PB162 Programování v jazyce Java — podzim 2018

Tomáš Pitner, Radek Ošlejšek, Marek Šabo

Programování v jazyce Java

Profil

- Prakticky zaměřený bakalářský předmět
- Cílem je naučit základním principům objektového návrhu a programování.
- Všechny materiály k přednášce jsou v IS MU.

Předchozí předměty IB111

IB111 **Základy programování**, kde studenti získávají

- základní znalosti programování (Python)
- znalost základních příkazů, řídicí struktury, pole
- částečnou znalost objektového přístupu

Předchozí předměty PB071

PB071 **Principy nízkoúrovňového programování**, kde studenti získávají

- znalost syntaxe jazyka C
- znalost základních datových typů
- znalost vnitřních struktur

Předchozí předměty IB002

IB002 **Algoritmy a datové struktury I**, kde studenti získávají

- základy algoritmizace vč. datových struktur

Předpoklady obecně

Předpokládají se základní znalosti strukturované algoritmizace a programování, tj.:

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

Návaznosti PV168

Na tento základní kurz PB162 navazují na úrovni Bc. studia:

PV168 **Seminář z jazyka Java** (jaro)

- 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.

Návaznosti PB138

PB138 **Moderní značkovací jazyky** (jaro)

- náplní jsou XML a související technologie,
- prvky týmového vývoje (projekty, využití služeb hostování projektů, jako je **GitHub**).
- Učí kolektiv zkušených cvičících pod vedením Luďka Bártka a Tomáše Pitnera.

Návaznosti pokročilých předmětů PA165

PA165 **Vývoj aplikací v jazyce Java** (podzim)

- 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*, *Bruno Rossi*, *Martin Kuba*, *Filip Nguyen*, *Matej Briškár*, *Tomáš Skopal*.

Návaznosti — webový vývoj

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 PV247 Moderní uživatelská rozhraní
- v podzimním semestru předmět PV249 Vývoj v Ruby
- v jarním semestru PV239 Mobilní platformy
- v podzimním návazný PV256 Projekt z programování pro Android

Hodnocení a harmonogram předmětu

- Harmonogram přednášek i cvičení
- Hodnocení

O přednášejícím - Radek Ošlejšek

- pracovna **A305** (budova A1 FI) laboratoře Lasaris
- tel. `54949 6121` (z tlf. mimo budovu), kl. `6121` (volání v rámci fakulty i celé MU)
- e-mail: oslejsek(at)fi.muni.cz
- Web RO: Osobní stránka RO

O přednášejícím - Tomáš Pitner

- pracovna **A303** (budova A1 FI) laboratoře Lasaris,
- 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(at)fi.muni.cz
- Web: Osobní web TP

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

Informační zdroje (knihy)

- Rudolf Pecinovský: Myslíme objektově v jazyku Java nebo
- Java 7 — Učebnice objektové architektury pro začátečníky Grada Publishing
- Rudolf Pecinovský: Java 5.0 — Novinky jazyka a upgrade aplikací (fulltext v PDF zdarma)
- Tomáš Pitner: *Java — začínáme programovat*, Grada Publishing, 2002, doprovodný web knihy (učebnice je orientovaná na Javu 1.4 a nižší; většina poznatků je platných i nadále, ale v Javě 5

