Informace o charakteru, průběhu a hodnocení předmětu. Úvod do jazyka Java a prostředí BlueJ

August 28, 2014

1 Informace o průběhu, hodnocení, apod.

1.1 O předmětu (1)

Prakticky zaměřený bakalářský předmětCílem je naučit základním principům objektového programování a algoritmizaceNavazuje na

- IB001 – **Úvod do programování** předpokládají se znalosti na úrovni IB001 nebo IB111, je v zásadě jedno, zda v C nebo Pythonu. Absolventům IB001 bude Java bližší syntaxí k C. Absolventi větve Python porozumí o něco snadněji objektovému přístupu v Javě, jelikož mají částečné základy objektového přístupu.
- IB002 – **Algoritmy a datové struktury I**

1.2 Předpoklady

Předpokládají se základní znalosti strukturované algoritmizace a programování (v rozsahu Úvodu do programování), tj. např.:

- základní příkazy, sestavování jednoduchých výrazů;
- základní datové typy (celá a reálná čísla, logické proměnné, řetězce);
- základní řídicí struktury – větvení, cykly, procedury/funkce.

1.3 Návaznosti – co dál studovat

Na tento základní kurz PB162 navazují na FI

- PV168 – **Seminář z jazyka Java** (jarní semestr), náplní je zvládnutí Javy umožňující vývoj jednodušších praktických aplikací s GUI, databázemi, základy webových aplikací. V průběhu semestru se pracuje na uceleném projektu formou párového programování plus některých individuálních úloh.Učí kolektiv zkušených cvičících pod vedením Tomáše Pitnera, Luďka Bártka, Petra Adámka a Martina Kuby.
- PB138 – **Moderní značkovací jazyky** (jarní semestr), náplní jsou XML a související technologie, s prvky týmového vývoje (projekty, využití služeb hostování projektů jako jsou *Google Code* a *GitHub*)Učí kolektiv zkušených cvičících pod vedením Luďka Bártka a Tomáše Pitnera.

1.4 Návaznosti pokročilých předmětů

Na Seminář z Javy navazují na FI

- PA165 – **Vývoj aplikací v jazyce Java** (podzimní semestr) - pokročilejší předmět spíše magisterského určení, předpokládá znalosti/zkušenosti z oblasti databází, částečně sítí a distribuovaných systémů - a také Javy zhruba v rozsahu PB162 a PV168. Náplní je zvládnutí netriviálních, převážně klient/server aplikací na platformě JavaEE.
- Přednáší Petr Adámek, Tomáš Pitner, Martin Kuba, Filip Nguyen, Dan Tovarňák a externí přednášející z firem.

1.5 Návaznosti – co dál studovat

Problematice webových a mobilních aplikací se na FI věnují např.

- každý semestr **PV226 Seminář Lasaris**;
- v jarním semestru **PV219 Seminář webdesignu**;
- v podzimním semestru předmět zaměřený na vývoj v **Ruby**;
- v jarním semestru **PV239 Mobilní platformy** a
- v podzimním návazný **PV256 Projekt z programování pro Android**;

1.6 Hodnocení předmětu

Naleznete jako tzv. osnovu v IS (`https://is.muni.cz/auth/el/1433/podzim2014/PB162/op/hodnoceni.qwarp`).

1.7 Předpokládaný harmonogram výuky

- Harmonogram přednášek i cvičení (`https://is.muni.cz/auth/el/1433/podzim2014/PB162/op/index.qwarp`)

1.8 O přednášejících (1)

Tomáš Pitner

- pracovna **A303** (budova A1) laboratoře Lasaris (`http://lasaris.fi.muni.cz`), příp. kanc. **správy Vědecko-technologického parku CERIT** (1. NP/přízemí budovy A2);
- tel. 54949 5940 (z tlf. mimo budovu), kl. 5940 (volání v rámci fakulty i celé MU)
- e-mail: tomp@fi.muni.cz (`mailto:tomp@fi.muni.cz`)
- Web: `http://www.fi.muni.cz/~tomp`materiály PB162-Java (IS): https://is.muni.cz/auth/el/1433/poc (`https://is.muni.cz/auth/el/1433/podzim2014/PB162/`)

1.9 O přednášejících (2)

Radek Ošlejšek

- pracovna **A305** (budova A1) laboratoře Lasaris (`http://lasaris.fi.muni.cz`)
- tel. 54949 6121 (z tlf. mimo budovu), kl. 6121 (volání v rámci fakulty i celé MU)
- e-mail: oslejsek@fi.muni.cz (`mailto:oslejsek@fi.muni.cz`)
- www RO: `http://www.fi.muni.cz/~oslejsek`

1.10 Konzultační hodiny

Primárním konzultačním bodem jsou vaši cvičící. Cvičení jsou vedena mj. právě z důvodu možnosti konzultací.Konzultace přímo s přednášejícími**Tomáš Pitner**: vždy v kanc. A303

- **Út 10.00 – 11.30**
- nebo jindy, dle dohody

1.11 Informační zdroje (online)

- Studijní materiály předmětu (`https://is.muni.cz/auth/el/1433/podzim2014/PB162/`) jsou sice vystaveny předem, ale mohou být aktualizovány i průběžně, popř. i zpětně (chyby, překlepy...)
- Materiály loňské (`https://is.muni.cz/auth/el/1433/podzim2013/PB162/`)

1.12 Informační zdroje (knihy)

- Rudolf Pecinovský: Myslíme objektově v jazyku Java (`http://knihy.pecinovsky.cz/mojj/`) nebo Java 7 – Učebnice objektové architektury pro začátečníky (`http://knihy.pecinovsky.cz/uoa1\_j7/index.html`) Grada Publishing
- Rudolf Pecinovský: Java 5.0 – Novinky jazyka a upgrade aplikací (`http://i.iinfo.cz/files/root/k/java-5-0-novinky-jazyka-a-upgrade-aplikaci.pdf`) (fulltext v PDF zdarma)
- Tomáš Pitner: **Java – začínáme programovat**, Grada Publishing (`http://www.gradapublishing.cz`), 2002, doprovodný web knihy (`http://www.fi.muni.cz/~tomp/java/ucebnice`) (učebnice je orientovaná na Javu 1.4 a nižší; většina poznatků je platných i nadále, ale v Javě 5, 6, 7 a 8 se objevila řada nových prvků)
- Pavel Herout: **Učebnice jazyka Java**, Kopp (`http://www.kopp.cz`), 2000-2010, doprovodný web knihy (`http://www.kiv.zcu.cz/~herout/java/ujj1/`)
- příp. i Pavel Herout: Java – grafické uživatelské rozhraní a čeština, Kopp (`http://www.kopp.cz`), 2001 – pro pokročilé

1.13 Informační zdroje (knihy)

- Bruce Eckel: **Myslíme v jazyce Java – příručka programátora**, Grada Publishing (`http://www.gradapublishing.cz`), 2000
- příp. Bruce Eckel: Myslíme v jazyce Java – příručka zkušeného programátora, Grada Publishing (`http://www.gradapublishing.cz`), 2000 – pro pokročilé
- Joshua Bloch: **Java efektivně – 57 zásad softwarového experta**, Grada Publishing (`http://www.gradapublishing.cz`)
- Bogdan Kiszka: **1001 tipů a triků pro programování v jazyce Java**, Computer Press, 2003
- Bruce Eckel: Thinking in Java Stáhnout zdarma (PDF) (`http://www.mindview.net/Books/TIJ/DownloadSites`)

2 Jazyk a prostředí Java

2.1 Java jako programovací jazyk

- Je jazykem "3. generace" (3GL), imperativním jazykem vysoké úrovně
- Je jazykem *univerzálním*; není určen výhradně pro specifickou aplikační oblast.
- Je jazykem *objektově-orientovaným*, výpočet je realizován jako volání metod objektů (zasílání zpráv objektům).
- Ideovým předchůdcem Javy je C++ (a evt. Delphi) (C++ zbaveno zbytečností a nepříjemností).
- Je jazyk *jednodušší* než C++, komplikovaností srovnatelný s jazykem C#, jenž se Javou inspiroval.

