Seminář Java

I

Radek Kočí

Fakulta informačních technologií VUT

7. ledna 2007

Téma přednášky

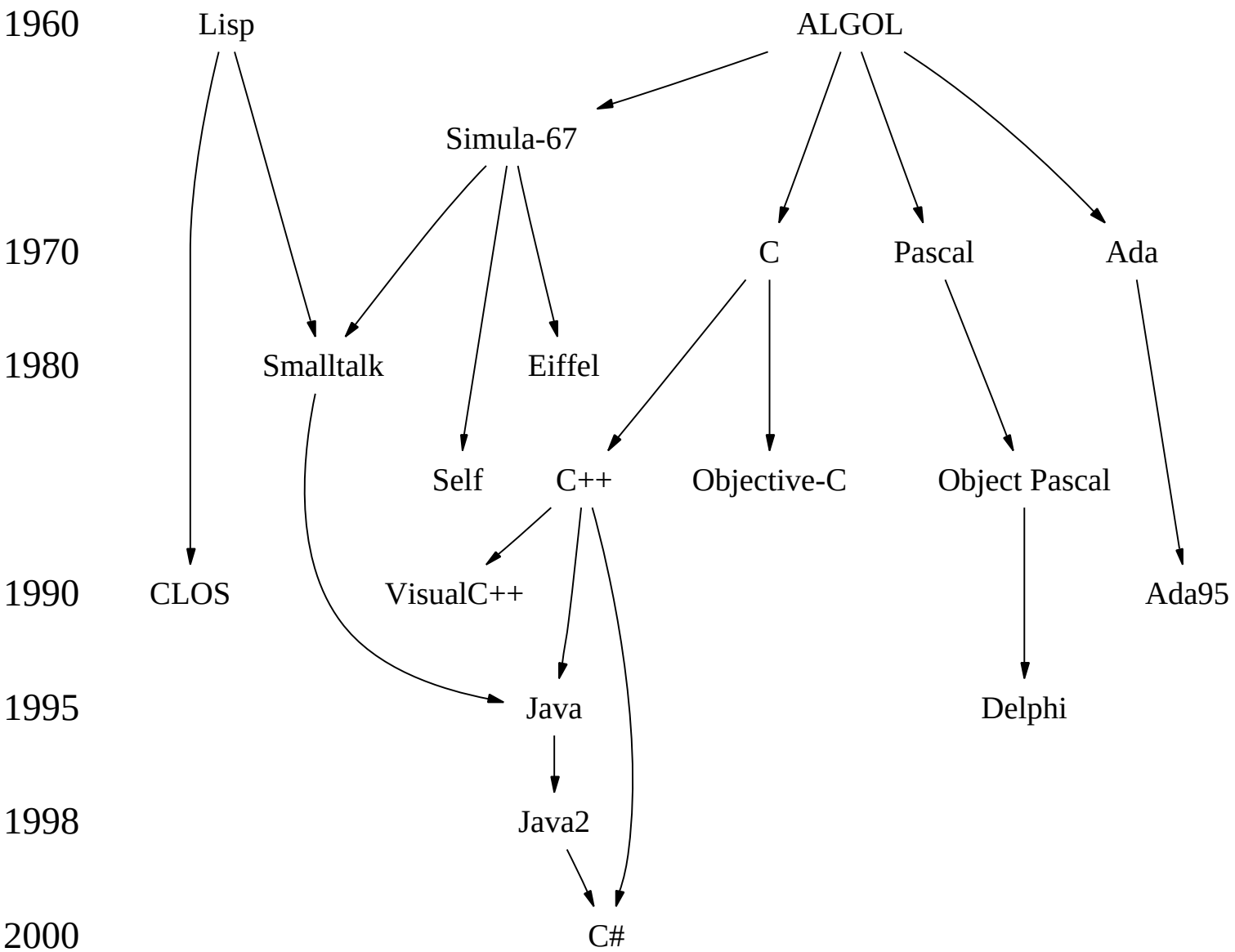
- Organizace semináře
- Úvod do programovacího jazyka Java
- Distribuce
- Struktura aplikace v Javě

Informace, studijní materiály

Stránky předmětu

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

Přehled jazyků



Programovací jazyk Java

Základní charakteristika

- 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)