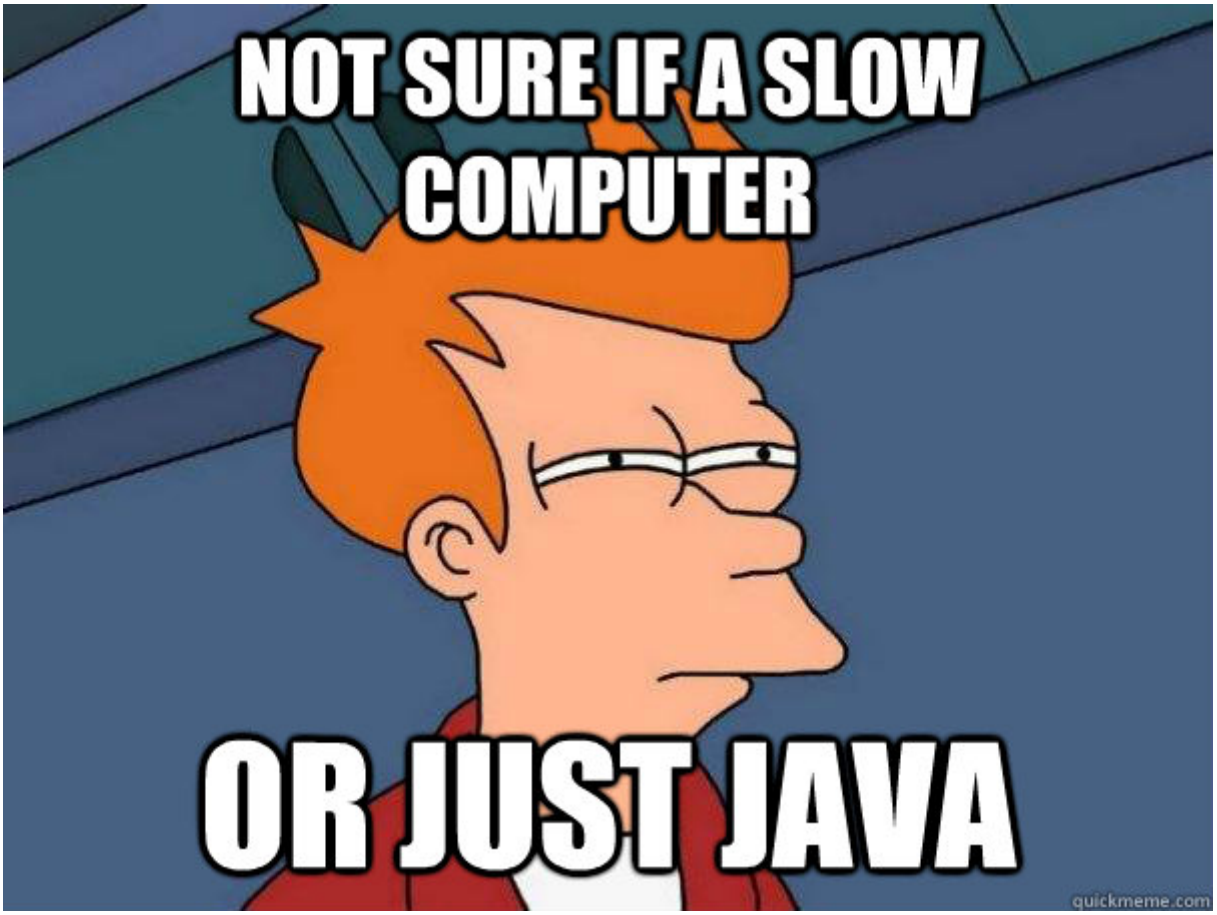
až 8 se objevila řada nových prvků)

- Pavel Herout: *Učebnice jazyka Java*, Kopp, 2000-2010, doprovodný web knihy
- příp. i Pavel Herout: *Java — grafické uživatelské rozhraní a čeština*, Kopp, 2001 — pro pokročilé

Informační zdroje (knihy)

- Bruce Eckel: Myslíme v jazyce Java — příručka programátora, Grada Publishing , 2000
- příp. Bruce Eckel: Myslíme v jazyce Java — příručka zkušeného programátora, Grada Publishing , 2000 — pro pokročilé
- Joshua Bloch: Java efektivně — 57 zásad softwarového experta, Grada Publishing
- Bogdan Kiszka: 1001 tipů a triků pro programování v jazyce Java, Computer Press, 2003
- Bruce Eckel: Thinking in Java Stáhnout zdarma (PDF) = Úvod do jazyka a prostředí Java = :course: PB162 :year: 2018 :term: podzim :description: Lecture slides for PB162 course taught at Masaryk University, Faculty of Informatics since 2001 :Author: Tomáš Pitner, Radek Ošlejšek, Marek Šabo :copyright: © 2001-2018 Tomáš Pitner, Masaryk University — PB162 Java :slideshowlocation: Masaryk University, Brno, Czech Republic :date: podzim 2018 :data-uri: :keywords: Java, object programming :email: tomp@fi.muni.cz :slidebackground: asciidocslidy

Rychlost Javy



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* — program používá volání metod objektů (zasílání zpráv objektům)
- Ideovým předchůdcem Javy je *C++*
- Svým způsobem je Java obdobou C++, ale zbavena zbytečností a nepříjemností

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*, *Kotlin*…)
- Mnoho jazyků bylo inspirovaných Javou: *C#*, *Groovy*, *Ruby*, *Scala*, z nových *Go* a *Kotlin*…

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.

Další charakteristiky

- Java *podporuje vytváření správných návyků* v objektovém programování a naopak systematicky brání přenosu některých špatných návyků z jiných jazyků.
- Program v Javě je *přenositelný* na úrovni *zdrojového i přeloženého* kódu.
- Přeložený javový program běží v tzv. Java Virtual Machine (JVM).
- Zdrojový i přeložený kód je tedy přenositelný mezi všemi obvyklými platformami (UNIX, Windows, Mac OS X).

