží na prvních místech nejpopulárnějších programovacích jazyků, což dokazují pravidelné výpočty popularity od společnosti TIOBE. Vývojáři Java aplikací mají nespočetně mnoho různých řešení pro návrh aplikací a záleží jen na nich a jejich zkušenostech, které z nich si zvolí pro vývoj svojí aplikace. Java je snadno uchopitelná i pro začínající vývojáře díky její rozsáhlé dokumentaci, kde se mimo jiné uvádějí i doporučené možné způsoby řešení problémů. Podle mého názoru je programovací jazyk Java všestranným jazykem pro vývoj malých i velkých aplikací.

6 Seznam použité literatury

Tištěné zdroje:

(1) KOEGH, J. Giannini M. OOP Objektově orientované programování bez předchozích znalostí - Průvodce pro samouky, Computer Press, 2006, ISBN: 80-251-0973-9.

(2) KOEGH, James. Java bez předchozích znalostí: průvodce pro samouky. Vyd. 1. Brno: Computer Press, 2005, 274 s. ISBN 80-251-0839-2.

(3) SCHILDT, Herbert. 2014. Mistrovství - Java: Kompletní průvodce vývojáře. 1. vyd. Brno: Computer Press. ISBN 978-80-251-4145-

(4) 6. SCHILDT, Herbert. 2012. Java 7: výukový kurz. 1. vyd. Brno: Computer Press, 664 s. ISBN 978-80-251-3748-2.

(5) DARWIN, I., F. Java - Kuchařka programátora, Computer Press, 2006, 800 s. ISBN: 80-251-0944-5

(6) HEROUT, P. Učebnice Jazyka Java. 2. vyd. České Budějovice : KOPP, 2006. 349 s. ISBN 80-7232-115-3.

(7) HEROUT, Pavel. 2007. Java: grafické uživatelské prostředí a čeština. 2. vyd. České Budějovice: Kopp, 316 s. ISBN 978-80-7232-328-9

Online zdroje:

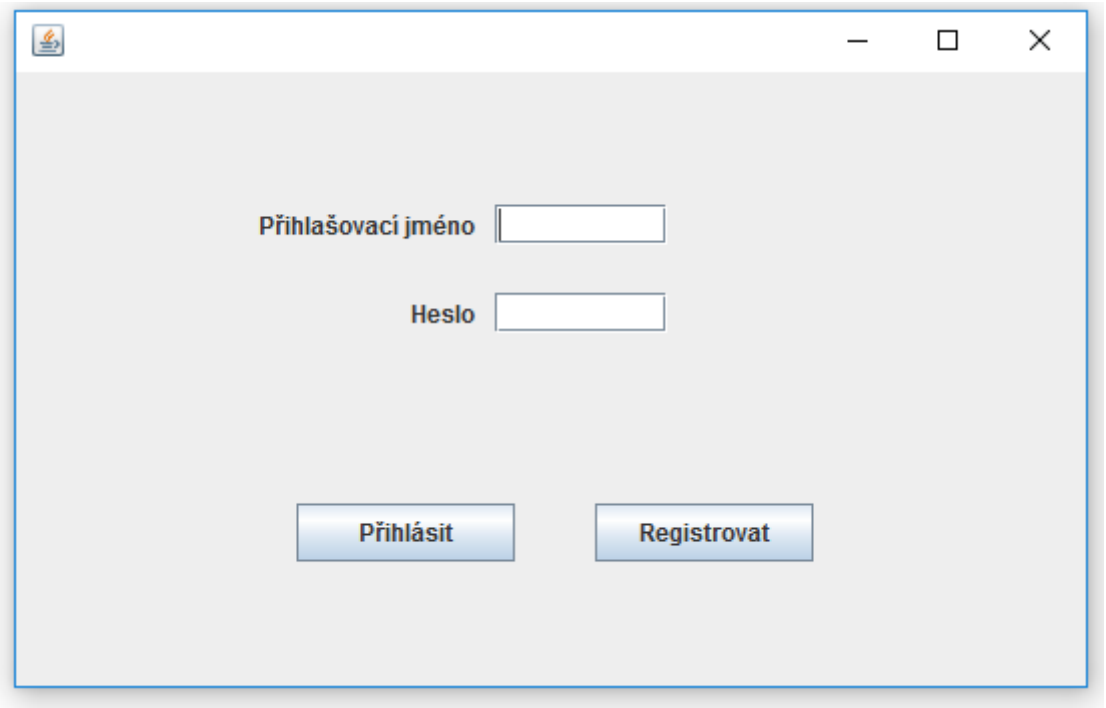
(8) RAJPUT, Dinesh. Java Versions, Features and History [online]. [cit. 2015-09-14]. Dostupné z: http://www.dineshonjava.com/2013/01/java-versions-features-andhistory.html#.VfaxNhHtlBe