Programovací jazyk Java

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

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

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
- J2SE $\Rightarrow$ Java SE

Verze Javy

Aktuální verze

- J2SE 5.0 (1.5.0)
- Java SE 6
- 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

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	2008

Java Technology History

Distribuce Javy

Podmínky získání a používání

- 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)

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

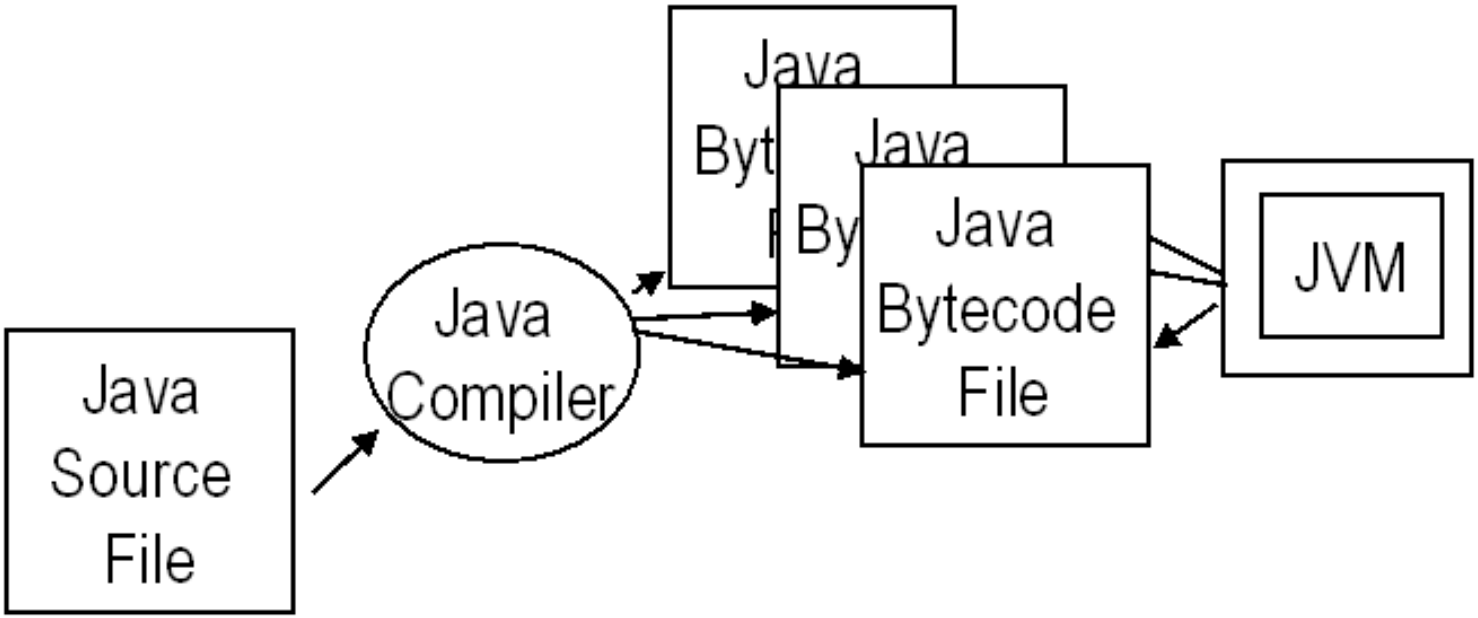
Nástroje ve vývojové distribuci

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)



Distribuce Javy na FIT

`merlin.fit.vutbr.cz`

- J2SE 5.0 (1.5.0)
- referenční instalace pro IJA

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`

Objektová orientace

Objekty a třídy

- abstrakce řešené domény $\Rightarrow$ objekty
- abstrakce na základě vyhledávání podobnosti $\Rightarrow$ klasifikace objektů do tříd
- aplikace je chápána jako kolekce vzájemně komunikujících objektů
- objekt $=$ sloučení dat a funkcionality do uzavřené jednotky

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 objektu a třídy

Vlastnosti objektů je třeba deklarovat

- proměnné
  - jsou nositeli "pasivních" vlastností, charakteristik objektů
  - datové hodnoty svázané (zapouzdřené) v objektu
- metody
  - jsou nositeli "výkonných" vlastností, "dovedností" objektů
  - v podstatě funkce (procedury) pracující primárně s proměnnými "mateřského" objektu
  - může mít další parametry (argumenty metody)
  - může vracet hodnotu

Programování v Javě spočívá ve vytváření tříd, neexistují metody a atributy deklarované mimo třídy.

