Seminář Java

I

Program

- Organizace semináře
- Základní principy OOP
- Úvod do programovacího jazyka Java
- Distribuce
- Demonstrační příklad

Informace, studijní materiály

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

Proces Objektově orientované tvorby

- Objektově orientovaná analýza
  - Porozumění řešené doméně
- Objektově orientovaný návrh
  - Návrh řešení, model domény (struktura, aktivity)
- Objektově orientované programování
  - Implementace řešení
- Dobrý návrh tvoří 2/3 práce ...
- Je to proces, ne vodopád ...
- OOA je jazykově nezávislá

Základy objektové orientace

- Objektově orientované systémy:
  - kompozice doménově specifických objektových abstrakcí
  - sloučení dat a funkčnosti do objektu
  - objekty komunikují zasíláním zpráv

- Vlastnosti objektové orientace
  - Abstrakce (abstraction)
  - Zapouzdření (encapsulation)
  - Polymorfismus (polymorphism)
  - Dědičnost (inheritance) – Hierarchie (hierarchy)

Abstrakce

- zjednodušený pohled na reálný objekt $\Rightarrow$ objektová abstrakce domény
- rozpoznávání podobností
- sloučení (kompozice) objektových abstrakcí (objektů) do softwarového systému
- relativní

Zapouzdření

- Seskupení souvisejících idejí do jedné jednotky, na kterou se lze následně odkazovat jediným názvem.
- 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.

- ukrývání detailů
- zaručené rozhraní

Objekt

Objekt je doménový koncept mající:

- atributy
  - atribut je vlastnost objektu
  - atribut není proměnná (i když je tak většinou deklarován)
  - atribut datum (dd/mm/rr) $\Rightarrow$ počet sekund od LP 1970
  - atribut objem $\Rightarrow$ součin tří hodnot
- stav
  - je reprezentován množinou hodnot atributů
  - v každém okamžiku je objekt v definovatelném stavu
- chování
  - operace (množina metod)
- identitu

Zpráva

- prostředek komunikace mezi objekty
- identifikátor objektu (příjemce zprávy)
- název operace, kterou má příjemce vykonat
- argumenty operace
- přímý výsledek zpracování zprávy

Java:

```
identifikator.pridej(100);
```

Smalltalk:

```
identifikator pridej: 100.
```

Role objektů ve zprávách

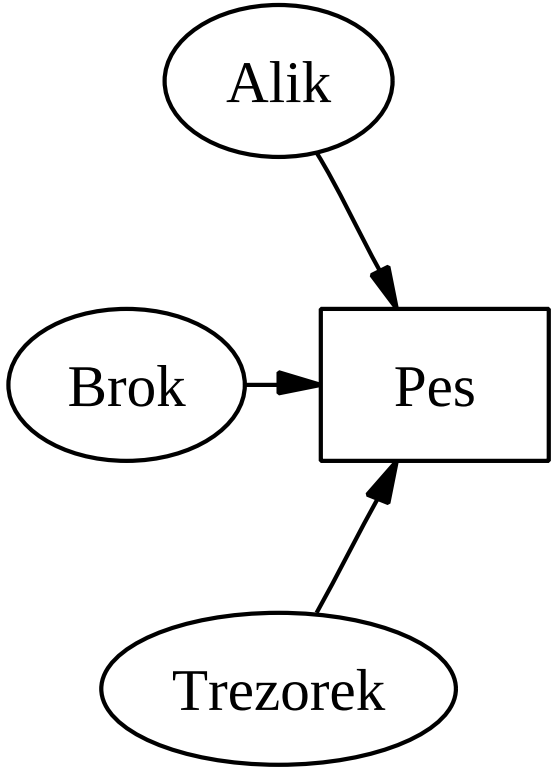
Čistá objektově orientovaná prostředí (např. Smalltalk) mají pouze objekty, které hrají jednu z těchto rolí:

- odesílatel zprávy
- cíl zprávy
- odkazován proměnnou v jiném objektu
- odkazován argumentem zprávy

V hybridních prostředích existují kromě objektů i datové typy.