Java pro programátora

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), což zlepšuje čitelnost a redukuje riziko chyb.

Aktuální verze Javy SE je **Java 8**.

Aktuální verze

Stav k září 2018:

- *Java Standard Edition 8* (u zákazníků s *Long Term Support* pokračují i SE 6 a 7)
- je stabilní verzí pro všechny platformy.
- U Java 7 běžná podpora skončila dubnem 2015.
- Aktuální informace najdete vždy na webu Oracle Oracle Technetwork/Java.
- K předpokládanému vývoji existuje Oracle roadmap

Stažení Javy

- Na webových stránkách Oracle je java dostupná ve všech platformách.
- Chceme vývojové prostředí (JDK), běhové prostředí (JRE) slouží jenom na spuštění, ne na vývoj.

Lze stáhnout:

- samotné *vývojové prostředí* (JDK), např. Java SE 8 JDK
- jen *běhové prostředí* (JRE), např. Java SE 8 JRE: to nám tady nestačí, chceme vyvíjet
- JDK v balíčku s grafickým (okénkovým) *integrovaným vývojovým prostředím* (IDE, Integrated Development Environment) NetBeans.

Připravili jsme pro vás tutoriál, jak Javu nainstalovat.

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.

Budoucnost Javy

- Na závěr optimistického úvodu si pročtěte zajímavý článek analytika od *Forrester*: Java Is A Dead-End For Enterprise App Development = První program v Javě, třída, objekt :course: PB162 :year: 2018 :term: podzim :description: Lecture slides for PB162 course taught at Masaryk University, Faculty of Informatics since 2001 :Author: Tomáš Pitner, Radek Ošlejšek, Marek Šabo :copyright: © 2001-2018 Tomáš Pitner, Masaryk University — PB162 Java :slideshowlocation: Masaryk University, Brno, Czech Republic :date: podzim 2018 :data-uri: :keywords: Java, object programming :email: tomp@fi.muni.cz :slidebackground: asciidocslidy

Program "Hello World!"

- Abychom měli kam náš kód psát, vytvoříme třídu Demo s hlavní funkci main, která se zavolá při spuštění programu.

```
public class Demo {
   public static void main(String[] args) {
      System.out.println("Hello World!");
   }
}
```

- Metoda main musí být **veřejná** (public), **statická** (static) a **nevrací žádnou hodnotu** (void). *Klíčová slova pochopíte časem, není to teď důležité.*
- Metoda musí mít parametry typu String (řetězec), které se předávají při spuštění z příkazového řádku do pole String[] args.

Motivace třídy I

- Jak reprezentovat složitou strukturu, aby se s ní dobře pracovalo?
- Příklad: Osoba s *jménem* a *rokem narození*

```
class Person {
   String name;
   int yearBorn;
}
```

- Části objektu nastavíme i zjistíme stejným způsobem jako v jazyce Python:

```
k.name = "Karel"; // set name to Karel
String karelsName = k.name; // get name value
```

Jednotlivé části (jméno, rok narození) nazýváme *atributy*.

Motivace třídy II

- Někdy bychom rádi měli funkce, které pracují přímo s částmi struktury.
- Pamatujeme si rok narození, ale co když chceme zjistit věk?
- Jak lehce zjistit informace o naší ~~struktuře~~ — *třídě*?

```
public class Person {
   private String name;
   private int yearBorn;
   public int getAge() {
     return 2018 - yearBorn;
   }
   public void printNameWithAge() {
     System.out.println("I am " + name + " and my age is " + getAge());
   }
}
```

V kódu třídy se nyní objevila klíčová slova `public` a `private`. Nemají vliv na funkcionalitu, ale na "viditelnost", na možnost či nemožnost z jiného kódu danou třídu nebo její vlastnost vidět a použít. Logicky `public` asi (určitě :)) půjde použít vždy a odevšad.

Vlastnosti třídy

- Třída představuje strukturu, která má *atributy* a *metody*.

Atributy

- jsou nositeli datového obsahu, údajů, "pasivních" vlastností objektů
- to, co struktura má, *z čeho se skládá*, např. auto se skládá z kol
- definují **stav** objektu, nesou **informace** o objektu

Metody

- jsou nositeli "výkonných" vlastností, *schopností* objektů něco udělat
- to, *co dokáže struktura dělat* — pes dokáže štěkat, osoba dokáže mluvit
- definují **chování** objektu (může být závislé na stavu)