(9) 17. KULANDAI, Joseph. Java Versions, Features and History [online]. [cit. 2015-09- 14]. Dostupné z: http://javapapers.com/core-java/java-features-and-history/

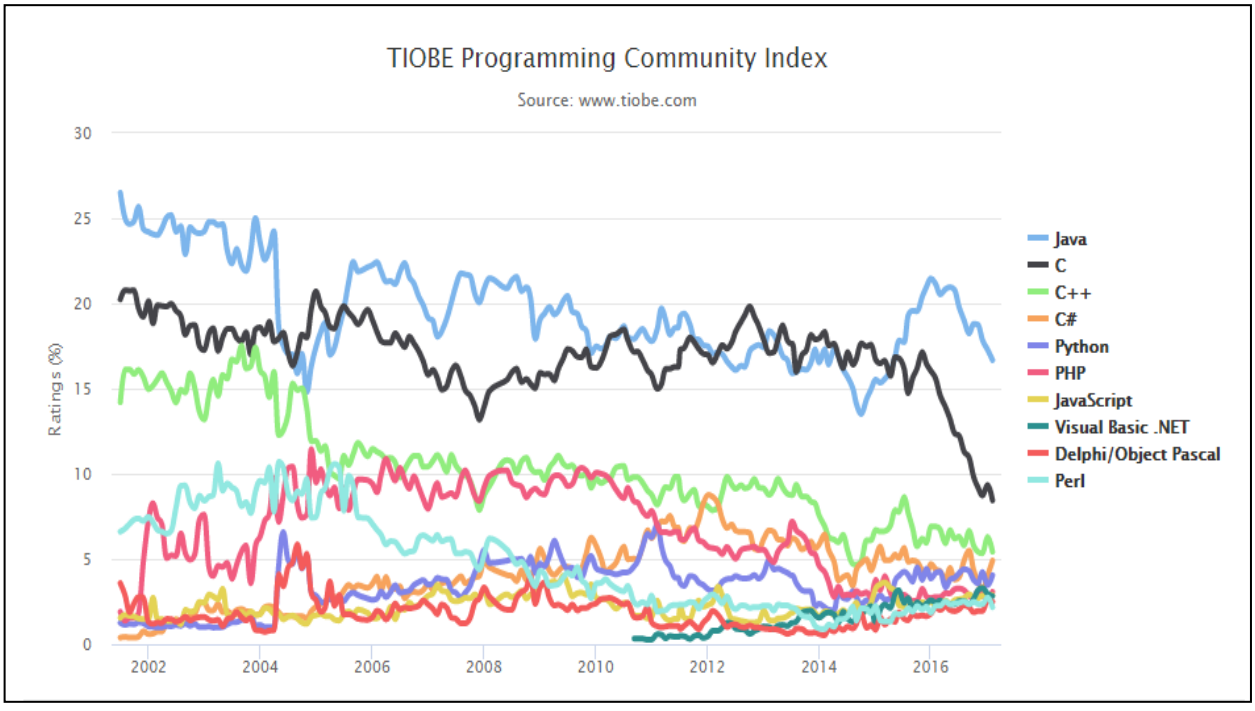
(10) Java Timeline [online]. [cit. 2015-09-14]. Dostupné z: http://oracle.com.edgesuite.net/timeline/java/

(11) TIOBE: The software quality company [online]. [cit. 2016-02-22]. Dostupné z: http://www.tiobe.com/tiobe_index?page=index