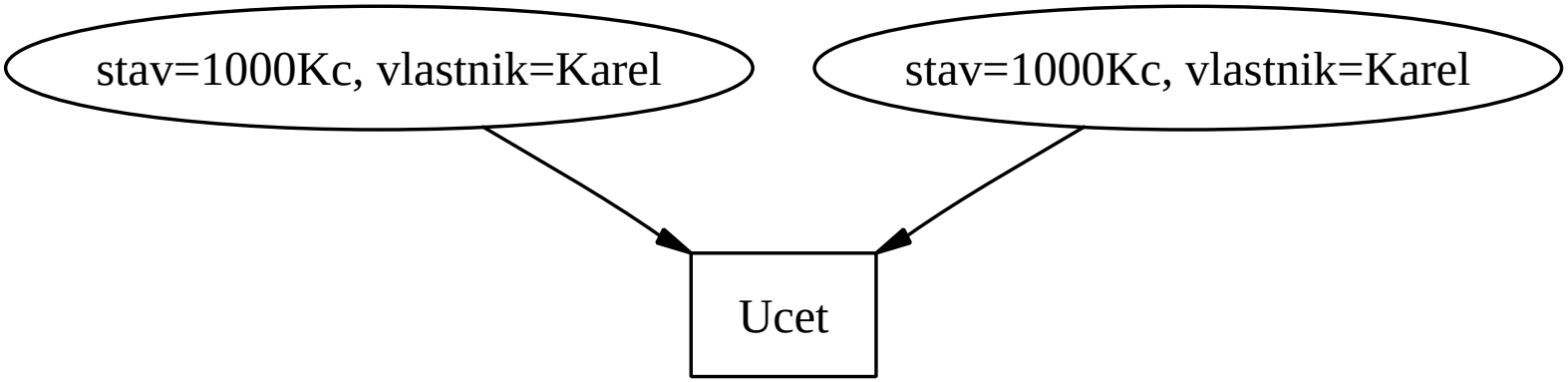
Třída

- je šablona, podle které se vytvářejí objekty (tj. instance tříd). Každý objekt má stejnou strukturu a chování jako třída, jejíž je instancí.
- je množina všech instancí stejného vzoru.



Identita objektu

- každý objekt je jedinečný
- objekty téže třídy jsou různé
- identita je vlastnost, podle které lze každý objekt identifikovat bez ohledu na jeho třídu nebo aktuální stav.
- většina OO jazyků vytváří jedinečné OID (např. adresa objektu)



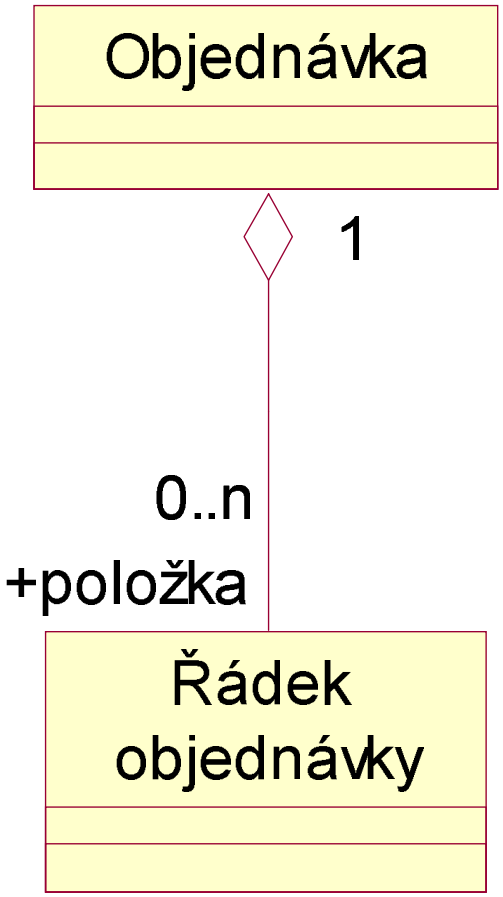
Polymorfismus (mnohotvarost)

