Eliška Roubalová

JAVA

BEZ PŘEDCHOZÍCH ZNALOSTÍ

Datové typy, operátory, řídicí struktury

Třídy, metody, dědičnost a polymorfizmus

Pole, řetězce, vstup a výstup programu

Tvorba grafického uživatelského rozhraní

computer press®

Eliška Roubalová

Java
bez předchozích znalostí

Computer Press
Brno
2015

Java
bez předchozích znalostí

Eliška Roubalová

Obálka: Martin Sodomka
Odpovědný redaktor: Martin Herodek
Technický redaktor: Jiří Matoušek

Objednávky knih:
http://knihy.cpress.cz
www.albatrosmedia.cz
eshop@albatrosmedia.cz
bezplatná linka 800 555 513
ISBN 978-80-251-4572-2

Vydalo nakladatelství Computer Press v Brně roku 2015 ve společnosti Albatros Media a. s.
se sídlem Na Pankráci 30, Praha 4. Číslo publikace 19 276.

1. vydání

ALBATROS MEDIA a.s.

Obsah

Úvod

Pokud čtete tuto knihu, zřejmě jste se rozhodli proniknout do tajů programování, a to konkrétně programování v jazyce Java. Budete potřebovat trpělivost, důslednost a pevné nervy, protože záhady programování mohou být pro začátečníky i pro zkušené programátory někdy tvrdý oříšek. Přesto je programování také zábava a pocit, když se vám podaří napsat první funkční program, se dá přirovnat k prvním přečteným slovům.

Základy programování

První kapitoly této knihy se věnují základním principům programování, které jsou s menšími obměnami v zápisu shodné pro téměř všechny programovací jazyky. Důležité je hlavně pochopit obecné principy. Pro příklady v této knize si zřejmě vystačíte, i pokud se naučíte zápis zpaměti, ale při dalším pokračování vám bude toto porozumění bolestně chybět.

Objektový přístup

V druhé části knihy se seznámíte se základy takzvaného objektového programování, což je základní stavební prvek Javy. Opět staví na obecných principech využitelných i v jiných jazycích, Java je však na tento styl programování přímo zaměřena a poměrně přísně ho vyžaduje. Složitě znějící principy jsou vysvětleny na příkladech z běžného života, a pokud jejich studiu věnujete potřebný čas a patřičné soustředění, jistě nebudete mít s jejich pochopením problém. Na konci knihy je pak kapitola popisující základní prvky grafického uživatelského rozhraní.

Procvičování

Na konci každé kapitoly najdete část nazvanou **Opakování**. Ta slouží k procvičení znalostí probraných v dané kapitole. Ze začátku jde především o opakování teoretických znalostí formou otázek, později následuje část **Vyzkoušejte si**, ve které najdete praktické úkoly k procvičení. Odpovědi na otázky i řešení jednotlivých cvičení naleznete v první příloze knihy. Druhou přílohou knihy je závěrečný test. Ten spolu s třetí přílohou, závěrečným projektem, slouží k ověření nabytých znalostí a dovedností.

Zvláštní odstavce

V celé knize můžete narazit na zvýrazněné odstavce, které mají speciální význam.

Poznámka: Poznámky zpravidla obsahují nějaké doplňující informace. Mohou vám pomoci lépe pochopit vysvětlovanou problematiku nebo se v nich dozvíte něco nad rámec této knihy. Také se mohou týkat informací z jiných oblastí, zejména IT, které by se vám mohli hodit.

Důležité: Takto označené odstavce obsahují informace nějakým způsobem klíčové buď pro pochopení daného problému, nebo pro další výuku programování. Také vás mohou varovat před možnými problémy a upozorňovat na ne vždy intuitivní chování programovacího jazyka.

Nápověda: Nápověda, se vyskytuje v poslední příloze knihy, kterou je závěrečný projekt. Najdete zde rady a tipy, jak řešit ty kroky v zadání, které mohou být obtížnější než ostatní.

Zpětná vazba od čtenářů

Nakladatelství a vydavatelství Computer Press, které pro vás tuto knihu připravilo, stojí o zpětnou vazbu a bude na vaše podněty a dotazy reagovat. Můžete se obrátit na následující adresy:

Computer Press
Albatros Media a.s., pobočka Brno
IBC
Příkop 4
602 00 Brno

nebo

sefredaktor.pc@albatrosmedia.cz

Computer Press neposkytuje rady ani jakýkoli servis pro aplikace třetích stran. Pokud budete mít dotaz k programu, obraťte se prosím na jeho tvůrce.

Errata

Přestože jsme udělali maximum pro to, abychom zajistili přesnost a správnost obsahu, chybám se úplně vyhnout nelze. Pokud v některé z našich knih najdete chybu, ať už chybu v textu nebo v kódu, budeme rádi, pokud nám ji oznámíte. Ostatní uživatele tak můžete ušetřit frustrace a pomoci nám zlepšit následující vydání této knihy.

Veškerá existující errata zobrazíte na adrese *http://knihy.cpress.cz/K2184* po klepnutí na odkaz Soubory ke stažení.

KAPITOLA 1

Na úvod o Javě

V této kapitole:
- Počítačový program
- Vysokoúrovňový programovací jazyk
- „Univerzální" Java
- Potřebná softwarová výbava
- Vývojová prostředí
- Nový program v Javě
- Běžné chyby
- Opakování

Dříve než se vrhnete do samotného psaní programů a vyzkoušíte si praktické příklady, je rozhodně užitečné seznámit se s trochou teorie. Například: Proč programovat právě v Javě? Co je to počítačový program a programovací jazyk? Jak získat potřebné nástroje? V této úvodní kapitole najdete odpovědi na tyto i další otázky. Pokud již zmíněné základy znáte a potřebné nástroje máte nainstalované, tuto kapitolu klidně přeskočte. Můžete se k ní v případě potřeby kdykoli vrátit.