Základní životní cyklus programu v Javě

Struktura aplikace v Javě

- aplikace sestává z alespoň jedné třídy
- zdrojový kód každé veřejně přístupné třídy je umístěn ve zvláštním souboru
  - `NazevTridy.java` (povinná přípona!)
- každá přeložená třída má svůj soubor s bytecode
  - `NazevTridy.class` (povinná přípona!)
- třídy jsou organizovány do balíků (packages)
- u běžné "desktopové" aplikace představuje vstupní bod do programu třída obsahující metodu `main`

- Java je *case sensitive*! (`ucet` x `Ucet`)
- API: `http://java.sun.com/reference/api/index.html`

Základní životní cyklus programu v Javě

Ukázka aplikace

- třída Pozdrav
- je umístěná v souboru Pozdrav.java
- je umístěna v balíku ija1

```
package ija1;
public class Pozdrav {
  // Program spouštíme aktivací funkce "main"
  public static void main(String[] args) {
    System.out.println("Ahoj!");
  }
}
```

Základní životní cyklus programu v Javě

Vytvoření zdrojového textu

- libovolný editor ⇒ `Pokus.java`

Překlad

- `javac Pokus.java`
- název souboru se udává včetně přípony `.java`
- vznikne soubor `Pokus.class`

Spuštění

- `java Pokus`
- udává se název třídy (tj. bez přípony `.class`)

Co znamená spustit program?

Spuštění javového programu odpovídá spuštění metody *main* jedné ze tříd tvořících program

Aplikace může mít parametry:

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

Metoda `public static void main(String[] args)`

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

Organizace tříd do balíků

- třídy jsou členěny do balíků (package)
- balíky vytvářejí stromovou strukturu
- organizaci balíků odpovídá organizace adresářů a umístění zdrojového souboru do příslušného adresáře
- může existovat více stromů

- Třída je plně kvalifikovaná svým názvem a balíkem!
- `ija1.ucty.Ucet`

Organizace tříd do balíků

```
package ija1;
```

```
$HOME
  |-- IJA
        |-- ija1
              |-- Pozdrav.java
              |-- Pozdrav.class
```

Překlad

1. jsme v adresáři `$HOME/IJA`
2. spustíme překlad `javac ija1/Pozdrav.java`

Organizace tříd do balíků

```
package ija1;
```

```
$HOME
  |-- IJA
        |-- ija1
              |-- Pozdrav.java
              |-- Pozdrav.class
```

Překlad

1. jsme v adresáři `$HOME/IJA`
2. spustíme překlad `javac ija1/Pozdrav.java`

Organizace tříd do balíků

```
$HOME
    |-- java
          |-- distribution
          |-- project
                |-- ija1
                |-- ija2
          |-- docs
    |-- sun
          |-- distribution
          |-- examples
                |-- ija3
          |-- docs
```

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

Organizace tříd do balíků

Nastavení cest pro balíky

- balíky (kořeny stromů) mohou být umístěny v různých adresářích
- je možné nastavit cesty do těchto adresářů
- v těchto adresářích se pak hledají balíky a třídy (.class)

Systémová proměnná `CLASSPATH`

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

Parametr `-classpath`

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

Ukázka aplikace

- Třída Pozdrav je umístěna v balíku ija1.
- Soubor `Pozdrav.java`:

```
package ija1;
public class Pozdrav { ...  }
```

```
$HOME
   |-- IJA
         |-- ija1
               |-- Pozdrav.java
```

Překlad: `javac -classpath "$HOME/IJA" $HOME/IJA/ija1/Pozdrav.java`

Spuštění:

```
java -classpath "$HOME/IJA" ija1.Pozdrav
```

Ukázka deklarace třídy

```
package ija1.ucty;

public class Ucet {
    protected String majitel;
    protected double zustatek;

    public void pridej(double castka) {
        zustatek += castka;
    }
    public void uber(double castka) {
        zustatek -= castka;
    }
    public void vypisZustatek() {
        System.out.println(zustatek);
    }
}
```