- Logický vztah podobných operací (aplikace operací na podobné, ale technicky různé situace)
  - vícenásobná definice operace s jedním názvem, která (operace) tak může nabývat více implementací (implementuje různé chování)
  - atribut (proměnná) může odkazovat (obsahovat identifikátor) objekty různých tříd v různých okamžicích
- Časná vazba
  - implementace operace (metoda) je vybrána v době kompilace
- Pozdní vazba (dynamická vazba)
  - je technika dosažení polymorfismu
  - implementace operace (metoda) se vybere za běhu podle třídy objektu

Hierarchie

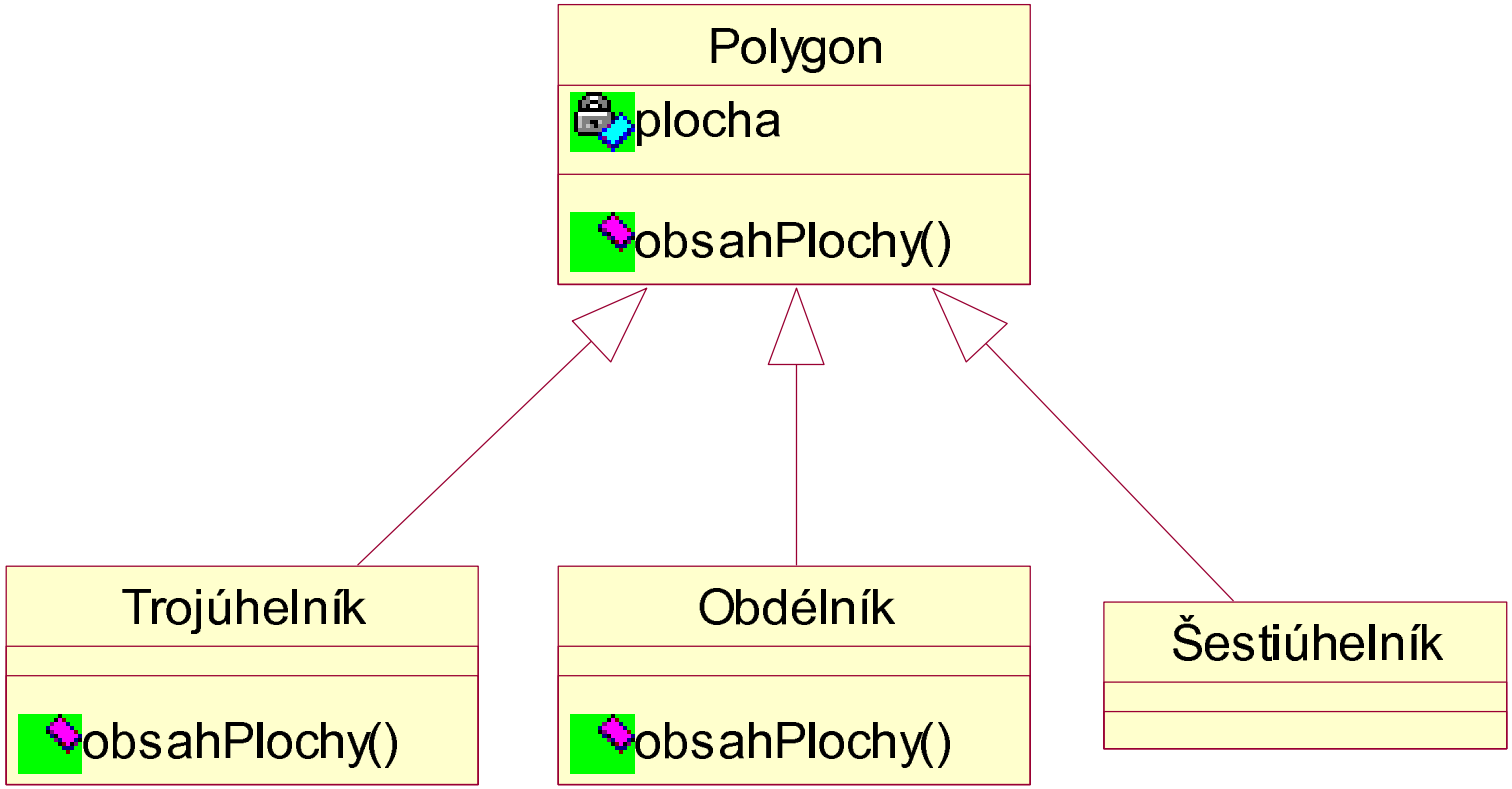
- Klasifikace pořadí abstrakcí
- Dědičnost (inheritance)
- Skládání
  - agregace (aggregation)
  - kompozice (composition)

Agregace



Dědičnost

- zobecňování (generalizace)
- specializace



Dědičnost

- přepisování (overriding) je změna definice metody zadané v třídě T v některé z podřízených tříd
- přetěžování (overloading) je technika vícenásobné definice operace v jedné třídě.