Počítačový program

Počítač sám o sobě je vlastně jen hromádka technického vybavení (takzvaný hardware). To, co z počítačů dělá možná jednu z nejdůležitějších věcí, které vlastníte, jsou právě programy (takzvaný software). Ty obsahují programátorem zadané instrukce, jak se má pracovat s dostupnými prostředky. Pracovní jednotky počítačů ani jiné techniky samozřejmě nerozumí pokynům v běžných jazycích. Pokyny se skládají dohromady z primitivních instrukcí typu „sečti číslo A s číslem B". Zejména proto bylo dříve programování vyhrazeno pouze lidem s nejvyšším technickým vzděláním, kteří ovládali takzvaný **strojový kód**. Později byl vytvořen **jazyk symbolických adres**, který už nevyžadoval zápis přímo pomocí nul a jedniček, ale používal předem určené základní instrukce spolu s adresami používaných registrů. Ani to ale k většímu rozšíření mezi programátory samozřejmě nestačilo.

```
404e40:   afb00010    sw  s0,16(sp)
404e44:   0080a021    move    s4,a0
404e48:   00a09821    move    s3,a1
404e4c:   00008021    move    s0,zero
404e50:   27a50008    addiu   a1,sp,8
404e54:   24060001    li  a2,1
```

Obrázek 1.1 Příklad zápisu kódu, kterému by rozuměl počítač, ale určitě ne běžný uživatel

Vysokoúrovňový programovací jazyk

Programovací jazyk je formální jazyk, který obsahuje sadu pravidel pro zápis příkazů. Protože lidí ovládajících výše zmíněný strojový kód překvapivě není většina, vznikly postupem času vysokoúrovňové (vyšší) programovací jazyky. Ty umožňují zapisovat příkazy pomocí běžně používaných výrazů, většinou anglických. Napsaný program se teprve potom přeloží do strojového kódu. Téměř všechny programovací jazyky, se kterými se dnes běžně setkáte, jsou právě tohoto typu – například jazyky Java, C, Python a mnoho dalších. Právě se vznikem vysokoúrovňových programovacích jazyků zažilo programování svůj první boom.

Překlad programu

V předchozím odstavci bylo zmíněno, že program ve vysokoúrovňovém programovacím jazyce je nutné před spuštěním nejdříve přeložit. V praxi to znamená, že jiný program nazvaný **kompilátor** (překladač) jednotlivé soubory programu převede do **objektového kódu**. Tento mezikrok generující obvykle strojové instrukce se nazývá kompilování. Následuje takzvané **linkování**, při kterém jsou spojeny a přeloženy do strojového kódu jednotlivé soubory tvořící program. V praxi se běžně setkáte s výrazem kompilování pro souhrn obou činností.

Pořád ale zůstávala jedna zásadní překážka. Stejně jako různí lidé mluví různými jazyky, i různé typy počítačů rozumí různým formám strojového kódu (a různým zápisům instrukcí). Každý typ počítače tedy musel mít vytvořen vlastní kompilátor a spojovač. Programy napsané a přeložené pro jeden typ počítačů na jiných počítačích nefungovaly.

„Univerzální“ Java

S rozšiřujícím se rozsahem počítačů začala nabývat na síle potřeba, především komerční sféry, vytvoření jazyka, jehož programy nebude nutné při každém přenosu na jiný typ počítače znovu zkompilovat (přeložit). Projekt, který vedl ke vzniku Javy, byl odstartován v roce 1991 společností **Sun Microsystems** a jeho výsledný název Java je prý převzat

podle kávy, kterou jeho tvůrci pili (Java coffee). Původně bylo předpokládáno využití Javy pro programování spotřební elektroniky (mikrovlnky, ledničky…).

Největší motivaci pro vývoj Javy však představovalo masové rozšíření internetu. Z uzavřené sítě pro akademiky a vojáky se stal fenomén měnící způsob komunikace mezi lidmi i jejich chování. Statické stránky vytvářené pouze pomocí HTML kódu (ten určuje, jak prohlížeč zobrazí jednotlivé části stránky) brzy přestaly webdesignérům stačit a pro prosazující se stránky s dynamickým obsahem byl jazyk Java ideální volbou.

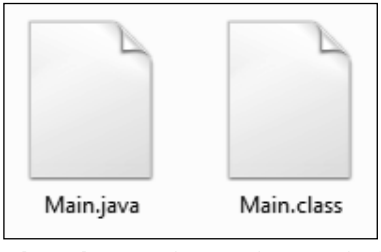
Poznámka: Statické webové stránky se zobrazují tak, jak jsou uloženy na serveru, a všem návštěvníkům stejně. Pokud například hledáte encyklopedii hub, stránky mohou být statické, protože tyto informace jsou pro všechny shodné a prakticky se nemění. Většina dnes používaných služeb jsou dynamické stránky, které každému návštěvníkovi generují obsah „na míru" podle jeho požadavků. Například účty na sociálních sítích jsou z principu dost odlišné a každému musí být vygenerována stránka s jeho účtem podle uložených informací.

Na počátku přenositelnosti

Několikrát už bylo zmíněno, že jedna z největších výhod Javy spočívá v přenositelnosti jejích programů. Nyní je tedy zapotřebí vysvětlit, jak vlastně Java v základu funguje. Na počátku všeho stojí pochopitelně člověk – programátor se svým nápadem a potřebnými znalostmi. Ten zapisuje instrukce programovacího jazyka a vytváří strukturu programu. Již víte, že program nemusí být celý v jednom souboru, ale může být rozdělen na části a logické celky. Tyto soubory mají příponu *.java* a označují se jako **zdrojový kód** programu.

Tajemství úspěchu – bajtový kód

Kompilování vysokoúrovňových jazyků do strojového kódu už pro vás není žádným tajemstvím. Jazyky založené na tomto principu však vyžadují implementovat pro každý typ počítače vlastní kompilátor, což je zdlouhavá, náročná a drahá záležitost. Tvůrci Javy proto přišli s novým řešením, program se ze zdrojového kódu nekompiluje přímo do toho strojového, ale do silně optimalizovaného mezikódu nazvaného **bajtový kód**. Soubory v bajtovém kódu poznáte podle přípony *.class*.