2.2 Java v budoucnu

- Pro tradiční typy serverových podnikových aplikací (IS) zůstává Java (Enterprise Edition) klíčovou platformou spolu s .NET.
- Perspektivním směrem vývoje je zachování Java platformy (JVM, stávající knihovny, aplikace, aplikační prostředí).
- Rychle se vyvíjejí skriptovací jazyky na této platformě (Groovy, JRuby, Jython...).
- Objevují se nové jazyky na javové platformě (Scala).

2.3 Proč Java

- Java je jazyk pro vývoj a běh jednoduchých i rozsáhlých aplikací.
- Vývoj je efektivnější než na jejich předchůdcích (C++) a výsledné aplikace "běží všude".
- Silnou typovaností, běhovou bezpečnostní kontrolou, stabilními knihovnami vč. open-source a rozsáhlým souborem dobrých praktik nabízí aplikacím velmi vysokou robustnost.
- Nezavádí zbytečnosti a vede ke správným a dále uplatnitelným návykům.
- Je velmi perspektivní platformou pro vývoj open-source i komerčního SW, mj. pro *extrémně* velké množství volně dostupných knihoven.

2.4 Z toho plyne, že...

- co se naučíme v Javě, v C# jako když najdeme.
- .NET/CLI jako platforma je silně ovlivněná Java EE.
- Rovněž moderní skriptovací jazyky (Groovy zcela, Ruby a Scala se inspirovaly) koncepčně i syntakticky vychází z Javy.

2.5 Java jako správný vzor

Java podporuje vytváření správných návyků v objektovém programování

- ... a naopak systematicky brání *přenosu některých špatných návyků* z jiných jazyků
- začít v imperativním paradigmatu ihned se *slabě typovaným* (např. skriptovacím) jazykem není vhodné,
- Je lépe nejdříve dobře zvládnout klasický (*silně typovaný*) jazyk, navyknout si na omezení a kontroly, vytvořit si styl, a pak snadno přejít.

2.6 Další charakteristiky

- Program v Javě je *meziplatformně přenositelný* na úrovni *zdrojového i přeloženého kódu.*
- Je to umožněno tím, že přeložený javový program běží v tzv. *Java Virtual Machine (`http://java.sun.com/docs/books/vmspec/`)* (JVM).
- Zdrojový i přeložený kód je tedy přenositelný mezi všemi obvyklými platformami (UNIX, Windows, Mac OS X, ale také sálové počítače, minipočítače typu IBM AS/400 apod.).
- Při respektování omezení mobilních zařízení (smartphones) je Java použitelná i tam (viz mnohé mobilní hry), byť dnes v podstatě úplně vytlačená Androidem.
- Tedy všude tam, kde *existuje příslušná JVM.*

2.7 Java jako běhové prostředí

Kód je při běhu *dobře zabezpečen*:

- Je možné velmi jemně nastavit úrovně přístupu k hostitelskému systému pomocí tzv. Security Manageru (`http://www.securingjava.com/chapter-two/chapter-two-8.html`) (knihu možné pročíst i z webu!).
- Je možné ověřovat před spuštěním *elektronický podpis* kódu.

2.8 Java pro programátora (produktivita)

- jazyk vhodný pro *efektivní (rychlé) psaní přehledných programů* (mj. také díky *dokumentačním* možnostem);
- v průměru *vyšší produktivita* programátorské práce v Javě než v C++;
- nesčetné množství zdarma dostupných knihoven pro různorodé aplikační oblasti, např. na SourceForge (`http://java.foundries.sourceforge.net/`) a tisících dalších místech;
- k dispozici je řada kvalitních vývojových prostředí (i zdarma) - NetBeans (`http://www.netbeans.org`), JBuilder (`http://www.borland.com`), Eclipse (`http://eclipse.org`), IDEA (`http://www.intellij.com/`), BlueJ - výukové vývojové prostředí - používáme v tomto předmětu! (`http://bluej.org`)

2.9 Java pro programátora (výhody oproti C++)

Konkrétní možnosti:

- V Javě se dobře píší *vícevláknové aplikace* (multithreaded applications).
- Java má automatické *odklizení nepoužitelných objektů* (automatic garbage collection).
- Java je jednodušší než C++ (méně syntaktických konstrukcí, méně nejednoznačností v návrhu).

2.10 Hlavní domény Javy (1)

- Škálovatelné výkonné *aplikace běžící na serverech* – Java Enterprise Edition (`http://www.oracle.com/technetwork/java/javaee/overview/index.html`)
- Přenositelné *desktopové* i *grafické/okénkové (GUI)* aplikace
- *Výukové účely* (nahradila Pascal jako referenční jazyk)
- Aplikace na *přenosných a vestavěných zařízeních* – Java Micro Edition (`http://www.oracle.com/technetwork/java/embedded/javame/index.html`)
- Aplikace na *chytrých kartách* (smart cards) –Java Card (`http://www.oracle.com/technetwork/java/embedded/javacard/overview/index.html`) technology

2.11 Hlavní domény Javy (2)

- Webové aplikace: jak back-end (Java EE), tak front-end (JavaFX)
- Velké podnikové (informační) systémy (Java EE)
- Integrační technologie (propojování systémů) (Java EE)

2.12 Javová platforma

Javovou platformu tvoří:

- **Java Virtual Machine**
- **Java Core API** (základní knihovna tříd)
- **překladač** (přístupný např. příkazem javac) a další vývojové nástroje

2.13 Java je tedy dána...

- **definicí jazyka** (The Java Language Specification, Java SE 8 Edition (`http://docs.oracle.com/javase/specs/index.html`)) – syntaxe a sémantika jazyka
- **popisem chování JVM** (The Java Virtual Machine Specification, Java SE 8 Edition (`http://docs.oracle.com/javase/specs/index.html`))
- **popisem Java Core API** (Java™ Platform, Standard Edition 8API Specification (`http://docs.oracle.com/javase/8/docs/api/index.html`))

2.14 Vývoj Javy

- *Nejrychleji* se vyvíjí Java Core API.
- Chování JVM se mění např. pokud se objeví bezpečnostní "díra" nebo nelze-li dosáhnout požadované změny chování pomocí modifikace Java Core API.
- Daleko konzervativnější je samotný *jazyk - mění se zřídka*, ale přece: např. Java2, v1.4 přidala nové klíčové slovo *assert*, Java 5.0 (postaru 1.5) obohacuje jazyk o *enum*, generické typy, anotace a další rysy. Java 7 přináší další spíše drobné změny jazyka, např. v ošetřování výjimek, uvolňování systémových zdrojů, ale také zásadnější prvky (lambda-výrazy).