Přetěžování metod (Java):

```
prevedNa(Ucet u, int castka);
prevedNa(Ucet u);
```

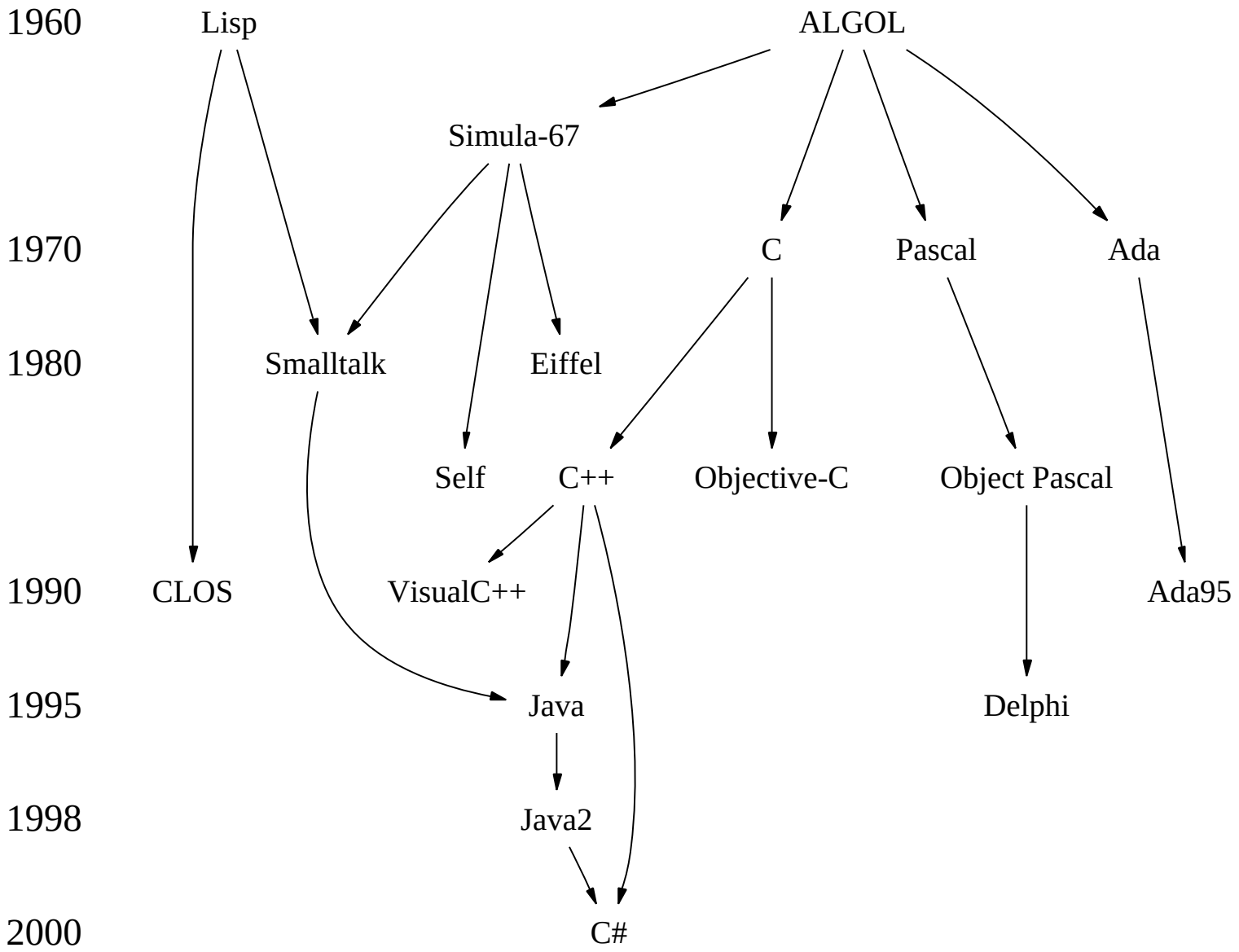
Smalltalk nezná přetěžování:

```
preved: castka na: u.
prevedNa: u.
```