Obrázek 1.2 Soubor ve zdrojovém kódu a stejný soubor přeložený do bajtového kódu

Bajtový kód je následně zpracován **virtuálním strojem** jazyka Java (JVM – *Java Virtual Machine*), který ho **interpretuje** neboli přímo vykonává zapsané instrukce. Tento virtuální stroj je implementován pro každý typ počítačů, jeho implementace je ale podstatně snazší a levnější než vytváření vlastního kompilátoru. Protože všechny virtuální stroje rozumí stejnému bajtovému kódu, stačí pro všechny typy počítačů kompilátor jen jeden. Virtuální stroj Javy přináší i další výhody, například zvyšuje bezpečnost programů, hlídá totiž za programátora některé potenciálně nebezpečné operace a stará se i o bezpečnou správu paměti.

Poznámka: Programovací jazyky se také dělí na kompilované (překládané) a interpretované. Kompilované jazyky jsou nejprve přeloženy do strojového kódu, který je následně vykonáván. Interpretované jazyky tento mezikrok postrádají a jejich instrukce jsou vykonávány v reálném čase bez předchozího překladu. Oba typy mají samozřejmě své výhody i nevýhody. Přímé vykonávání instrukcí může být rychlejší, ovšem postrádá kontroly a optimalizace, které může kompilátor nabídnout.

Potřebná softwarová výbava

Abyste mohli začít programovat v Javě a své programy kompilovat a spouštět, budete potřebovat sadu **Java Development Kit** (JDK). Tuto sadu si můžete zcela zdarma stáhnout ze stránek společnosti Oracle (ta v roce 2010 získala společnost Sun Microsystems) na adrese *http://www.oracle.com/technetwork/java/javase/downloads/index.html*.

V době psaní této knihy byla poslední verzí JDK 8, kterou využívá Java SE (Standard Edition) 8. Všechny programy uvedené v této knize bez problémů spustíte i s verzí JDK 7 pro vydání Java SE 7. Vzhledem k tomu, že je tato kniha zaměřena na základní prvky jazyka, neměli byste se s žádnými většími problémy setkat, ani pokud používáte starší verzi Javy (Java 6…). Je však možné, že některé programy s novějšími prvky Javy nepůjdou zkompilovat a spustit, zvláště pokud budete postupovat dále k pokročilým prvkům.

Podle zvolené verze klepněte na správný odkaz ke stažení Java Platform JDK (Download JDK). Potvrďte, že souhlasíte s licenčními podmínkami společnosti Oracle (Accept License Agreement), a stáhněte si verzi JDK vhodnou pro váš operační systém. Instalací vás pak provede připravený průvodce.

Spouštění s využitím příkazového řádku

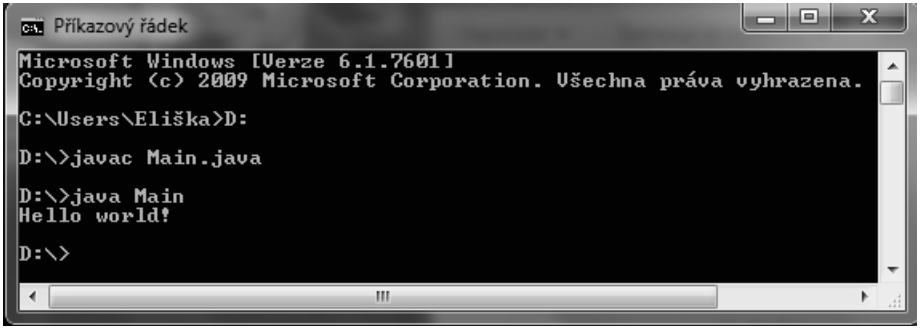
Nyní již můžete využívat programy stažené v této sadě s pomocí příkazového řádku. Pokud nechcete z nějakého důvodu instalovat další aplikaci s grafickým uživatelským prostředím, vystačíte si jen s obyčejným textovým editorem. Při ukládání vašich zdrojových souborů ale nezapomeňte změnit jejich příponu na *.java*. Ke zkompilování vytvořeného programu napíšete do příkazového řádku následující pokyn a stisknete klávesu Enter:

```
javac JmenoSouboru.java
```

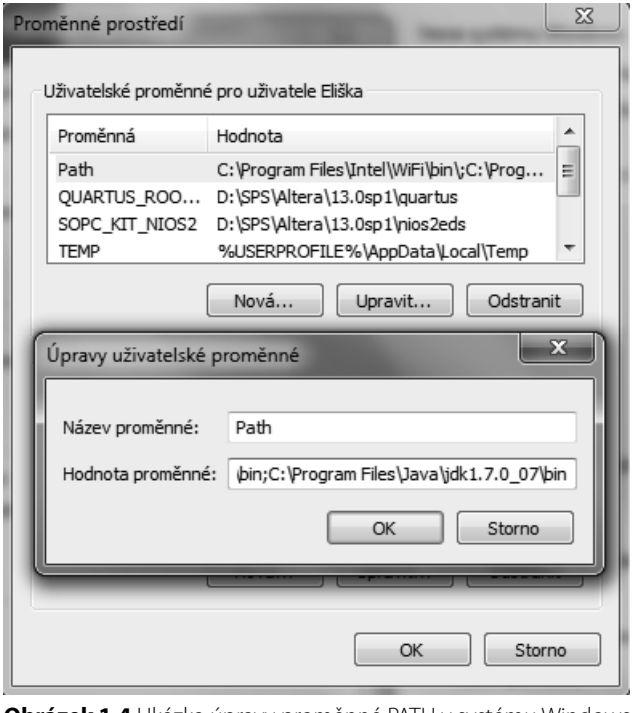
Po správně proběhlém zkompilování přibude ve složce soubor v bajtovém kódu, který bude mít stejný název a příponu *.class*. Ten využijete ke spuštění programu pomocí virtuálního stroje: do příkazového řádku napíšete následující pokyn a opět stisknete klávesu Enter:

```
java JmenoSouboru
```

Všimněte si, že přípona *.class* se u tohoto příkazu neuvádí. Pokud jste instalaci provedli správně a nevyskytly se další chyby, měl by se nyní váš program spustit.