2.15 Specifikace a implementace Javy

- **Specifikace** Javy (tzv. "Editions") - např.: *Java 2 Standard Edition, 1.4* nebo *Java Standard Edition 6*
- **Implementace** Javy ("Development Kits" nebo "Runtime Environments"), např.: *Java Software Development Kit,* 6.0 – obsahuje vývojové nástroje.
- *Java Runtime Enviroment, 6.0* – obsahuje jen **běhové prostředí** pro spouštění hotových přeložených programů.

2.16 Verze Javy - starší konvence

- Verze Javy byly až do verze 5.0 označovány jako "Java 2, vX.Y, tedy následovně:
  - tzv. major číslo, např. Java 2, v**1.4**
  - tzv. minor číslo, např. Java 2, v1.4.**2**
- změnu minor (třetího) čísla doprovází jen odstraňování chyb
- při změně major (druhého) čísla se může měnit Core API a někdy i jazyk
- ke změně prvního čísla dochází až s verzí 5 (postaru 1.5) - tj. s celkovou změnou pojmenovávacího schématu

2.17 Aktuální verze

Stav k září 2014:

- *Java Standard Edition 8* (pokračují i SE 6 a 7) a
- *Java Enterprise Edition 7* jsou stabilní verzí pro všechny platformy
- Aktuální informace najdete vždy na webu Oracle `http://www.oracle.com/technetwork/java/index.html`.

2.18 Licence k použití (a redistribuci) Javy

- používání Javy pro běžný vývoj (i komerční) je zdarma, licenční ujednání Java SE for Business License (`http://www.oracle.com/technetwork/java/javasebusiness/downloads/jfb-license-461063.html`)
- distribuce vyvíjí Oracle i další výrobci (např. IBM) a tvůrci open source – jako OpenJDK (`http://openjdk.java.net/`)

2.19 Stažení distribuce Oracle

- `http://www.oracle.com/technetwork/java/index.html` pro Windows, Solaris, Linux, Mac OS X
- Lze stáhnout jak samotné vývojové prostředí (JDK), jen běhové prostředí (JRE) nebo JDK v balíčku s IDE (Integrated Development Environment) NetBeans (`http://netbeans.org`) (nyní NetBeans 8.0).
- Dokumentace se stahuje z téhož místa, ale *samostatně*. Obvyklejší je číst z webu – *pozor na verzi Javy, jejíž dokumentaci čteme*!

2.20 Typy distribucí Javy (Oracle)

Lze stáhnout:

- samotné vývojové prostředí (JDK), např. Java SE 7 JDK (`http://www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html`)

- jen běhové prostředí (JRE), např. Java SE 7 JRE
- JDK v balíčku s grafickým (okénkovým) integrovaným vývojovým prostředím (IDE, Integrated Development Environment) NetBeans (`http://netbeans.org`).

2.21 Obsah vývojové distribuce Javy

- **Vývojové nástroje (Development Tools)** v `bin` – určené k vývoji, spouštění, ladění a dokumentování programů v Javě.
- **Běhové prostředí Javy (Java Runtime Environment)** se nalézá v `jre`. Obsahuje Java Virtual Machine (JVM), knihovnu tříd Java Core API a další soubory potřebné pro běh programů v Javě.
- **Přídavné knihovny (Additional libraries)** v podadresáři `lib` jsou další knihovny nutné pro běh vývojových nástrojů.
- **Ukázkové applety a aplikace (Demo Applets and Applications)** v `demo`. Příklady zahrnují i zdrojový kód.

2.22 Nástroje ve vývojové distribuci

Pod Windows jsou to `.exe` soubory umístěné v podadresáři `bin`

- `java` **(nebo** `jexec`**)** - spouštěč (přeloženého bajtkódu)
- `javac` - překladač (`.java` → `.class`)
- `javadoc` - generátor dokumentace API
- `jar` – správce archivů JAR (sbalení, rozbalení, výpis) a `jarsigner` – podpisovač archivů JAR
- `jdb` – debugger a `jcmd` - zasílá diagnostické příkazy JVM
- `appletviewer` – referenční prostředí pro spouštění appletů
- `javaws` – referenční prostředí pro spouštění aplikací typu "Java Web Start" prostřednictvím Java Network Launching Protocol (JNLP) a `javafxpackager` – nástroj na sbalení JavaFX aplikace

2.23 Pomocné nástroje ve vývojové distribuci

- `javah` – generátor hlavičkových souborů pro C - používá se při programování tzv. nativních (platformově závislých) metod dostupných přes *Java Native Interface* (JNI)
- `javap` – disassembler bajtkódu (např. pro ruční optimalizace, hledání chyb)

2.24 Budoucnost Javy

- Na závěr optimistického úvodu si pročtěte zajímavý článek analytika od Forrester: Java Is A Dead-End For Enterprise App Development (`http://blogs.forrester.com/mike\_gualtieri/10-11-23-java\_is\_a\_dead\_end\_for\_enterprise\_app\_development`)

3 První kroky javového programování v prostředí BlueJ

3.1 Struktura javového programu

- Každý netriviální javový program sestává z *více tříd* (class)
- Třídy jsou členěny do *balíků* (packages)
- U běžné "desktopové" aplikace představuje vždy jedna (evt. více) tříd *vstupní bod* do programu – je to třída/y obsahující metodu main.

3.2 Spuštění a běh javového programu

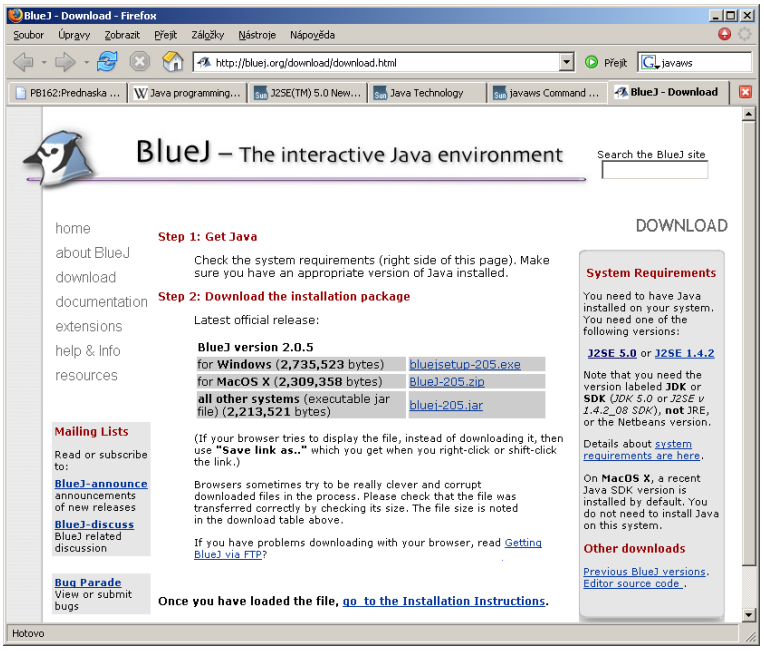
1. Spuštění programu se zahajuje vstupem do metody main.
2. Běh programu spočívá ve vytváření objektů a interakci s nimi, tj. volání jejich metod.
3. Běh končí, jakmile jsou provedeny všechny příkazy aktivované metodou main.

