Seminář Java

I

Radek Kočí

Fakulta informačních technologií VUT

Únor 2011

Téma přednášky

- Organizace semináře
- Java – úvod, distribuce
- Základy objektové orientace
  - abstrakce
  - zapouzdření
  - objekty a třídy
- Vytváření objektů

Informace, studijní materiály

Stránky předmětu

- `http://www.fit.vutbr.cz/study/courses/IJA/`
- zadání úkolů a projektu, informace
- konzultace
- studijní materiály

Diskuzní fóra (dostupná v IS)

- diskuze problémů při řešení úkolů a projektu

Hodnocení předmětu: zápočet

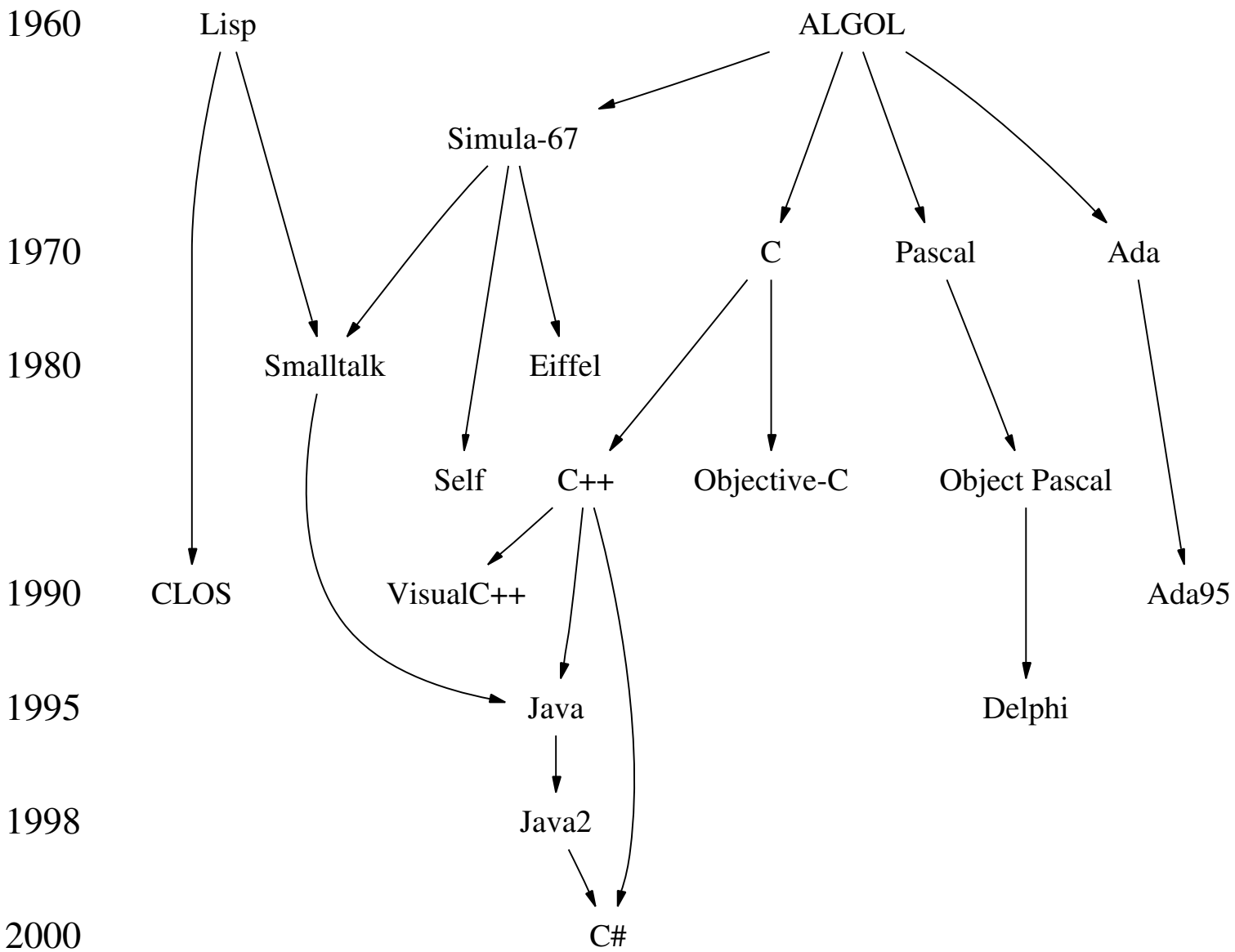
- alespoň 1 bod z každého úkolu
- alespoň 50% bodů z projektu