Obrázek 1.3 Spuštění programu vypisujícího text „Hello world!" uloženého v souboru Main.java



Obrázek 1.4 Ukázka úpravy proměnné PATH v systému Windows

Důležité: Je možné, že při pokusu o kompilaci počítač nebude schopen nalézt program *javac* nebo *java*, ačkoli jste sadu JDK správně nainstalovali. Pravděpodobně bude nutné stanovit cestu k těmto nástrojům. V systému Windows to znamená přidat jejich adresu do proměnné prostředí PATH. Jak upravit proměnnou prostředí se nejlépe dozvíte z dokumentace vašeho operačního systému. Budete potřebovat znát cestu k adresáři *bin*. Například při instalaci do výchozího adresáře v systému Windows je cesta většinou *C:\ProgramFiles\Java\jdk1.7.0_07\bin*; raději si ale přesnou cestu sami zkontrolujte. Tuto cestu přidejte do příslušné proměnné.

Vývojová prostředí

V době grafických aplikací a dotykového ovládání už jen málokomu vyhovuje práce s příkazovým řádkem, zvláště v systémech Windows. I programátoři oceňují při práci pohodlí a možnosti, které jim nabízí používání různých vývojových prostředí neboli **IDE – Integrated Development Environment**. Kolik je programátorů, tolik je i názorů na to, které vývojové prostředí je to nejlepší. Mezi nejčastěji používaná vývojová prostředí, která můžete vyzkoušet i vy, určitě patří ta následující.

NetBeans

Zcela zdarma si můžete na webových stránkách *netbeans.org/downloads* stáhnout prostředí NetBeans, původně studentský projekt na Karlově univerzitě. Jeho komerční verzi odkoupila společnost Sun Microsystems a v jeho vývoji dále pokračuje i společnost Oracle. Můžete si vybrat balíček podle nástrojů, které potřebujete, samozřejmostí je také podpora různých operačních systémů. Pro začátek vám bude stačit balíček Java SE. V prostředí se snadno orientuje a snadno se používá. Pro začínající uživatele může být NetBeans o něco pohodlnější, nemá ale tak širokou rozšiřitelnost zásuvnými moduly (plug-iny) jako Eclipse a programátoři přecházející na jazyk C také často preferují druhé zmíněné prostředí.

Eclipse

Prostředí Eclipse si můžete také stáhnout zdarma, a to ze stránek *www.eclipse.org/downloads*. Na výběr máte z široké škály balíčků uzpůsobených pro různé potřeby i z podpory různých operačních systémů. Můžete si vybrat verzi pro Windows, Linux i Mac OS, pro začátek vám jistě bude stačit Eclipse Standard. Jedná se o široce využívané prostředí, které si ovšem jde svou cestou a pro začínající uživatele může být matoucí a nepohodlné. Na jeho ovládání si zřejmě budete muset chvíli zvykat. Na druhou stranu Eclipse vyniká dostupností různých zásuvných modulů (plug-inů), a tudíž nepřebernými možnostmi rozšíření.

Poznámka: Ať už budete programy spouštět pomocí příkazového řádku, nebo si oblíbíte některé vývojové prostředí, všechny příklady v této knize by měly fungovat stejně a v jejich zápisu není žádný rozdíl. Vývojová prostředí však navíc umí generovat nejčastěji používané části kódu za vás a tím vám mohou ušetřit hodně práce. Jestliže se pro některé z nich rozhodnete (což vám mohu jen doporučit), prohlédněte si pečlivě jeho prostředí a položky menu, jistě pro vás potom nebude žádný problém nalézt příkazy k provedení mnoha akcí popsaných dále i bez uvedení přesného umístění.

Nový program v Javě

Před ukončením první kapitoly je ještě vhodné doplnit, jak vytvoříte nový program v jazyce Java. Najdete zde postup pro tři zmíněné možnosti:

- použití textového editoru a příkazové řádky
- použití vývojového prostředí NetBeans
- použití vývojového prostředí Eclipse

Pokud chcete použít jiné vývojové prostředí, nic vám v tom nebrání, budete ale muset tento základ zvládnout sami. V dalších kapitolách bude tato znalost předpokládána.

Nový program v textovém editoru

Bylo již zmíněno, že můžete použít libovolný textový editor. Vystačíte si i s Poznámkovým blokem nebo například s editorem WordPad. Postupujte následovně:

1. Spusťte zvolený textový editor.
2. Do prázdného dokumentu napište následující kód:

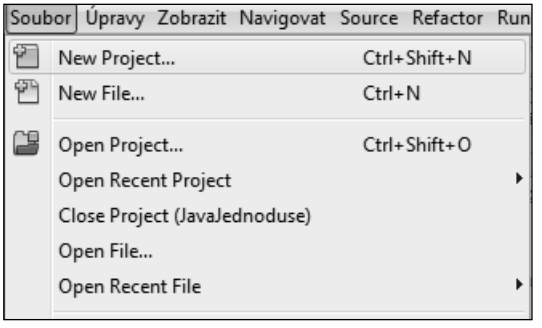
```
public class AhojSvete {
    public static void main (String[] args) {
        System.out.println("Ahoj svete!");
    }
}
```

3. Při ukládání pojmenujte soubor *AhojSvete* a změňte jeho příponu na *.java*.
4. Přeložte a spusťte program podle postupu uvedeného dříve v této kapitole.