3.3 Vývojové prostředí pro přednášky a cvičení

- Pro demostraci na přednáškách v kurzu PB162 budeme převážně používat jednoduché, pro výuku určené, vývojové prostředí BlueJ.
- Pro výuku ve cvičeních kurzu PB162 budeme již používat profesionální vývojové prostředí NetBeans.

3.4 Prostředí BlueJ

- BlueJ je pro všechny platformy (Win, Linux, Mac OS X...) zdarma dostupné na `http://bluej.org`.
- BlueJ je lokalizované pro češtinu vč. základního manuálu.
- Pro BlueJ existuje i jednoduchý animátor běhu programů - vizualizuje spouštění a chod programů.
- Toto prostředí je již předinstalované v síti FI.



3.5 Stažení BlueJ

- BlueJ je již v síti FI instalováno, pro vlastní potřebu jej lze zdarma stáhnout z `http://bluej.org`.
- Doporučujeme aktuální verzi – v současnosti (září 2014) BlueJ 3.1.1
- Předtím je třeba mít korektně nainstalováno samotné vývojové prostředí Java.

3.6 Spuštění BlueJ

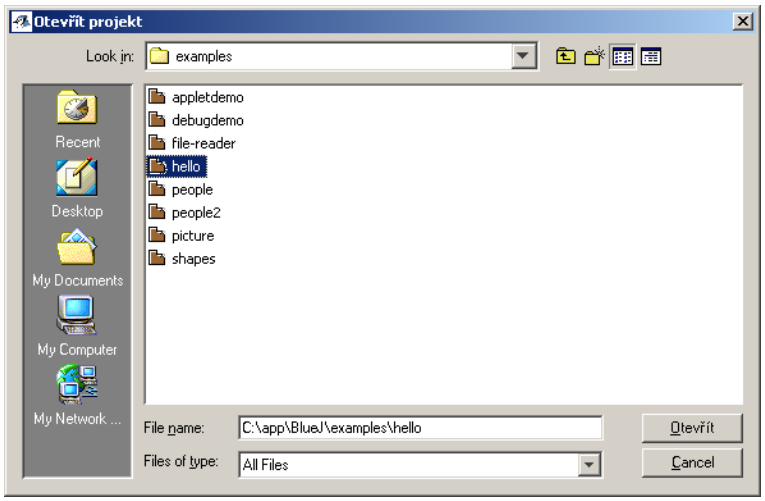
Po úspěšné instalaci na:

Windows Najděte v menu Start systému Windows složku *BlueJ* a zde spouštěcí odkaz *BlueJ* s ikonou ptáka.

Linux pomocí `bluej` (odkaz je v `/usr/bin`) – tím lze prostředí spustit.

3.7 Otevření hotového programu (projektu) v BlueJ

- Každý javový program vytvořený v BlueJ představuje jeden *projekt*. Toto je specificky vlastnost BlueJ, ne Javy jako takové.
- Práce s hotovým programem v BlueJ tedy znamená *otevřít projekt* → menu Projekt/Otevřít projekt.

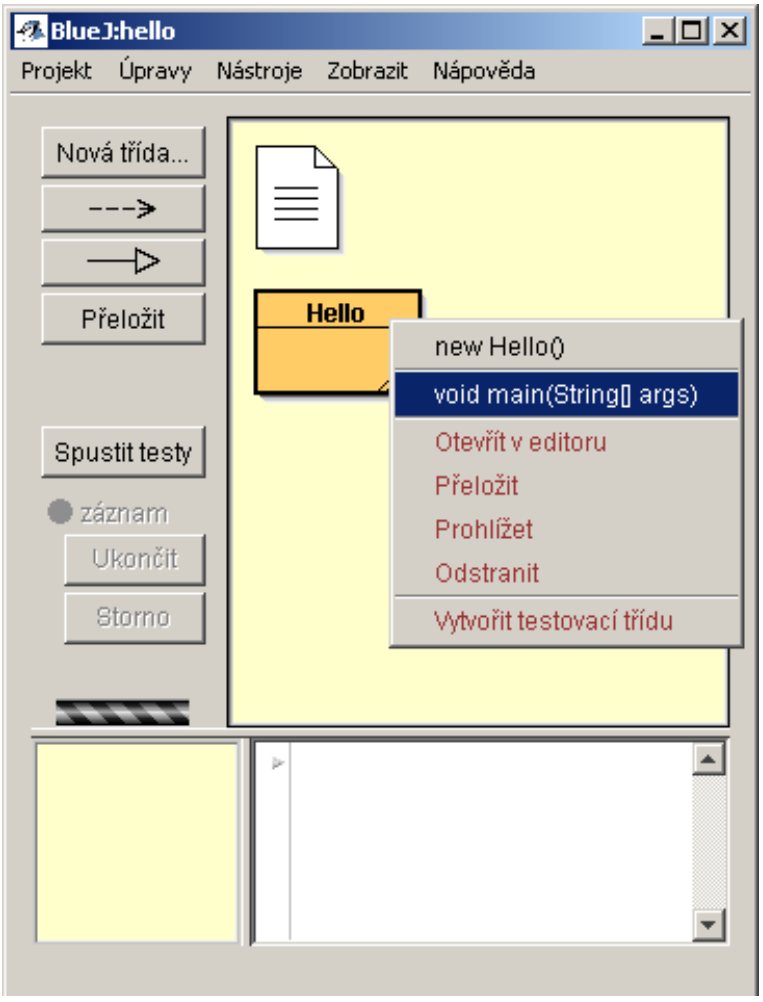


3.8 Spuštění hotového programu

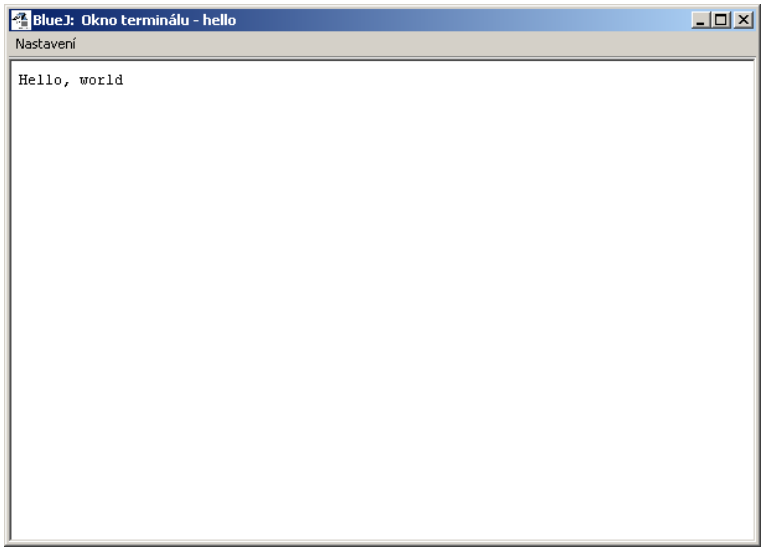
- Každý běžný javový program spouštíme (nejen v BlueJ) aktivací metody main.
- V BlueJ to znamená kliknout pravým tlačítkem myši na ikoně příslušné třídy a vybrat *void main(String[] args)*.
- Při spuštění uvádíme parametry, které metoda main dostane. Jelikož zatím žádné specifikovat nepotřebujeme, ponecháme dialog, jak je a stiskneme Enter (klikneme OK).
- Výsledek spuštění programu se projeví v tzv. okně terminálu (konzole) BlueJ a vypadá takto:

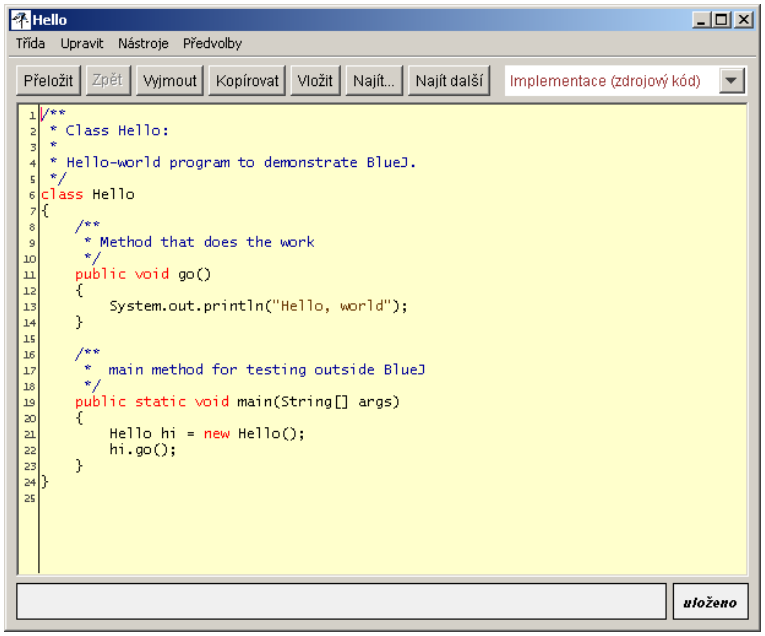
3.9 Interaktivní vytvoření objektu (instance)

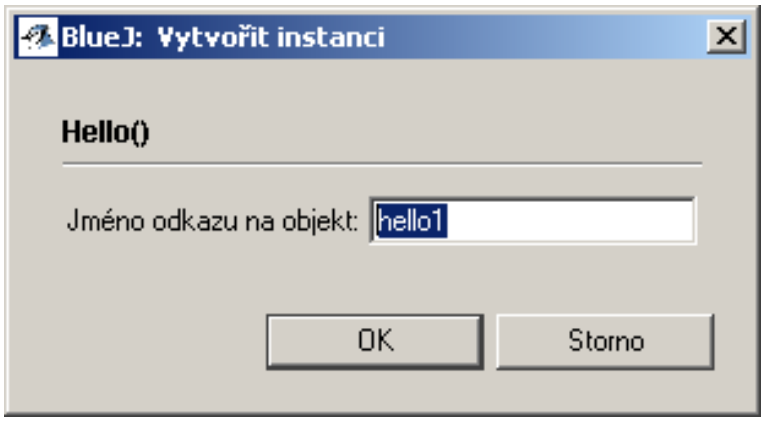
- Objektový přístup znamená především to, že program za běhu vytváří objekty a volá jejich metody. To lze v BlueJ snadno realizovat.
- V demostračním programu Hello máme jednu třídu (Hello), od níž lze vytvořit objekt (instanci) a na ní volat metody.
- Zdrojový kód třídy Hello (dostupný poklepáním na její ikonu) prozradí, které metody zde máme k dispozici.
- Dále postupujeme:
  1. Pravým tlačítkem klikneme na ikonu Hello a vybereme *new Hello()*.
  2. do dialogu uvedeme název odkazu na vytvářený objekt typu Hello - můžeme ponechat *hello1*.
  3. Vytvořený objekt (instance třídy Hello) je k dispozici jako červená krabička vlevo dole.


BlueJ:hello
Projekt
Úpravy
Nástroje
Zobrazit
Nápověda
Nová třída...
Přeložit
Spustit testy
záznam
Ukončit
Storno
Hello
new Hello()
void main(String[] args)
Otevřít v editoru
Přeložit
Prohlížet
Odstranit
Vytvořit testovací třídu


BlueJ: Volání metody
// main method for testing outside BlueJ
void main(String[] args)
Hello.main ({})
OK
Storno


BlueJ: Okno terminálu - hello
Nastavení
Hello, world


Hello
Třída Upravit Nástroje Předvolby
Přeložit Zpět Vyjmout Kopírovat Vložit Najít... Najít další
Implementace (zdrojový kód)
/**
* Class Hello:
*
* Hello-world program to demonstrate BlueJ.
*/
class Hello
{
/**
* Method that does the work
*/
public void go()
{
System.out.println("Hello, world");
}
/**
* main method for testing outside BlueJ
*/
public static void main(String[] args)
{
Hello hi = new Hello();
hi.go();
}
}
uloženo



3.10 Práce s objektem (instancí)

- Vytvořený objekt (instance *hello1* třídy *Hello*) je k dispozici jako červená krabička vlevo dole.
- S instancí je nyní možné komunikovat/interagovat - tj. volat metody.
- Jelikož v úvahu (viz zdrojový kód Hello) připadá jen metoda *void go()*, zavoláme právě ji:
- Metoda proběhne s výsledkem podobným, jako když jsme program spouštěli metodou main. Metoda main totiž realizovala postupně stejné kroky, jako jsme teď provedli interaktivně (krok po kroku) sami.