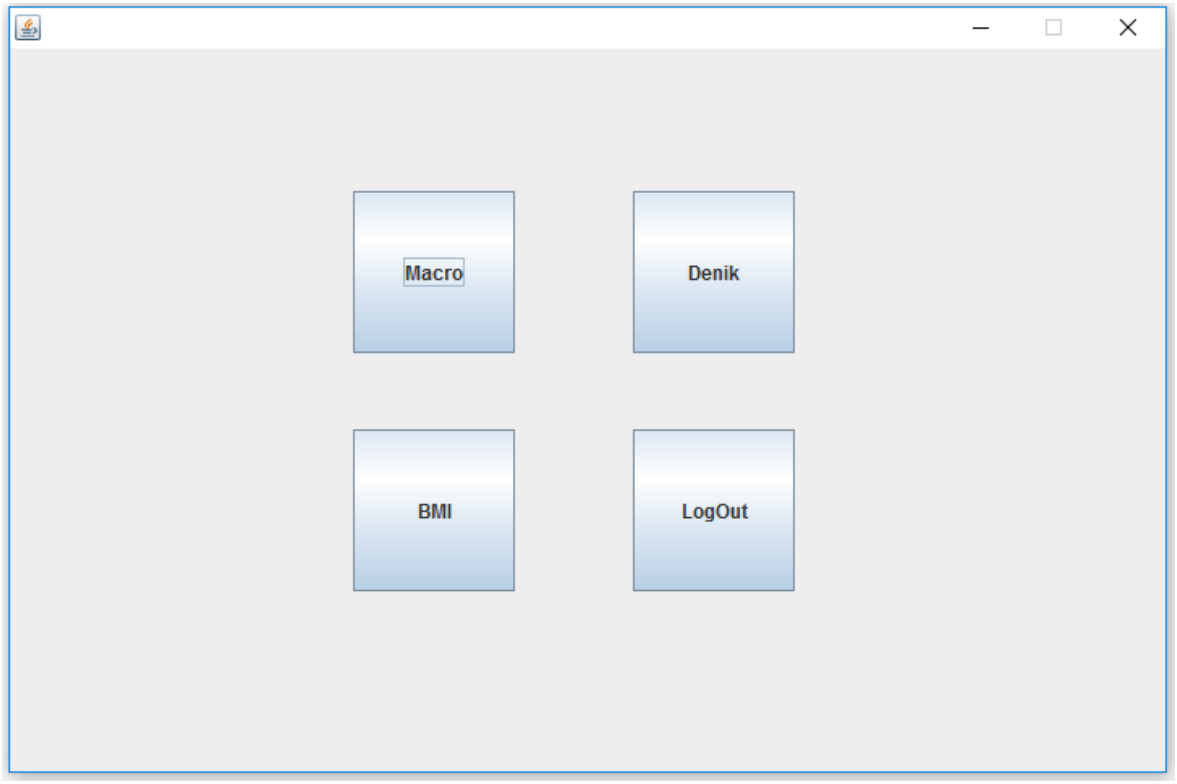
7 Seznam obrázků



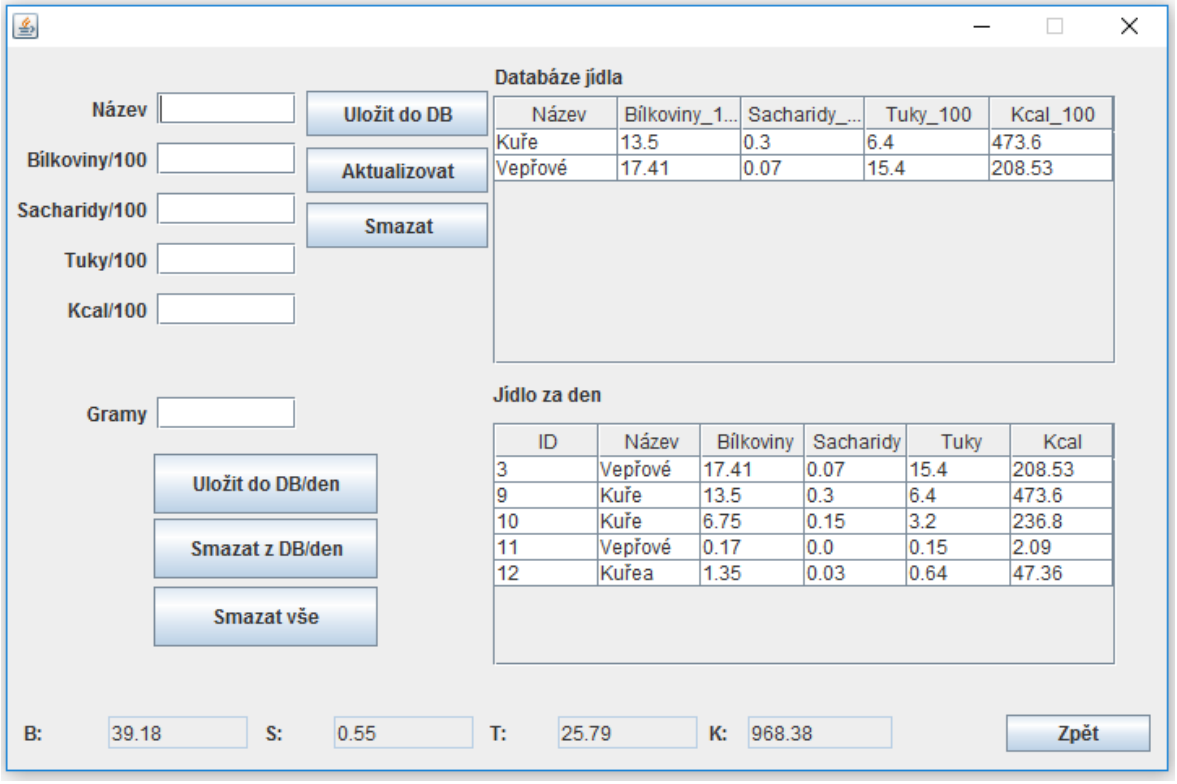
Obrázek 1 Přihlašovací obrazovka



Obrázek 2 : Graf indexů popularity programovacích jazyků (11)



Obrázek 3 Úvodní obrazovka



Obrázek 4 Obrazovka pro výpočet nutričních hodnot