Důležité: Význam jednotlivých částí kódu bude vysvětlen dále v této knize. Prozatím stačí, abyste si zapamatovali, že je vždy nutné uvést řádek `public class VášZvolenýNázev` a řádek `public static void main (String[] args)` přesně v této podobě. Všechny levé složené závorky musí mít své ukončovací pravé závorky. Jméno ukládaného souboru musí mít tvar *VášZvolenýNázev.java*. Vaše příkazy se zapisují na místo řádku `System.out.println();` mezi vnitřní složené závorky.

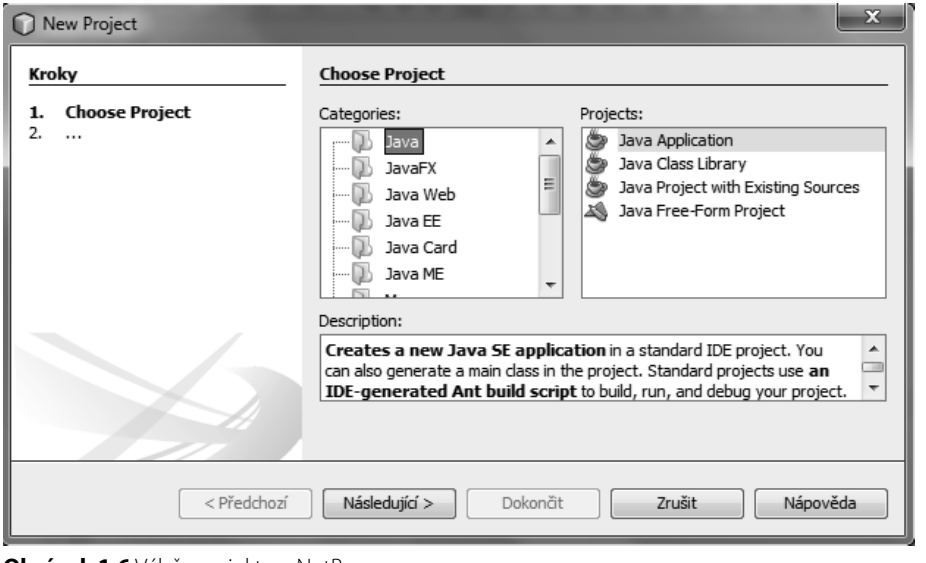
Nový program v prostředí NetBeans

Vývojové prostředí NetBeans používá pro psaní programů takzvané projekty. Ty vám umožňují zachovat si přehled o struktuře vašeho programu a nabízejí snazší manipulaci s jeho soubory. Nový program vytvoříte podle následujících kroků:



Obrázek 1.5 Založení nového projektu v NetBeans

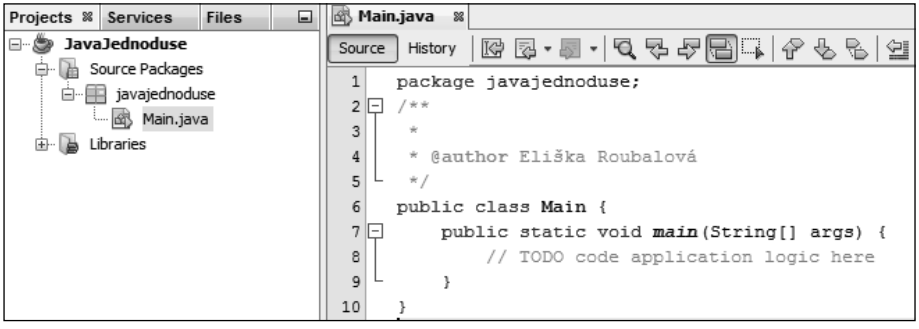
1. Založte nový projekt buď pomocí volby v nabídce **Soubor** → **Nový Projekt** (**New Project**), nebo klepnutím na tlačítko **Nový Projekt** (**New Project**) na panelu nástrojů, nebo stisknutím klávesové zkratky **Ctrl+Shift+N**.



Obrázek 1.6 Výběr projektu v NetBeans

2. Zvolte kategorii projektu **Java** a projekt **Java Application** (**Aplikace v Javě**).
3. V následujícím dialogovém okně zvolte název projektu (**Project Name**) a případně vyberte jeho umístění. Můžete ponechat základní nastavení, které většinou vytvoří

ve složce *Dokumenty* novou složku *NetBeansProject* a do ní ukládá vytvořené projekty. Zaškrtněte pole **Vytvořit hlavní třídu** (**Create Main Class**) a případně zvolte její jméno. Dejte však pozor, abyste neupravili část jména před tečkou. Tu tvoří název základního balíčku, vaše pojmenování hlavní třídy následuje až za tečkou. Pro cvičné účely se často volí jednoduše pojmenování `Main`.



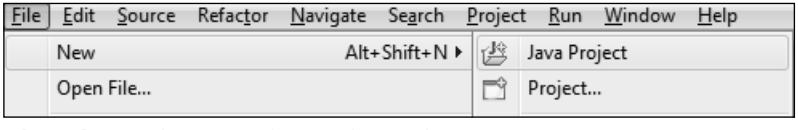
Obrázek 1.7 Vytvořený projekt v NetBeans

4. V části se seznamem projektů nyní přibyl váš projekt. Po rozbalení seznamu souborů vidíte ve složce **Zdrojové balíčky** (**Source packages**) váš balíček pojmenovaný podle názvu aplikace a v něm soubor **Main.java** (podle vámi zvoleného jména). Po jeho otevření vidíte, že NetBeans za vás automaticky vygeneroval potřebné části kódu, a vy můžete proto rovnou začít psát vlastní program.

Pro spuštění programu otevřete nabídku **Spustit** (**Run**) a u položky **Nastavit hlavní projekt** (**Set Main Project**) vyberte ten váš. Pak už stačí na panelu nástrojů nebo opět v nabídce **Spustit** (**Run**) klepnout na volbu **Spustit hlavní projekt** (**Run Main Project**). Váš program se zkompiluje a spustí automaticky.