3.11 Úprava hotového programu v BlueJ

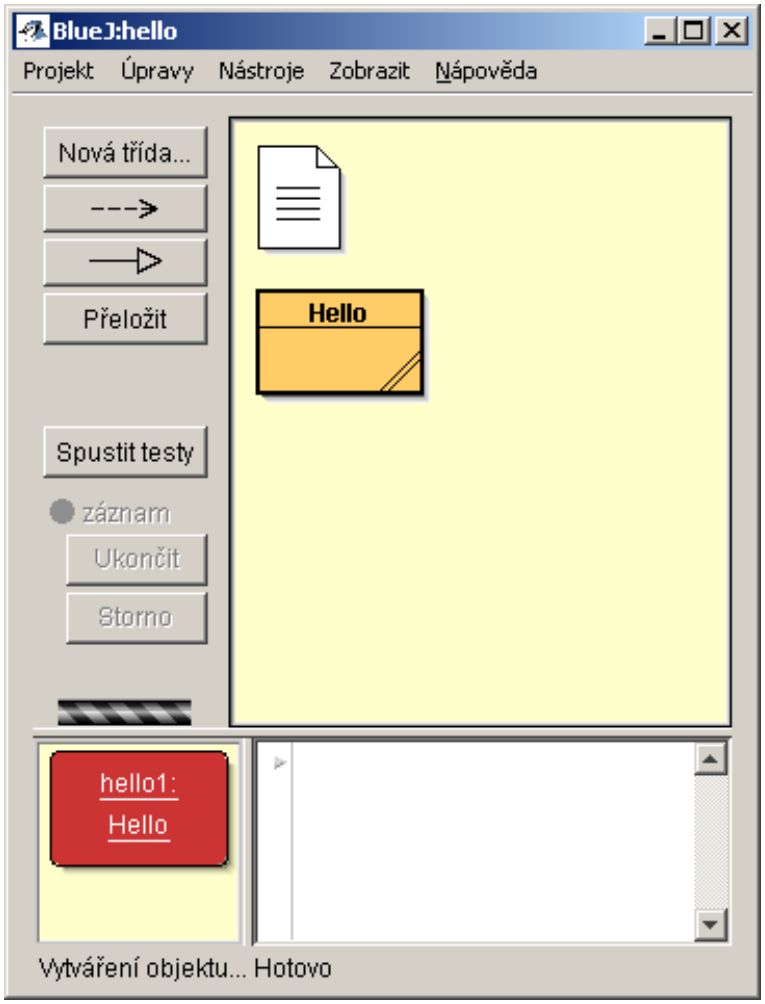
Nejjednodušší úpravou programu je změna těla některé z metod, v našem případě již známé metody *go()*.

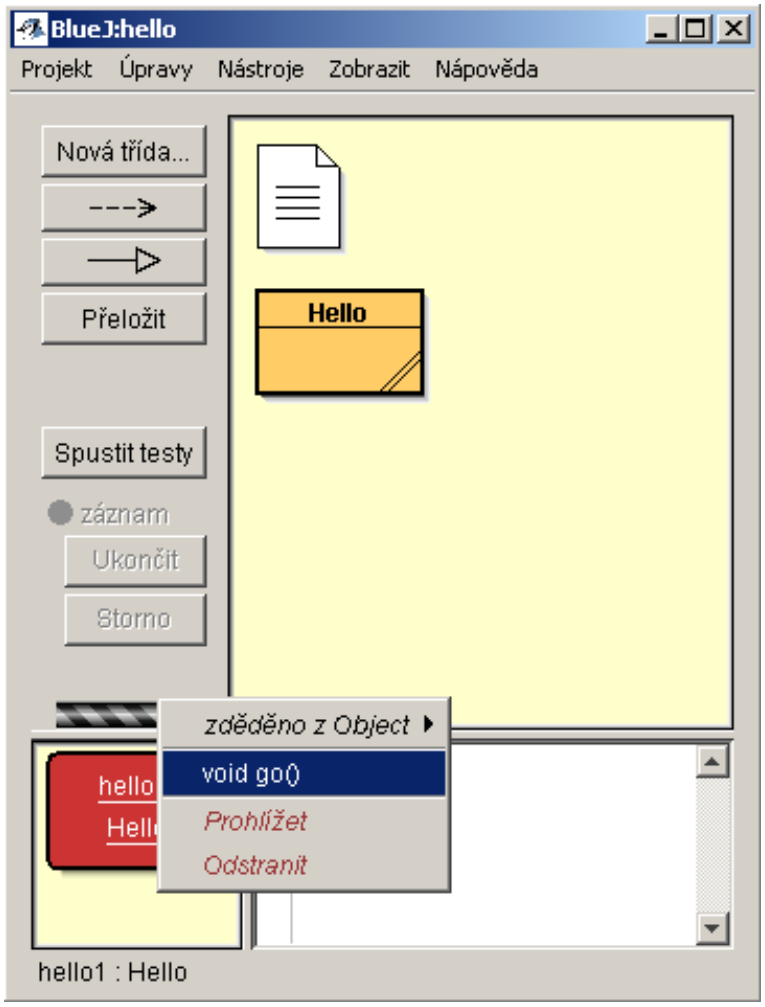
- Editor zdrojového kódu třídy Hello aktivujeme pravým tlačítkem - položkou *Otevřít v editoru*.
- Kód metody go() změníme přepsáním hlášky "Hello, world" na "Ahoj!".Po kliknutí na tlačítko Přeložit by se mělo objevit:
- Třídu Hello s upravenou metodou spustíme jako v předchozím - např. pravým tlačítkem a metodou main.Objeví se:

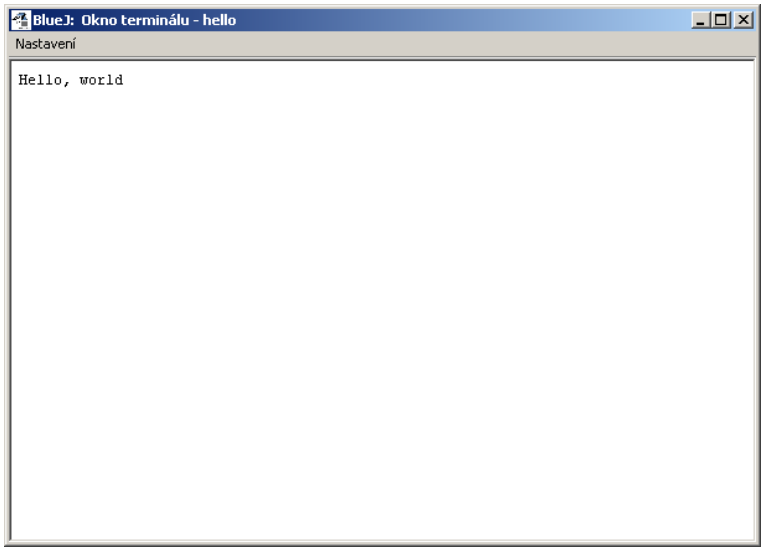
4 Sestavení a spuštění programu bez integrovaného vývojového prostředí

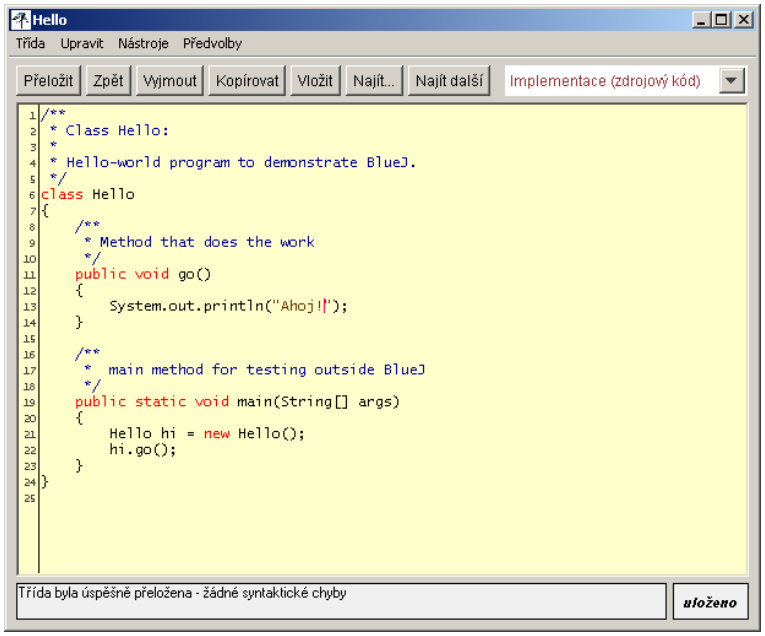
4.1 Základní životní cyklus javového programu