Informace, studijní materiály

Projekt

- tým mající 2 členy
- práce s svn
- je třeba aktivovat účet
- je třeba registrovat týmy

Přehled jazyků



Programovací jazyk Java

Základní charakteristika

- objektově orientovaný
- statická typová kontrola
- Java Virtual Machine – JVM
  - program v Javě je meziplatformně přenositelný na úrovni zdrojového i přeloženého kódu
  - automatické odklizení nepoužitelných objektů (automatic garbage collection)

Základní charakteristika

- dostupné velké množství knihoven pro různorodé aplikační oblasti, např. na SourceForge, ...
- k dispozici je řada kvalitních vývojových prostředí, např. *NetBeans*, *JBuilder*, *Visual Age for Java*, *Eclipse*, *IDEA*

Srovnání (názory)

- Java vs. C++ (`http://c2.com/cgi/wiki?JavaVsCpp`)
- Java vs. Smalltalk (`http://c2.com/cgi/wiki?JavaVsSmalltalk`)

Využití Javy

- vícevláknové aplikace (multithreaded applications)
- škálovatelné výkonné aplikace běžící na serverech (*Java Enterprise Edition*)
- aplikace na přenosných a vestavěných zařízeních (*Java Micro Edition*)
- webové aplikace (*servlety*, *JSP*) – alternativa proprietárních *ASP*, *SSI*, *CGI*
- zpracování semistrukturovaných dat (*XML*)
- přenositelné aplikace s GUI
- aplikace distribuované po síti (*Applets* nebo *Java Web Start*)

Typy aplikací

- Konzolové aplikace
  - jednoduchá textová konzole
- GUI aplikace
- Applety
  - běží v HTML prohlížečích
  - mají silná bezpečnostní omezení

Java – platforma

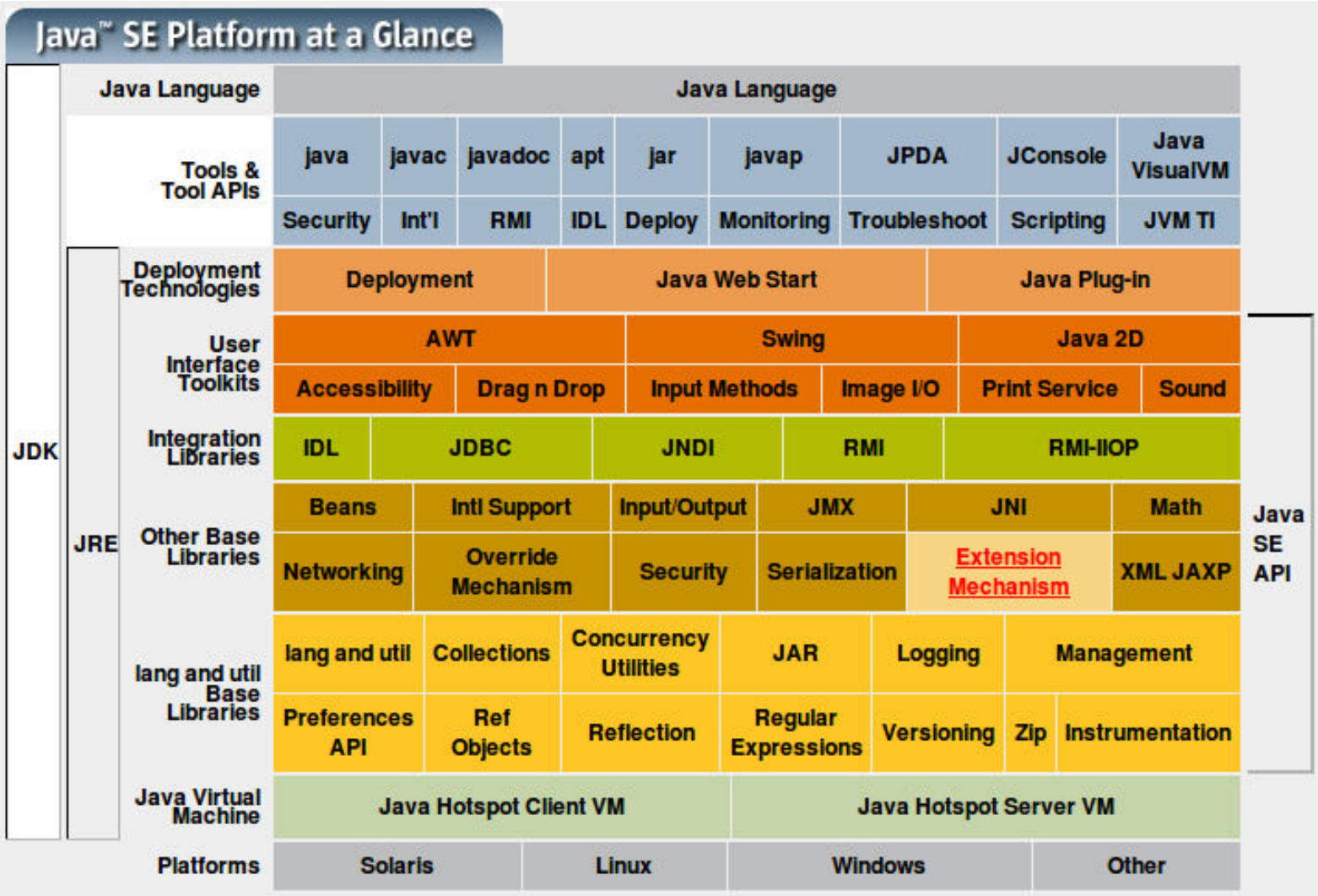
Java platformu tvoří:

- Java Virtual Machine (JVM)
- překladač a další vývojové nástroje
- Java Core API (základní knihovna tříd)

Java je tedy dána...

- definicí jazyka (Java Language Definition) – syntaxe a sémantika jazyka
- popisem chování JVM
- popisem Java Core API

Java – platforma



Převzato z http://java.sun.com

Specifikace a implementace Javy

Specifikace Javy

- Standard Edition
- Enterprise Edition
- Micro Edition

Implementace Javy

- Java Development Kit – obsahuje vývojové nástroje
- Runtime Enviroment – obsahuje jen běhové prostředí pro spouštění hotových přeložených programů

Verze Javy

Hrubé členění

- verze Java (před Java 2, v1.2)
- verze Java 2
- verze Java (po Java 2, v1.5)

Číslování verzí (dříve)

- major číslo (např. Java 2, v1.4)
  - při změně major čísla se může měnit Core API a někdy i jazyk
- minor číslo (např. Java 2, v1.4.2)
  - změnu minor (třetího) čísla doprovází jen odstraňování chyb
- J2SE $\Rightarrow$ Java SE

Verze Javy

version	*code name*	*release date*
JDK 1.1.4	Sparkler	Sept 12, 1997
JDK 1.1.5	Pumpkin	Dec 3, 1997
JDK 1.1.6	Abigail	April 24, 1998
JDK 1.1.7	Brutus	Sept 28, 1998
JDK 1.1.8	Chelsea	April 8, 1999
J2SE 1.2	Playground	Dec 4, 1998
J2SE 1.2.1	*(none)*	March 30, 1999
J2SE 1.2.2	Cricket	July 8, 1999
J2SE 1.3	Kestrel	May 8, 2000
J2SE 1.3.1	Ladybird	May 17, 2001
J2SE 1.4.0	Merlin	Feb 13, 2002
J2SE 1.4.1	Hopper	Sept 16, 2002
J2SE 1.4.2	Mantis	June 26, 2003

Verze Javy

version	*code name*	*release date*
J2SE 5.0 (1.5.0)	Tiger	Sept 29, 2004
Java SE 6	Mustang	Dec 11, 2006
Java SE 7	Dolphin	