Nový program v prostředí Eclipse

Také vývojové prostředí Eclipse používá pro lepší správu souborů programu projekty. Vytvoření nového programu je tak podobné jako ve vývojovém prostředí NetBeans:



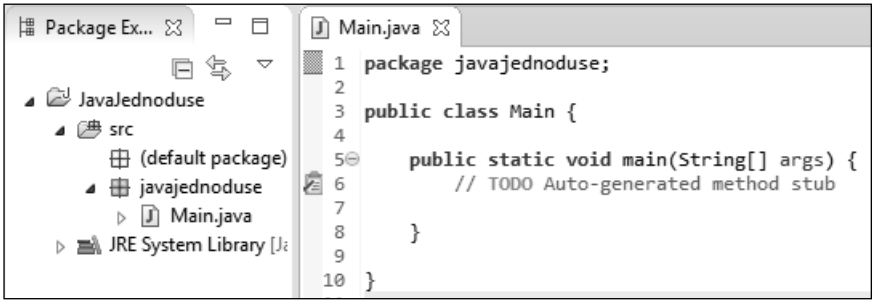
Obrázek 1.8 Založení nového projektu v Eclipse

1. Založte nový projekt. Můžete opět použít volbu v nabídce **Soubor** (**File**) → **Nový** (**New**) → **Java Project** nebo rozevírací seznam u tlačítka **Nový** (**New**) na panelu nástrojů.

2. Zvolte název projektu (**Project Name**) a případně složku pro ukládání projektu (**Location**) po odškrtnutí políčka **Use default location**. Ostatní nastavení zatím ponechte beze změn i v následujícím dialogovém okně.
3. V části **Projekty** se vám objeví vámi vytvořený projekt obsahující složku **src**. Klepněte na ni pravým tlačítkem myši a ze zobrazivší se nabídky vyberte **Nový** (**New**) → **Třída** (**Class**).

Obrázek 1.9 Vytvoření nové třídy v Eclipse

4. Vložte vámi zvolené jméno, případně jméno balíčku, do kterého chcete třídu umístit, a zaškrtněte pole **public static void main(String[] args)**. Ostatní nastavení ponechte.
5. Do projektu přibyla vámi vytvořená třída s automaticky vygenerovaným kódem.



Obrázek 1.10 Vytvořený projekt v Eclipse

Pro spuštění vašeho programu klepněte na panelu nástrojů na tlačítko **Spustit** (**Run**) nebo vyberte možnost **Spustit** (**Run**) v příslušné nabídce. Program se zkompiluje a spustí automaticky.

Běžné chyby

Při psaní prvních zkušebních programů se mohou vyskytnout některé chyby, které sice vypadají na první pohled složitě a závažně, ale dají se mnohdy velice jednoduše odstranit. Některá chybová hlášení, se kterými se můžete setkat, jsou například tato:

- **‚javac' / ‚java' is not recognized as an internal or external command, operable program or batch file** – Je možné, že jste nenainstalovali staženou sadu JDK nebo že vám chybí cesta k programům *javac* a *java* v proměnné prostředí PATH. Tento problém byl popsán už v části *Spuštění pomocí příkazového řádku*.
- **javac: file not found: NazevSouboru.java** – Zřejmě se snažíte zkompilovat soubor, který se v daném umístění nenachází. Zkontrolujte, že je ve složce, ve které se právě nacházíte, i soubor *NazevSouboru.java*.
- **Error: Class names ‚NazevSouboru', are only accepted if annotation processing is explicitly requested** – Pokusili jste se nejspíš zkompilovat soubor bez uvedené přípony *.java*.
- **Error: Could not find or load main class NazevSouboru** – Možná se pokoušíte spustit program bez předchozí kompilace, ve složce se nenachází soubor *NazevSouboru.class*. Nebo jste při zadávání příkazu zapsali navíc i příponu souboru (při spouštění se přípona nikdy neuvádí).
- **Error: Class Nazev is public, should be declared in a file named Nazev.java** – Váš název uvedený v programu za identifikátory `public class` se neshoduje s názvem souboru. Pokud jste v programu například uvedli `public class Ahoj`, musí být tento program uložen v souboru s názvem *Ahoj.java*.

Opakování

Dokážete odpovědět na následující otázky? Správné odpovědi najdete v příloze na konci knihy.

1. Jaký je rozdíl mezi vysokoúrovňovými jazyky a strojovým kódem?
2. K čemu slouží takzvaný kompilátor?
3. V jakém typu kódu jsou soubory s příponou .java?
4. Kolik typů bajtového kódu dokáže interpretovat virtuální stroj Javy?
5. Jaká je hlavní výhoda programů napsaných v Javě?
6. Jaký pokyn slouží ke kompilaci programu v příkazovém řádku?

Jestliže jste při odpovídání výrazněji neváhali a měli jste alespoň přibližnou představu o tom, jak Java funguje, můžete bez obav pokročit k následující kapitole.

KAPITOLA 2

Proměnné a datové typy

V této kapitole:
- Primitivní datové typy
- Proměnné
- Opakování

Mezi základní dovednosti každého programátora bezesporu patří dobrá znalost datových typů. Ta vám umožní efektivní využívání proměnných a v neposlední řadě vyvarování se zdlouhavého hledání chyb. Nesprávné použití datových typů proměnných totiž může vést i k na první pohled překvapivým výsledkům. Ale co vlastně znamenají ty datové typy, o kterých se pořád mluví? Jak a k čemu se používají proměnné? To se dozvíte v této kapitole.

Primitivní datové typy

Datové typy především usnadňují práci programátorům. Určují, kolik místa v paměti bude daná proměnná zabírat, abyste se o to nemuseli starat sami. Java je navíc jazyk se **silnou typovou kontrolou**, při kompilaci programu se tedy projdou proměnné a zkontroluje se, zda jsou v nich uložená data toho správného typu. To může ušetřit cenný čas při hledání chyb. Moderní vývojová prostředí vás navíc na případné chyby umí upozornit už při psaní díky průběžnému kompilování kódu.

Primitivní neboli základní datové typy lze rozdělit na číselné, znakové a logické. Samy názvy dobře napovídají, jaká data tyto typy zastupují.