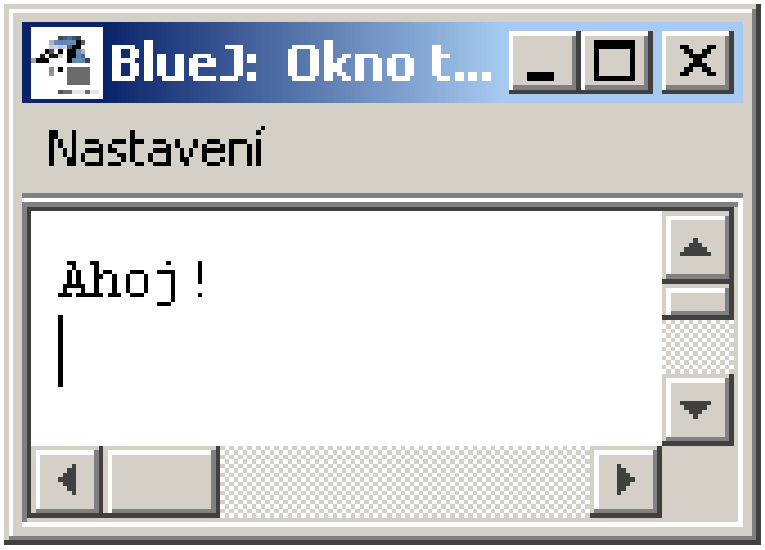
- Program sestává z jedné (ale obvykle více) **tříd** (class). Ukážeme na příkladu třídy s názvem *NazevTridy*:
- Zdrojový kód každé (veřejně přístupné, public) třídy je umístěn v jednom souboru (`NazevTridy.java`)


BlueJ:hello
Projekt
Úpravy
Nástroje
Zobrazit
Nápověda
Nová třída...
Přeložit
Hello
Spustit testy
záznam
Ukončit
Storno
hello1:
Hello
Vytváření objektu... Hotovo


BlueJ:hello
Projekt
Úpravy
Nástroje
Zobrazit
Nápověda
Nová třída...
Přeložit
Hello
Spustit testy
záznam
Ukončit
Storno
zděděno z Object
void go()
Prohlížet
Odstranit
hello
Hell
hello1 : Hello


BlueJ: Okno terminálu - hello
Nastavení
Hello, world


Hello
Třída Upravit Nástroje Předvolby
Přeložit Zpět Vyjmout Kopírovat Vložit Najít... Najít další
Implementace (zdrojový kód)
/**
* Class Hello:
*
* Hello-world program to demonstrate BlueJ.
*/
class Hello
{
/**
* Method that does the work
*/
public void go()
{
System.out.println("Ahoj!");
}
/**
* main method for testing outside BlueJ
*/
public static void main(String[] args)
{
Hello hi = new Hello();
hi.go();
}
}
Třída byla úspěšně přeložena - žádné syntaktické chyby
uloženo


BlueJ: Okno t...
Nastavení
Ahoj!

- Postup:
  - vytvoření zdrojového textu (libovolným editorem čistého textu) → `NazevTridy.java`
  - překlad (nástrojem `javac`) `NazevTridy.java` → `NazevTridy.class`
  - spuštění, např. `java NazevTridy`
- překládá se `javac NazevTridy.java` (název souboru se třídou včetně přípony `.java`!!!)
- spouští se vždy udáním java NazevTridy (název třídy bez přípony `.class`!!!)

4.2 Demo "Ahoj!"

- Půjde o podobný jednoduchý demoprogram, jaký jsme spouštěli v BlueJ pod názvem *Hello*. Dále uvedený postup (cesty v souborovém systému, spouštění programů z příkazové řádky apod. odpovídá systémům Windows.Pro Linux a Mac OS X nutné mírně modifikovat (odlišné uvádění cest v souborovém systému apod.). Princip je ale v každém případě shodný.
- Zdrojový kód bude vypadat takto:

```
public class Hello {
    // Program spouštíme aktivací funkce "main"
    public static void main(String[] args) {
        System.out.println("Ahoj!");
    }
}
```

Zdrojový kód (tj. `Hello.java`) bude umístěn do našeho pracovního adresáře, tj. např. `c:/devel/pb162`.

4.3 Překlad "Ahoj!"

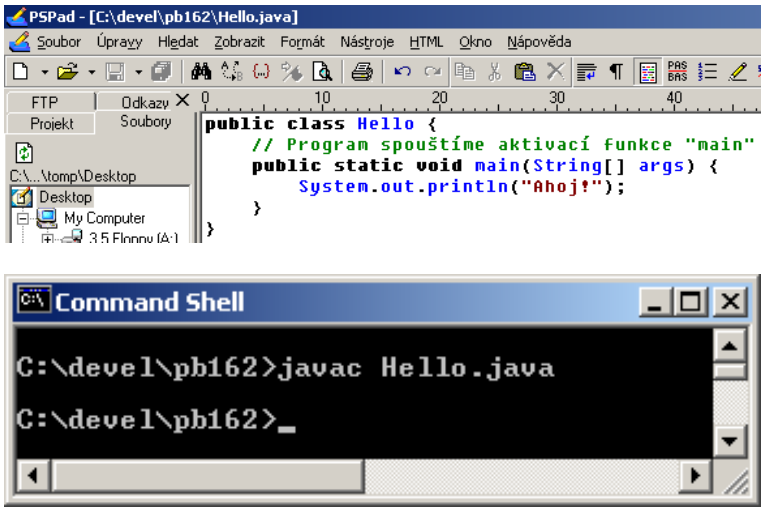
Překlad

1. Máme nainstalován Java SDK 6, 7 nebo 8
2. Jsme v adresáři `c:/devel/pb162`, v něm je soubor `Hello.java`
3. Spustíme *překlad* `javac Hello.java`
4. Je-li program správně napsán, přeloží se "mlčky"
5. (výsledný `.class` soubor bude v témže adresáři jako zdroj)

4.4 Spuštění "Ahoj!"

Spuštění

1. Poté spustíme *program* Hello: `java -classpath .  Hello`
2. Volba překladače -classpath adresář zajistí, že (dříve přeložené) třídy používané při spuštění této třídy budou přístupné pod adresářem *adresář* .



3. -classpath . tedy značí, že třídy (soubory `.class`) se budou hledat v odpovídajících podadresářích aktuálního adresáře (adresáře .)