Java Technology History

- 1990 – Green Project
- 1992 – OAK, použitý na PDA
- 1995 – první verze Javy, Java pro Netscape
- 1996 – Java 1.0, další podpora Javy
- 1997 – Java 1.1, Java Web Server
- 1999 – XML, NetBeans (Praha), J2SE, J2EE, J2ME
- 2004 – Java SE 5
- 2006 – Java SE 6
- 2007 – JDK uvolněno pod GPLv2 – OpenJDK
- 2009 – Java SE 7 ve vývoji – OpenJDK
- 2010 – Akvizice Sun Microsystems firmou Oracle
- 2010 – Spolupráce Oracle a IBM na OpenJDK

Internetové zdroje

- `http://java.sun.com`
  $\Rightarrow$ `http://www.oracle.com/technetwork/java/`
- `http://openjdk.java.net`
- `http://jdk7.java.net/`

Distribuce Javy

Podmínky získání a používání

- vývoj různých distribucí (Sun Microsystems, IBM, Open Source, ...), dnes hlavní proud soustředěn pod Oracle
- používání Javy včetně redistribuce JRE pro běžný vývoj je zdarma
- používání a redistribuce JDK včetně zdrojových kódů se řídí licencí (různé licence)

Stažení distribuce Sun

- `http://java.sun.com` (pro Windows, Solaris, Linux)
- dokumentace se stahuje z téhož místa, ale samostatně (nebo lze číst z WWW)

Obsah vývojové distribuce Javy

Obsah adresářů

- `bin` – vývojové nástroje (Development Tools) určené k vývoji, spouštění, ladění a dokumentování programů v Javě.
- `jre` – běhové prostředí Javy (Java Runtime Environment); obsahuje Java Virtual Machine (JVM), knihovnu tříd Java Core API a další soubory potřebné pro běh programů v Javě
- `lib` – přídavné knihovny (Additional libraries) jsou další knihovny nutné pro běh vývojových nástrojů
- `demo` – ukázkové applety a aplikace (Demo Applets and Applications); příklady zahrnují i zdrojový kód