!

Důležité: Tyto datové typy se nazývají primitivní proto, že je Java nechápe jako objekty. O objektech se více dozvíte v dalších kapitolách. Pro teď stačí, když si zapamatujete, že každý z primitivních typů má svou obalující třídu a v případě shodnosti názvu je navzájem odlišíte pomocí počátečního písmena. Primitivní typy vždy musí začínat malým písmenem a obalující třídy písmenem velkým.

Číselné datové typy

Pro správné porozumění rozdílům mezi jednotlivými číselnými typy je potřeba si zopakovat některé základní znalosti ze střední školy. Čísla lze rozdělit do několika množin, které se navzájem překrývají, a tudíž čísla z nižší množiny vždy zároveň patří i do všech

vyšších množin. Nás budou zajímat dvě základní množiny, a to **celá čísla** a **reálná čísla**. Důležitým faktorem je také velikost každého datového typu.

Pro zjednodušení si můžete paměť programu představit jako přidělenou zásuvku s mnoha přihrádkami. Každá přihrádka má velikost jeden bit (bit je základní a nejmenší počítačová jednotka) a je potřeba vědět, kolik přihrádek daný typ zabere.

Byte

Nejmenší typ, do kterého je možné ukládat pouze **celá čísla**. V paměti zabírá velikost **8 bitů** neboli právě jeden bajt (anglicky byte). Pokud již máte nějaké povědomí o základní reprezentaci čísel v počítači, jistě tušíte, že do tohoto typu lze uložit pouze ta čísla, která je možné zapsat pomocí 8 bitů. V praxi to znamená čísla v rozsahu -128 až +127. Označuje se celým názvem `byte`.

Short

Druhý nejmenší číselný typ, do kterého lze opět ukládat pouze **celá čísla**. Zabírá v paměti **16 bitů** a moc často se s ním zřejmě nesetkáte. Lze do něj uložit čísla v rozsahu -32 768 až +32 767. Stejně jako `byte` se označuje celým svým názvem `short`.

Integer

Číselný typ, se kterým se setkáte zřejmě nejčastěji, je **integer**. Stále slouží pouze k ukládání **celých čísel**, ale díky velikosti **32 bitů** již jeho rozsah stačí na velké množství potřebných výpočtů. Můžete do něj ukládat čísla v rozsahu –2 147 438 648 až +2 147 438 647. Na rozdíl od výše uvedených typů se neoznačuje celým názvem, ale zkratkou `int`.

Long

Jedná se o poslední a největší typ pro ukládání **celých čísel**. Jeho velikost je **64 bitů**, což v naprosté většině případů postačí i pro vaše nejsložitější výpočty. Má rozsah –9 223 372 036 854 775 808 až +9 223 372 036 854 807 a znovu se označuje celým svým názvem `long`. Všechna ta čísla si samozřejmě není nutné přesně pamatovat, u všech uvedených rozsahů je důležité jen přibližně řádově tušit, jak velké číslo se do nich ještě vejde. Pokud budete potřebovat, najdete přesné hodnoty také v níže uvedené tabulce.

Tabulka 2.1 Celočíselné datové typy

Označení typu	Velikost	Nejmenší možné číslo	Největší možné číslo
`byte`	8 bitů	-128	+127
`short`	16 bitů	-32 768	+32 767
`int`	32 bitů	-2 147 438 648	+2 147 438 648
`long`	64 bitů	-9 223 372 036 854 775 808	+ 9 223 372 036 854 775 807

Důležité: Možná vás už napadla otázka, co se stane, pokud se číslo svým rozsahem do zvoleného typu prostě nevejde. V tom případě dojde k takzvanému přetečení a výsledek operace nebude na první pohled vůbec dávat smysl. Například při sečtení dvou příliš velkých kladných čísel můžete dostat jako výsledek číslo záporné. Je potřeba si na podobné věci dávat pozor a volit datové typy pečlivě a opatrně.

Float

Nyní jsme se dostali k prvnímu typu pro reprezentaci **reálných čísel** neboli čísel s desetinou částí. Java ukládá reálná čísla podle mezinárodního standardu IEEE 754, to znamená, že se zobrazují stejně jako v jiných jazycích používajících tento standard. Typ **float** ukládá reálná čísla o velikosti **32 bitů** a používá se, pokud potřebujete ušetřit místo. Není nutné přesně vědět jeho rozsah, maximální uložená hodnota může být přibližně $3{,}4x10^{38}$. Označuje se celým svým názvem `float`. Reálné typy se v programech zapisují s desetinnou tečkou.

Double

Obvykle základní používaný typ pro ukládání **reálných čísel** je typ **double**. Má takzvanou dvojitou přesnost a zabírá v paměti prostor **64 bitů**. Rozhodně by se vám nemělo podařit přesáhnout rozsah tohoto typu. Opět se označuje celým svým názvem `double`. Stejně jako `float` by tento typ nikdy neměl být používán pro ukládání hodnot, u kterých je nutná absolutní přesnost, jako jsou například peníze. U běžných výpočtů (téměř všech) ale ztrátu přesnosti vůbec nepoznáte.

Poznámka: Ačkoli vás možná po upozornění na možnost přetečení napadlo, že stačí všechna čísla reprezentovat největšími typy, není to vhodné řešení. Pokud se celé číslo, které potřebujete použít, nevejde ani do rozsahu typu `long`, je sice na místě nahradit ho typem `double`, to je ale krajní řešení. Je dobrým zvykem používat nejmenší možný typ kvůli úspoře místa a v případě reálných typů i kvůli jinému způsobu jejich ukládání. Celá čísla jsou z principu ukládána přesně, reálná čísla nejdou vždy precizně vyjádřit a může dojít ke ztrátě jejich přesnosti.