4. Je-li program správně napsán a přeložen, vypíše se `Ahoj!`

4.5 Vytvoření zdrojového textu "Ahoj!" ("for dummies")

Vytvoření a editace zdrojového kódu v editoru PSPad (`http://pspad.zde.cz`) (dostupný zdarma, instalovaný na všech Win strojích v učebnách na FI)

4.6 Překlad "Ahoj!" ("for dummies")

Překlad překladačem javac (úspěšný, bez hlášení překladače)

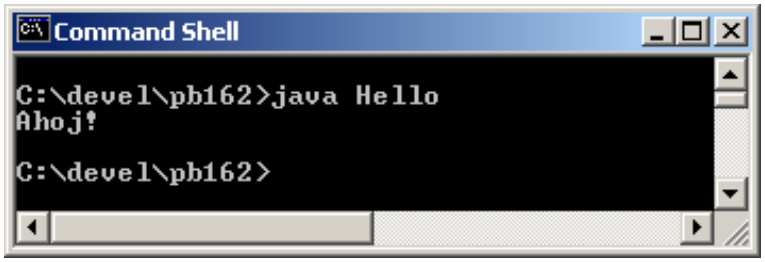
4.7 Spuštění "Ahoj!" ("for dummies")

Spuštění voláním java

4.8 Co znamená spustit program?

Spuštění javového programu= **spuštění metody *main* jedné ze tříd tvořících program**Metoda může mít parametry:

- podobně jako např. procedury a funkce v Pascalu nebo funkce v C



- jsou typu String (řetězec)
- předávají se při spuštění z příkazového řádku do pole `String[] args`

Metoda main musí být static a nevrací žádnou hodnotu - návratový typ je vždy(!) voidJejí hlavička musí *vypadat vždy přesně tak*, jako ve výše uvedeném příkladu, jinak nebude spuštěna! (Jedinou možnou odchylkou je jiné pojmenování formálního parametru args.)

4.9 Praktické informace (aneb co je nutné udělat)

Cesty ke spustitelným programům `(PATH)` musejí obsahovat i adresář `<JAVA_HOME>/bin`

4.10 Praktické informace (aneb co je vhodné udělat)

Systémové proměnné by měly obsahovat:

- `JAVA_HOME=<kořenový adresář instalace` Javy¿, např. `JAVA_HOME=c:/jdk6.0`
- Možné je nastavit i proměnnou `CLASSPATH=<cesty ke třídám¿` (podobně jako v `PATH` jsou cesty ke spustitelným souborům), např. `CLASSPATH=c:/devel/pb162`

Místo nastavení proměnné `CLASSPATH` je vhodnější volat spuštění programu (podle potřeby i překlad) s volbou -classpath.

5 Dokumentace javového kódu

5.1 Dokumentace javových programů

Základním a standardním prostředkem je tzv. *dokumentace API*

- Dokumentace je *naprosto nezbytnou součástí* javových programů.
- Rozlišujeme dokumentaci např. instalační, systémovou, uživatelskou, programátorskou...

Zde se budeme věnovat především dokumentaci programátorské, určené těm, kdo budou náš kód využívat ve svých programech, rozšiřovat jej, udržovat jej. Programátorské dokumentaci se říká *dokumentace API* (apidoc, apidocs).Při jejím psaní dodržujeme tato pravidla:

- Dokumentujeme především *veřejné* (public) a *chráněné* (protected) prvky (metody, proměnné). Ostatní dle potřeby.
- Dokumentaci píšeme *přímo do zdrojového kódu* programu *ve speciálních dokumentačních komentářích* vpisovaných *před příslušné prvky* (metody, proměnné).
- Dovnitř metod píšeme jen *pomocné komentáře* pro programátory (nebo pro nás samotné).

5.2 Typy komentářů

Podobně jako např. v C/C++:

řádkové od značky // do konce řádku, nepromítnou se do dokumentace API

blokové začínají `/*` pak je text komentáře, končí */ na libovolném počtu řádků

dokumentační od značky /** po značku */ může být opět na libovolném počtu řádkůKaždý další řádek může začínat mezerami či *, hvězdička se v komentáři neprojeví.

5.3 Kde uvádíme dokumentační komentáře

Dokumentační komentáře uvádíme:

- Před *hlavičkou třídy* - pak komentuje třídu jako celek.
- Před *hlavičkou metody nebo proměnné* - pak komentuje příslušnou metodu nebo proměnnou.
- Celý balík (package) je možné komentovat *speciálním samostatným HTML souborem*`package-summary.html` uloženým v adresáři balíku.

5.4 Generování dokumentace

Dokumentace má standardně podobu HTML stránek (s rámy i bez)Dokumentace je generována nástrojem `javadoc` z

1. dokumentačních komentářů
2. i ze samotného zdrojového textu

Lze tedy (základním způsobem) dokumentovat i program bez vložených komentářů!Chování `javadoc` můžeme změnit

1. volbami (options) při spuštění,
2. použitím jiného tzv. `docletu`, což je třída implementující potřebné metody pro generování komentářů.

Princip generování ze zdrojových textů pomocí speciálních `docletů` se dnes používá i po jiné než dokumentační účely - např. pro generátory zdrojových kódu aplikací EJB apod.

5.5 Značky javadoc

`javadoc` můžeme podrobněji instruovat pomocí značek vkládaných do dokumentačních komentářů, např.:

@author specifikuje autora API/programu

@version označuje verzi API, např. "1.0"

@deprecated informuje, že prvek je zavrhovaný

@exception popisuje informace o výjimce, kterou metoda propouští ("vyhazuje")

@param popisuje jeden parametr metody

@since uvedeme, od kdy (od které verze pg.) je věc podporována/přítomna

@see uvedeme odkaz, kam je také doporučeno nahlédnout (související věci)

5.6 Příklad zdrojového textu se značkami javadoc

Zdrojový text třídy *Window*:

```
/**
 * Klasse, die ein Fenster auf dem Bildschirm repräsentiert
 * Konstruktor zum Beispiel:
 * <pre>
 * Window win = new Window(parent);
 * win.show();
 * </pre>
 *
 * @see awt.BaseWindow
 * @see awt.Button
 * @version 1.2 31 Jan 1995
 * @author Bozo the Clown
 **/
class Window extends BaseWindow {
    ...
}
```

Příklad dokumentačního komentáře k proměnné:

```
/**
  * enthält die aktuelle Anzahl der Elemente.
  * muss positiv und kleiner oder gleich der Kapazität sein
  **/
protected int count;
```

Tyto a další příklady a odkazy lze vidět v původním materiálu JavaStyleGuide des IGE (`http://www.iam.unibe.ch/~scg/Resources/PSE/2001/WWW/projektHandbuch/codeInspections/JavaStyleGuide.html`), odkud byly ukázky převzaty.

5.7 Spouštění javadoc

- `javadoc [options] [packagenames] [sourcefiles] [classnames] [@files]`
- možné volby:
  - `-help`, `-verbose`
  - `-public`, `-protected`, `-package`, `-private` - specifikuje, které prvky mají být v dokumentaci zahrnuty (implicitně: `-protected`)
  - `-d destinationdirectory` - kam se má dok. uložit
  - `-doctitle title` - titul celé dokumentace

5.8 Konvence pro psaní komentářů

Doporučení Sun/Oracle k psaní dokumentačních komentářů – How to Write Doc Comments for the Javadoc Tool (`http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html`)