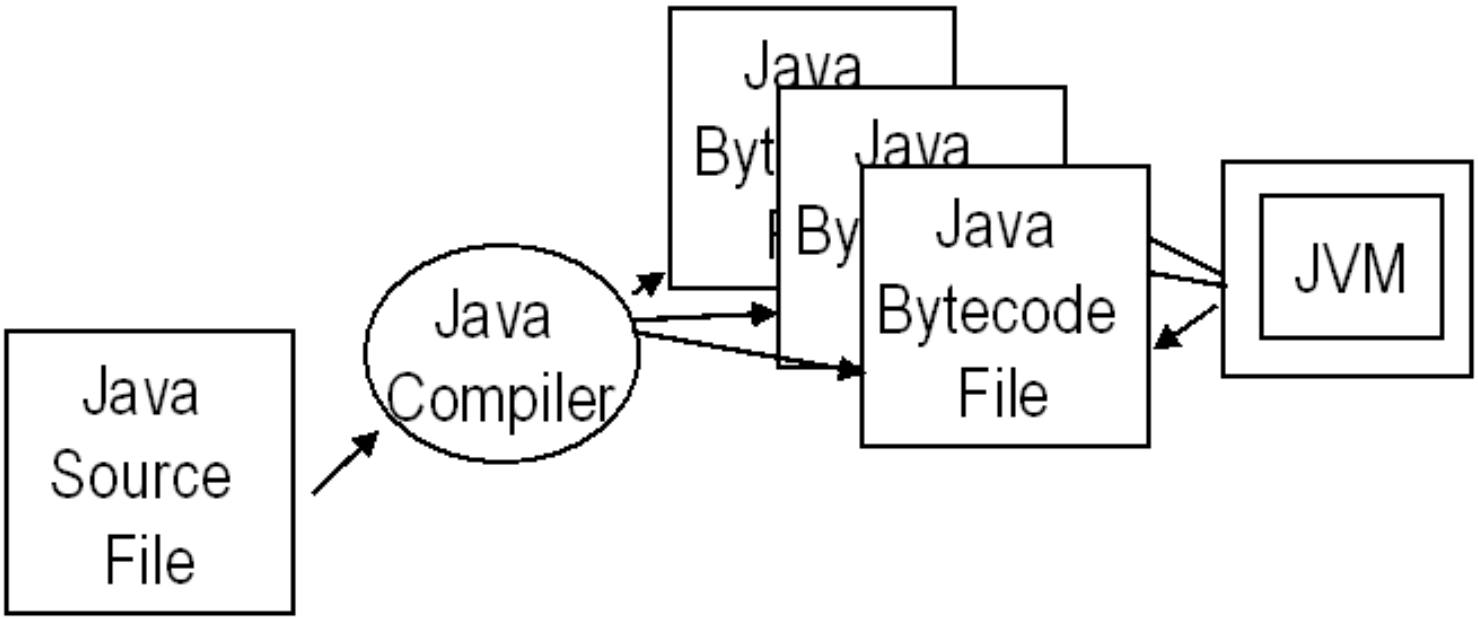
Nástroje ve vývojové distribuci

Pod Windows jsou to .exe soubory umístěné v podadresáři bin

- `java` – spouštěč (přeloženého bajtkódu)
- `javac` – překladač (.java -> .class)
- `javadoc` – generátor dokumentace API
- `jar` – správce archivů JAR (sbalení, rozbalení, výpis)
- `jdb` – debugger
- `appletviewer` – referenční prostředí pro spouštění appletů

Java Virtual Machine

- Překladač generuje byte-kód pro JVM
- JVM interpretuje byte-kód
- Optimalizace (JIT)



Co je nutné udělat

- Cesty ke spustitelným programům (`PATH`) musejí obsahovat i adresář `$JAVA_HOME/bin`

Co je vhodné udělat

Systémové proměnné by měly obsahovat:

- `JAVA_HOME` = kořenový adresář instalace Javy, např. `JAVA_HOME=/usr/local/java`
- `CLASSPATH` = cesty ke třídám (podobně jako v `PATH` jsou cesty ke spustitelným souborům), např. `CLASSPATH=$HOME/java`

Distribuce Javy na FIT

`merlin.fit.vutbr.cz`

- Java SE 6 (1.6.0_23) — `/usr/local/share/Java`
- Netbeans 6.8
- Ant 1.7.1

- základní systémové proměnné jsou nastavené

Distribuce Javy na FIT

Test spuštění Javy

- `javac -version`
- `java -version`
- `java -client -version`

Základy objektové orientace

Základy objektové orientace

Základní vlastnosti objektové orientace

- Abstrakce (abstraction)
- Zapouzdření (encapsulation)
- *Polymorfismus (polymorphism)*
- *Dědičnost (inheritance)*

Vlastnosti objektové orientace – Abstrakce

Abstrakce

- vytvářený systém objektů je abstrakcí řešeného problému
- analýza problému $\Rightarrow$ klasifikace do abstraktních struktur $\Rightarrow$ objekty $\Rightarrow$ třídy
- objekt je abstrakcí (zjednodušením) části řešené domény, má definovanou zodpovědnost za řešení části problému