Logický datový typ

Tento typ představuje jeden bit informace, který může nabývat pouze dvou hodnot. Jeho velikost ale není přesně specifikována. Zastupuje logické konstanty používané ve výrazech: `true` – logická jednička neboli pravda a `false` – logická nula neboli nepravda. Používá se zejména v podmínkách a jiných testovacích výrazech. Označuje se anglickým výrazem `boolean` a lze mu přiřadit výše zmíněné dvě hodnoty, `true` a `false`.

Znakový datový typ

Znak je jediný typ zastupující text mezi primitivními datovými typy. Označuje se anglickou zkratkou `char` (ze slova character) a má velikost **16 bitů**. Představuje právě jeden

znak v kódování UNICODE, které Java vnitřně používá, jelikož u něj nedochází k problémům při používání národních znaků. Hodnoty tohoto typu musí být vždy uzavřeny do apostrofů a je více možností jejich zápisu:

- Právě jedním znakem zapsatelným na klávesnici, například: `'A'`, `'ž'`, `'8'`, `'%'`.
- Specifickým číselným kódem ve formátu `\uXXXX`. Tabulku těchto čísel naleznete snadno na internetu například po zadání „*unicode table*“ do vašeho vyhledávače. Tento způsob se většinou používá pro znaky, které nelze snadno zadat na klávesnici, například: `'\u0024'` je znak `$` a `'\u0126'` je znak `Ħ`.
- Použitím sekvence se zvláštním významem pro netisknutelné znaky nebo pro znaky, které mají v Javě zvláštní význam. Využijete je později v části o textových řetězcích. Nejčastěji používané sekvence najdete v následující tabulce.

Tabulka 2.2 Znakové sekvence se zvláštním významem

Sekvence	Význam
`'\n'`	Nový řádek – ve výpisu textu dojde na tomto místě k jeho zalomení, nezobrazuje se.
`'\t'`	Tabulátor – do textu se vloží znak tabulátoru, nezobrazuje se.
`'\b'`	Backspace – smaže ve výpisu předchozí znak, nezobrazuje se.
`'\r'`	Návrat na začátek řádky, nezobrazuje se.
`'\\'`	Zpětné lomítko, jako speciální znak ruší funkci ostatních speciálních znaků za nimi, které je potom možné vypsat jako text.
`'\''`	Apostrof, bez zpětného lomítka by ukončil sekvenci znaku a ten by se jevil jako prázdný.
`'\"'`	Uvozovky, bez zpětného lomítka by začínaly nebo ukončovaly textový řetězec.

Poznámka: Je možné, že se vám bude funkce sekvencí `'\n'` a `'\r'` jevit jako stejná. Rozdíl je daný použitým operačním systémem, každý operační systém používá jako znak konce řádku jinou sekvenci. Původně rozdíl vznikl podle zvyku z psacích strojů, kdy nový řádek pouze posunul papír nahoru, ale nepřesunul se na začátek, zatímco návrat na začátek řádku zase posunul psací hlavu doleva, ale neumožňoval přechod na nový řádek.

Proměnné

Teď už víte, jaká data je možné ukládat. Stále ale ještě nevíte kam. K ukládání dat slouží právě **proměnné**. Představují adresu místa, kde jsou v paměti uložená určitá data. Podle datového typu, který dané proměnné určíte, Java vyhradí potřebné místo v paměti a zapamatuje si jeho adresu. Protože není v lidských silách zapamatovat si přesnou adresu tak, jak ji používá počítač, mohou být proměnné téměř libovolně pojmenované. Jménu, které proměnné přidělíte, se říká **identifikátor** a v programu místo přesné adresy pracujete s ním.

Deklarace proměnné

Aby bylo možné proměnnou v programu použít, je nutné ji nejprve deklarovat neboli upozornit Javu na její přítomnost. Díky deklaraci proměnné Java vyhradí a pojmenuje místo v paměti, které pak můžete dále používat. Při deklaraci proměnné je tedy nutné mít vybrán její datový typ a název. Nejdříve zapíšete označení datového typu, za mezeru zapíšete identifikátor proměnné neboli vámi zvolené jméno a celý příkaz již bez mezer ukončíte středníkem.

Například zápis celočíselné proměnné pojmenované `cislo` by tedy vypadal takto:

```
int cislo;
```

Z jakých datových typů máte na výběr, bylo řečeno v předchozí části kapitoly. Jména proměnných si můžete zvolit téměř libovolně, i zde ovšem existují jistá omezení:

- jméno **nesmí začínat číslem**, ale jinde v názvu se čísla vyskytovat smí,
- jméno **nesmí obsahovat mezery**,
- jméno **nesmí tvořit hodnoty** `true`, `false` a `null`,
- jméno **musí být unikátní ve svém rozsahu platnosti** (bude vysvětleno dále v této kapitole),
- jménem **nesmí být některé z rezervovaných klíčových slov jazyka Java**, ačkoli obsaženo být může. Seznam rezervovaných klíčových slov najdete v tabulce níže.

Tabulka 2.3 Rezervovaná klíčová slova jazyka Java

`abstract`	`continue`	`for`	`new`	`switch`
`assert`	`default`	`goto`	`package`	`synchronized`
`boolean`	`do`	`if`	`private`	`this`
`break`	`double`	`implements`	`protected`	`throw`
`byte`	`else`	`import`	`public`	`throws`
`case`	`enum`	`instanceof`	`return`	`transient`
`catch`	`extends`	`int`	`short`	`try`
`char`	`final`	`interface`	`static`	`void`
`class`	`finally`	`long`	`strictfp`	`volatile`
`const`	`float`	`native`	`super`	`while`

Poznámka: Programátoři v Javě se poměrně důsledně drží konvencí v pojmenovávání proměnných, které byste měli dodržovat i vy. Alespoň pokud chcete, aby byly vaše programy pro ostatní srozumitelné. Názvy proměnných podle konvence vždy začínají malým písmenem, co nejlépe vystihují účel proměnné, a pokud obsahují více slov, každé další slovo začíná velkým písmenem. Příkladem takového názvu je třeba `vypoctenyObsahSteny`.