Další vlastnosti OOP

- Typy
  - třída je chápána jako komplexní typ
  - statická kontrola typů
  - dynamická kontrola typů
- Souběžnost
  - objekty mohou konat ve stejném čase
  - procesy, vlákna
- Perzistence
  - Uložení stavu / dat během evoluce
  - Serializace

OOP – historie



Programovací jazyk Java

Základní charakteristika

- univerzální (není určen výhradně pro specifickou aplikační oblast)
- objektově-orientovaný
- statická typová kontrola
- jednodušší než C++ (méně syntaktických konstrukcí, méně nejednoznačností v návrhu)
- v průměru vyšší produktivita programátorské práce v Javě než v C++
- Java Virtual Machine – JVM (program v Javě je meziplatformně přenositelný na úrovni zdrojového i přeloženého kódu)
- automatické odklizení nepoužitelných objektů (automatic garbage collection)

Programovací jazyk Java

Základní charakteristika

- zdarma dostupné nezměrné množství knihoven pro různorodé aplikační oblasti, např. na SourceForge a tisících dalších místech
- k dispozici je řada kvalitních vývojových prostředí (i zdarma) - NetBeans, JBuilder, Visual Age for Java, Eclipse, IDEA
- reálným soupeřem je (Microsoft) C# (zatím převážně na platf. Windows)

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) - konkurence proprietárním ASP, SSI, CGI
- zpracování semistrukturovaných dat (XML)
- přenositelné aplikace s GUI
- aplikace distribuované po síti (applety nebo Java Web Start)

Typy aplikací

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

Java – platforma

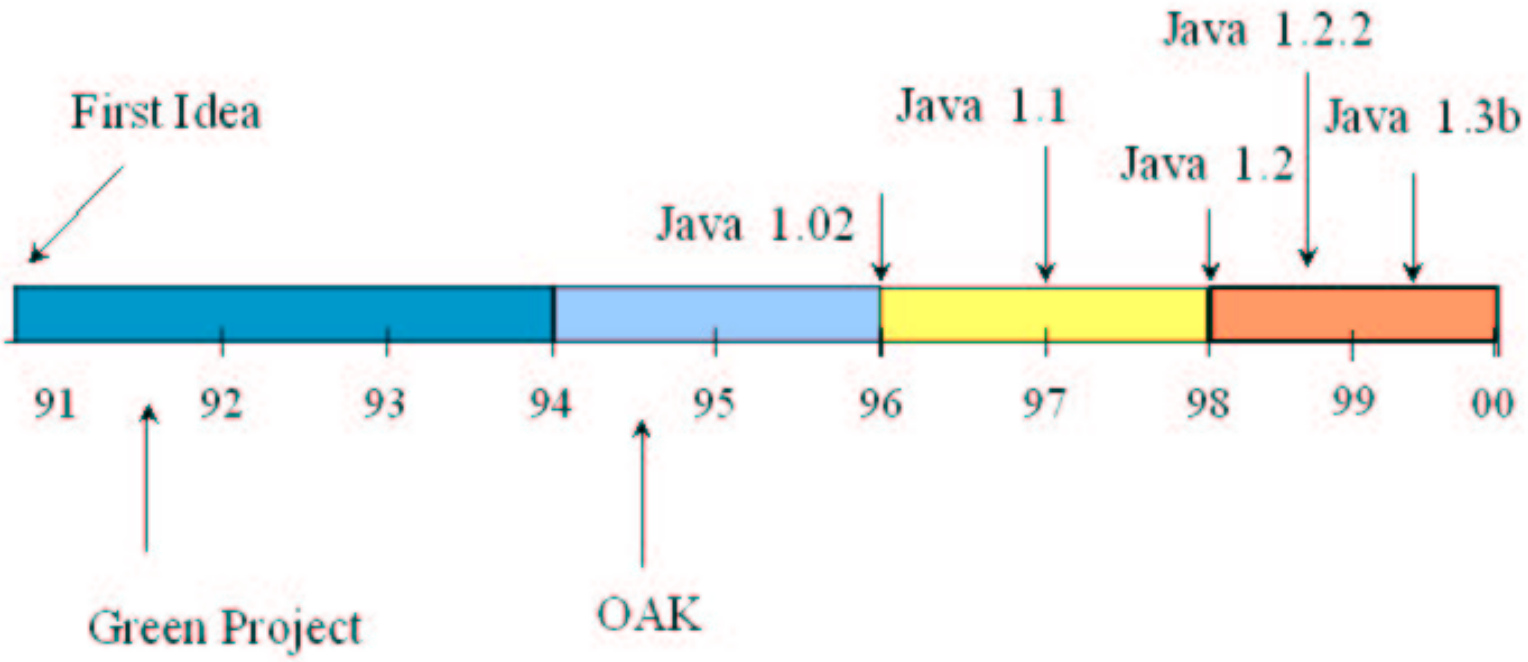
Java platformu tvoří:

- 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 – vývoj


First Idea
Java 1.02
Java 1.1
Java 1.2
Java 1.2.2
Java 1.3b
91
92
93
94
95
96
97
98
99
00
Green Project
OAK

Specifikace a implementace Javy

- Specifikace Javy
  - např. Java 2 Standard Edition, v1.4
  - např. Java 2 Enterprise Edition, v1.4

- Implementace Javy
  - např. Java 2 Software Development Kit, v1.4.2 - obsahuje vývojové nástroje
  - např. Java 2 Runtime Enviroment, v1.4 - obsahuje jen běhové prostředí pro spouštění hotových přeložených pg.

Verze Javy

Hrubé členění

- verze Java (před Java 2)
- verze Java 2

Číslování verzí:

- 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
- ke změně prvního čísla zatím nedošlo ... (?)

Aktuální verze

- Java 2 Standard Edition v1.5.0 (We have changed the version of this release from 1.5.0 to 5.0 to better reflect the level of maturity, stability, scalability and security built into J2SE.)
- aktuálně vždy na webu `http://java.sun.com`

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
J2SE 5.0 (1.5.0)	Tiger	Sept 29, 2004

Získání distribuce Javy

- používání Javy pro běžný vývoj (i komerční) je zdarma
- redistribuce javového vývojového prostředí je dovolena pouze s licencí od Sunu
- redistribuce javového běhového prostředí je možná zdarma
- distribuce vyvíjí Sun Microsystems Inc. (Javasoft) i další výrobci (např. IBM) a tvůrci Open Source

Stažení distribuce Sun

- `http://java.sun.com` (pro Windows, Solaris, Linux)
- dokumentace se stahuje z téhož místa, ale samostatně (nebo lze číst z WWW)
- celkově vývojové prostředí J2SDK 1.4.2 vč. dokumentace zabere cca 220 MB na disku
- velikost operační paměti - doporučeno 128 MB (a více :-))