Vytvoření konkrétní osoby

- Máme třídu Person, to je něco jako *abstraktní šablona* pro objekty—osoby.
- Jak vytvořím **konkrétní** osobu s jménem Jan?

```
public class Demo {
 public static void main(String[] a) {
   Person jan = new Person();
   jan.name = "Jan";
   jan.yearBorn = 2000;
   System.out.println(jan.name);
   System.out.println(jan.yearBorn);
 }
}
```

== Poznámky k příkladu `Demo` - **Třída** Person má vlastnost name a age, to jsou její *atributy*. - **Objekt** jan typu Person má vlastnost name s hodnotou *Jan* a yearBorn s hodnotou *2000*. - Klíčová slova `public` a `private` vám z Pythonu nejsou známá, zde v Javě i jiných jazycích označují "viditelnost" položky — jednoduše řečeno, co je veřejné a co soukromé. Soukromé atributy "vidíme" jen z metod třídy, v níž jsou uvedeny.

== Objekt

- Objekt je jeden **konkrétní jedinec** příslušné třídy.
- Všechny vytvořené objekty nesou stejné vlastnosti, např. všechny objekty třídy Person mají vlastnost name.
- Vlastnosti mají však pro různé lidi různé hodnoty — lidi mají různá jména.
- Konkrétní objekt určité třídy se také nazývá *instance* (jedincem) své třídy.

== Vytváření objektů

- Co znamená new Person()?
- Proč musíme psát Person jan = new Person() a ne jen `Person jan`?

```
Person jan = new Person();
// why not just:
Person jan;
```

- Pouhá deklarace proměnné objektového typu (`Person jan`) žádný objekt nevytvoří.
- K vytvoření slouží operátor `new`.

== Co se děje při vytváření objektů přes `new`

- Alokuje se paměť v oblasti dynamické paměti, tedy na *haldě* (heap).
- Vytvoří se tam objekt a naplní jeho atributy výchozími hodnotami.

- Zavolá se speciální metoda objektu, tzv. konstruktor, který objekt dotvoří.

== Konstruktor

- Slouží k "oživení" vytvořeného objektu bezprostředně po jeho vytvoření:
  - Jednoduché typy, jako například `int`, se vytvoří a inicializují samy a konstruktor nepotřebují.
  - Složené typy, *objekty*, je potřeba vždy zkonstruovat!
- V našem příkladu s osobou operátor `new` *vytvoří prázdný objekt typu Person* a naplní jeho atributy výchozími (default) hodnotami.
- Další přednáška bude věnována konstruktorům, kde se dozvíte víc.

== Třída a objekt

Třída

- Je komplexní struktura, reprezentuje prvky z reálného světa (např. pes, člověk).
- Je určitý vzor pro tvorbu podobných objektů (konkrétních psů či lidí).
- Definice třídy sestává převážně z *atributů* a *metod* (říkáme jim také prvky nebo členy třídy).
- Skutečné objekty této třídy pak budou mít prvky, které byly ve třídě definovány.

Objekt

- Objekty jsou instancemi "své" třídy vytvořené dle definice třídy a obsahující atributy.
- Vytváříme je operátorem `new`.
- Odkazy na vytvořené objekty často ukládáme do proměnné typu té třídy, např. `Person jan = new Person();`

== Komplexnější příklad I ==

Následující třída `Account` modeluje jednoduchý bankovní účet.

- Každý bankovní účet má jeden *atribut* `balance`, který reprezentuje množství peněz na účtu.
- Pak má *metody*:
  - `add` přidává na účet/odebírá z účtu
  - `writeBalance` vypisuje zůstatek
  - `transferTo` převádí na jiný účet

== Komplexnější příklad II ==

public class Account { private double balance; // 0.0 public void add(double amount) { balance += amount; } public void writeBalance() { System.out.println(balance); } public void transferTo(Account whereTo, double amount) { balance -= amount; whereTo.add(amount); // whereTo is another account } }

- Metoda `transferTo` pracuje nejen se svým "mateřským" objektem, ale i s objektem `whereTo` předaným do metody.

== Komplexnější příklad - definice vs. použití třídy ==

- Třída sama je definovaná v samostatném souboru `Account.java`.
- Její použití pak třeba v `Demo.java`.

public static void main(String[] args) { Account petrsAccount = new Account(); Account ivansAccount = new Account(); petrsAccount.add(100.0); ivansAccount.add(20.0); petrsAccount.transferTo(ivansAccount, 30.0); petrsAccount.writeBalance(); // prints 70.0 ivansAccount.writeBalance(); // prints 50.0 }

== `println` vs. `return` ==

- Pozor na rozdíl mezi vypsáním řetězce a jeho vrácením:

```
public void writeString() {
  System.out.println("Sample text"); // writes it
}
```

```
public String returnString() {
  return "Sample text"; // does not write it
}
```