Vlastnosti objektová orientace

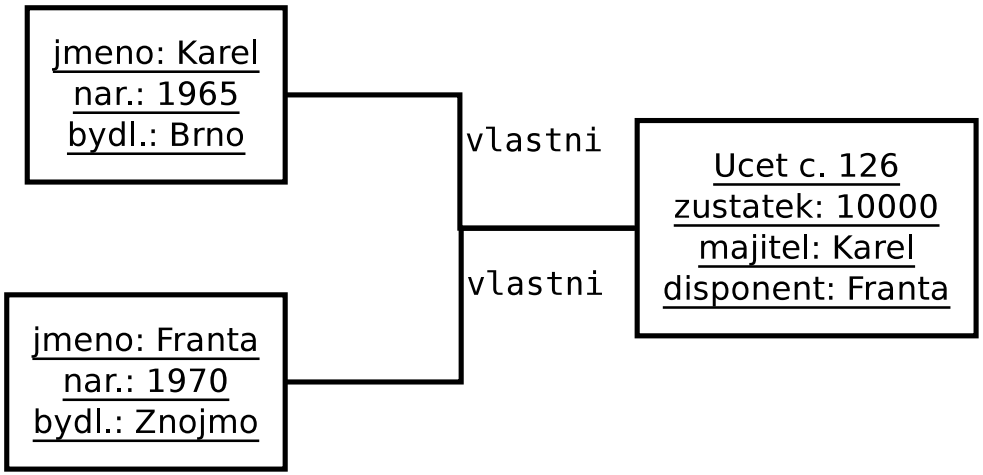
Třída

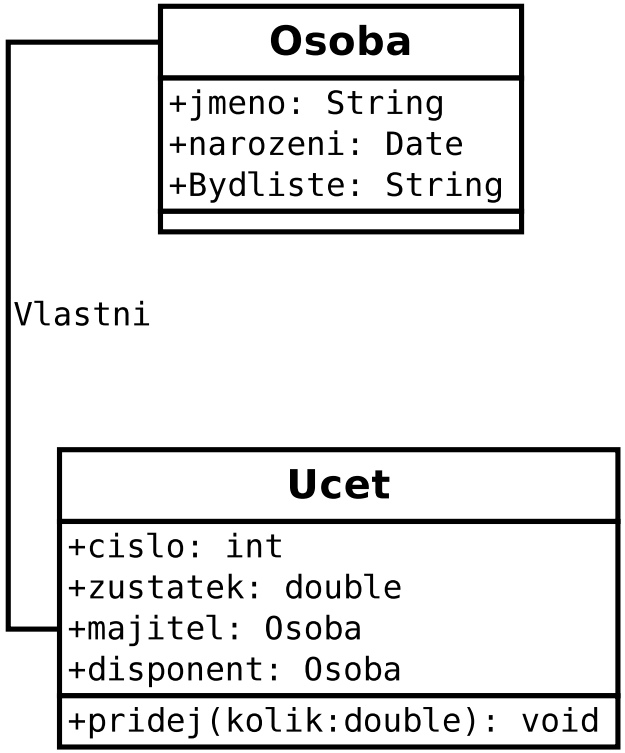
- vzor popisující strukturu a chování objektů stejného druhu
- množina objektů stejného druhu
- deklaruje proměnné (atributy) a metody *objektu*
- může deklarovat proměnné (atributy) a metody *třídy*

Objekt

- instance třídy
- objekty mají vlastní data (atributy) – kopie
- objekty sdílí chování – metody

Vlastnosti objektové orientace – Abstrakce





Vlastnosti objektové orientace – Zapouzdření

Zapouzdření

- Seskupení souvisejících idejí do jedné jednotky, na kterou se lze následně odkazovat jediným názvem (objekt).
- Objektově orientované zapouzdření je seskupení operací a atributů (reprezentujících stav) do jednoho typu objektu. Stav je pak dostupný či modifikovatelný pouze prostřednictvím rozhraní (operace, metody).
- Omezení externí viditelnosti informací nebo implementačních detailů.
- Zaručené rozhraní.

Vlastnosti objektové orientace – Zapouzdření

```
int obsah(int x, int y) {
    return x * y;
}
```

```
struct Obdelnik {
    int x, y;
}
int obsah(struct Obdelnik o) {
    return o.x * o.y;
}
```

```
struct Obdelnik {
    int x, y;
    int obsah() { return x * y; }
}
```

Vlastnosti objektové orientace – Zapouzdření

```
int obsah(int x, int y) {
    return x * y;
}
```

```
struct Obdelnik {
    int x, y;
}
int obsah(struct Obdelnik o) {
    return o.x * o.y;
}
```

```
struct Obdelnik {
    int x, y;
    int obsah() { return x * y; }
}
```

Vlastnosti objektové orientace – Zapouzdření

```
int obsah(int x, int y) {
    return x * y;
}
```

```
struct Obdelnik {
    int x, y;
}
int obsah(struct Obdelnik o) {
    return o.x * o.y;
}
```

```
struct Obdelnik {
    int x, y;
    int obsah() { return x * y; }
}
```

Vlastnosti objektové orientace – Zapouzdření

```
class Obdelnik {
    int x;
    int y;
    int obsah() { return x * y; }
}
```

Ukrývání implementačních detailů, omezení přístupu k vlastnostem tříd (zajištění integrity dat)

```
public class Obdelnik {
    protected int x;
    protected int y;
    public int obsah() { return x * y; }
}
```