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

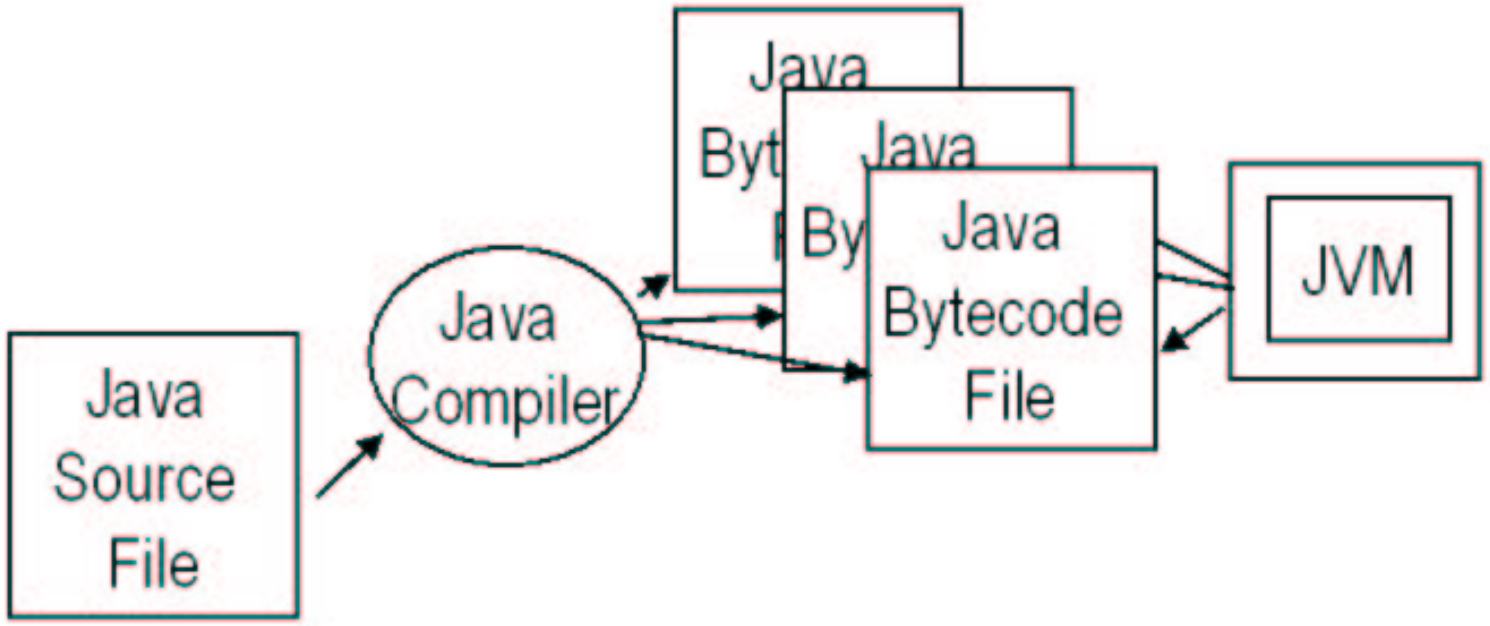
Základní životní cyklus javového programu

- Program sestává z jedné (ale obvykle více) tříd (class)
- Zdrojový kód každé veřejně přístupné třídy je umístěn v jednom souboru (`NazevTridy.java`)
- Postup:
  - vytvoření zdrojového textu (libovolným editorem čistého textu) $\Rightarrow$ `Pokus.java`
  - překlad (nástrojem javac) `Pokus.java` $\Rightarrow$ `Pokus.class`
  - spuštění, např. `java Pokus`

- překládá se `javac název_souboru_s_třídou` (včetně přípony .java!!!)
- spouští se vždy udáním `java název_třídy` (bez přípony .class!!!)

Java Virtual Machine

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



Struktura javového programu

- Každý netriviální javový program sestává z více tříd (class).
- Každá (veřejná) třída odpovídá jednomu souboru.
- Třídy jsou členěny do balíků (package).
- Zařazení do balíků znamená mj. umístění zdrojového souboru do příslušného adresáře!!!
- U běžné "desktopové" aplikace představuje vždy jedna (evt. více) třída vstupní bod do programu - je to třída/y obsahující metodu main.
- Java je *case sensitive*! (`ucet` x `Ucet`)

Komentáře

- Základní typy komentářů (podobně jako např. v C/C++)
  - *řádkové* od značky `//` do konce řádku
  - *blokové* (na libovolném počtu řádků) začínají `/*` pak je text komentáře, končí `*/`
  - *dokumentační* (na libovolném počti řádků) od značky `/**` po značku `*/` Každý další řádek může začínat mezerami či *, hvězdička se v komentáři neprojeví.

```
// řádkový komentář
/*
   blokový
   (víceřádkový) komentář
*/
/**
   dokumentační
   (víceřádkový) komentář
*/
```

Generování dokumentace

- Dokumentace má standardně podobu HTML stránek (s rámy i bez)
- Dokumentace je generována nástrojem javadoc
  1. z dokumentačních komentářů
  2. a ze samotného zdrojového textu
- Lze tedy (základním způsobem) dokumentovat i program bez vložených komentářů!
- Chování javadoc můžeme změnit volbami (options) při spuštění
- Dokumentační komentáře uvádíme:
  - Před hlavičkou třídy - pak komentuje třídu jako celek.
  - Před hlavičkou metody nebo proměnné - pak komentuje příslušnou metodu nebo proměnnou.