Vlastnosti objektové orientace – Zapouzdření

```
class Obdelnik {
    int x;
    int y;
    int obsah() { return x * y; }
}
```

Ukrývání implementačních detailů, omezení přístupu k vlastnostem tříd (zajištění integrity dat)

```
public class Obdelnik {
    protected int x;
    protected int y;
    public int obsah() { return x * y; }
}
```

Vlastnosti objektové orientace – Zapouzdření

```
class Obdelnik {
    int x;
    int y;
    int obsah() { return x * y; }
}
```

Ukrývání implementačních detailů, omezení přístupu k vlastnostem tříd (zajištění integrity dat)

```
public class Obdelnik {
    protected int x;
    protected int y;
    public int obsah() { return x * y; }
}
```

Základní pojmy – Objektové rozhraní

Operace vs. metoda

- množina operací reprezentuje chování objektu
- metoda implementuje operaci

Rozhraní objektu

- množina operací, které objekt nabízí
- pouze definuje *co* objekt umí (nabízí)

Příklad

```
public class Obdelnik {
    public int obsah() { ... }
    public int obvod() { ... }
}
```

```
public interface Ctyruhelnik {
    public int obsah();
    public int obvod();
}

public class Obdelnik implements Ctyruhelnik {
    public int obsah() { // implementace operace }
    public int obvod() { // implementace operace }
}
```

Příklad

```
public class Obdelnik {
    public int obsah() { ... }
    public int obvod() { ... }
}
```

```
public interface Ctyruhelnik {
    public int obsah();
    public int obvod();
}

public class Obdelnik implements Ctyruhelnik {
    public int obsah() { // implementace operace }
    public int obvod() { // implementace operace }
}
```

Objekty a komunikace

Komunikace objektů

- objekty spolu komunikují zasíláním zpráv
- příjemce chápe zprávu jako požadavek na provedení služby (operace)
- zpráva obsahuje *identifikátor příjemce*, *název operace* a *argumenty*
- obsluha zprávy (protokol) vyhledá implementaci operace (metody) a provede ji
- po ukončení obsluhy může metoda vracet výsledek

Vytvoření objektu (instance třídy) a zaslání zprávy

- ```
  Obdelnik o = new Obdelnik();
  int obsah = o.obsah();
  ```
  ```

Vytváření objektů

Voláním `new Obdelnik()` jsme použili:

- operátor `new`, který vytvoří prázdný objekt a
- volání *konstruktoru*, který prázdný objekt naplní počátečními údaji (daty).

Konstruktory

- Konstruktury jsou speciální metody volané při vytváření nových instancí dané třídy.
- Typicky se v konstruktoru naplní (inicializují) proměnné objektu.
- Konstruktory lze volat jen ve spojení s operátorem `new` k vytvoření nové instance třídy – nového objektu, eventuálně volat z jiného konstruktoru.

Implicitní konstruktor

Každá třídá má *implicitní* (*bezparametrický*) konstruktor

- nemá žádné parametry
- nemá žádný návratový typ!
- nemusí se deklarovat
- deklarace: `JmenoTridy() {...}`

```
public class Obdelnik {
    public Obdelnik() {
        ...
    }
}
```

Použití: `new Obdelnik();`

Další konstruktory

Každá třída může mít další (jiné) konstruktory než implicitní

- odlišují se parametry
- pokud se deklaruje alespoň jeden konstruktor, implicitní se již negeneruje!!

```
public class Obdelnik {
    protected int x;
    protected int y;
    public Obdelnik(int x, int y) {
        this.x = x;
        this.y = y;
    }
}
```

Použití:

```
new Obdelnik(50, 20);
new Obdelnik();          ⇐ chyba!
```

Další konstruktory

Pokud chceme deklarovat další konstruktory a současně používat implicitní, musíme ho také deklarovat!

```
public class Obdelnik {
    public Obdelnik() { }
    public Obdelnik(int x, int y) {
        this.x = x;
        this.y = y;
    }
}
```

Použití:

```
new Obdelnik(50, 20);
new Obdelnik();
```

$\Leftarrow$ OK (ale zbytečné)