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)

Ukázka aplikace

Soubor `Pozdrav.java` je umístěn v balíku `IJA.seminar1` (tj. v adresáři `IJA/seminar1`)

```
package IJA.seminar1;
public class Pozdrav {
    // Program spouštíme aktivací funkce "main"
    public static void main(String[] args) {
        System.out.println("Ahoj!");
    }
}
```

Nezbytná pomůcka při programování v Javě:
`http://java.sun.com/reference/api/index.html`

Překlad

1. Máme nainstalován J2SDK 1.4.2
2. Jsme v adresáři `$HOME`, v něm je podadresář `IJA/seminar1`, v něm je soubor `Pozdrav.java`
3. Spustíme překlad
   `javac IJA/seminar1/Pozdrav.java`
4. Je-li program správně napsán, přeloží se "mlčky"
5. Výsledný .class (`Pozdrav.class`) soubor bude v témže adresáři jako zdroj

Spuštění

1. Poté spustíme program Pozdrav:
   `java -classpath .  IJA.seminar1.Pozdrav`
2. Volba překladače `-classpath directory`
   - zajistí, že (dříve přeložené) třídy používané při spuštění této třídy budou přístupné pod adresářem `directory`.
   - `-classpath .` tedy značí, že třídy (soubory .class) se budou hledat v odpovídajících podadresářích aktuálního adresáře (adresáře .)
3. Je-li program správně napsán a přeložen, vypíše se Ahoj!

Volba classpath

- definuje adresáře tvořící "kořenový" adresář pro hledání balíků a tříd

```
$HOME
  |-- java
        |-- distribution
        |-- project
        |-- docs
  |-- sun
        |-- distribution
        |-- examples
        |-- docs
```

Kořenový adresář:
`$HOME/java/project`
`$HOME/sun/examples`

```
export CLASSPATH="$CLASSPATH:$HOME/java/project:..."

java -classpath "$HOME/java/project:..."
```

Co znamená spustit program?

Spuštění javového programu = spuštění metody main jedné ze tříd tvořících program

Aplikace může mít parametry:

- podobně jako např. v Pascalu nebo v C
- jsou typu String (řetězec)
- předávají se při spuštění z příkazového řádku do pole String[] args (argument metody main)

Metoda main

- nevrací žádnou hodnotu – návratový typ je vždy(!) void
- její hlavička musí vypadat vždy přesně tak, jako ve výše uvedeném příkladu, jinak nebude spuštěna!

Praktické informace

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/j2sdk1.4.2`
- `CLASSPATH` = cesty ke třídám (podobně jako v `PATH` jsou cesty ke spustitelným souborům), např. `CLASSPATH=$HOME/java`

Distribuce Javy na FIT

- `sun00.fit.vutbr.cz` – `sun11.fit.vutbr.cz`
  - J2SE 1.4.2

- `merlin.fit.vutbr.cz`
  - J2SE 5.0 (1.5.0)

Ukázkový příklad – I

Adresář `$HOME`:

```
java
  |--- IJA
         |--- seminar1
                  |--- Pozdrav.java
```

Soubor `Pozdrav.java`

```
package IJA.seminar1;
public class Pozdrav {
    // Program spouštíme aktivací funkce "main"
    public static void main(String[] args) {
        System.out.println("Ahoj!");
    }
}
```

Ukázkový příklad – II

Překlad

- `cd $HOME/java`
- `javac IJA/seminar1/Pozdrav.java`

Spuštění

- `java -classpath .  IJA.seminar1.Pozdrav`

Spuštění

- `cd $HOME`
- `java -classpath $HOME/java IJA.seminar1.Pozdrav`

Spuštění

- `export CLASSPATH="$CLASSPATH:$HOME/java"`
- `java IJA.seminar1.Pozdrav`