Datové typy

Java striktně rozlišuje mezi hodnotami

- primitivních datových typů
  - čísla
  - logické hodnoty
  - znaky
- objektových typů
  - řetězce
  - uživatelem definované typy – třídy a rozhraní

Základní rozdíl je v práci s proměnnými:

- proměnné primitivních datových typů přímo obsahují danou hodnotu
- proměnné objektových typů obsahují pouze odkaz na příslušný objekt

Primitivní datové typy

Charakteristika

- Proměnné těchto typů nesou atomické, dále nestrukturované hodnoty
- Deklarace způsobí
  - rezervování příslušného paměťového prostoru
  - zpřístupnění (pojmenování) tohoto prostoru identifikátorem proměnné

Primitivní datové typy

Typ `boolean`

- logická hodnota, přípustné hodnoty jsou `false` a `true`
- na rozdíl od Pascalu na nich není definováno uspořádání

Typ `void`

- není v pravém slova smyslu datovým typem, nemá žádné hodnoty
- označuje "prázdný" typ pro sdělení, že určitá metoda nevrací žádný výsledek

Primitivní datové typy

Čísla s pohyblivou řádovou čárkou

- `float`
  - 32 bitů
- `double`
  - 64 bitů
- zápis literálů
  - `float f = -.777f, g = 0.123f, h = -4e6f, i = 1.2E-15f;`
  - `double f = -.777, g = 0.123, h = -4e6, i = 1.2E-15;`

Primitivní datové typy

Integrální typy – celočíselné

- v Javě jsou celá čísla vždy interpretována jako znaménková
- `int`
  - 32 bitů ($-2\,147\,483\,648$ .. $2\,147\,483\,647$)
  - základní celočíselný typ
- `long`
  - 64 bitů (cca $+/-\ 9 \cdot 10^{18}$)
- `short`
  - 16 bitů (-32768 .. 32767)
- `byte`
  - 8 bitů (-128 .. 127)

Primitivní datové typy

Integrální typy – `char`

- char představuje jeden 16bitový znak v kódování UNICODE
- konstanty typu char zapisujeme
  - v apostrofech: `'a'`, `'ř'`
  - pomocí escape-sekvencí: *n* (konec řádku) *t* (tabulátor)
  - hexadecimálně: *u*0040 (totéž, co `'a'`)
  - oktalově: \127
- *Pozor na kódové stránky při překladu/spouštění – dochází k překódování textu! (komentář, znak, řetězec, identifikátor)*

```
javac -encoding ISO8859-2 ...
```

Proměnné objektového typu

Charakteristika

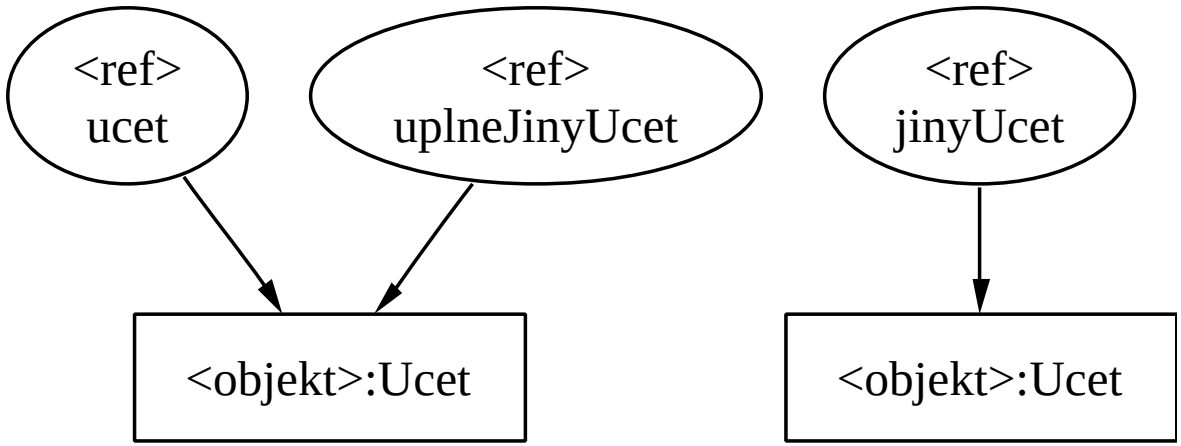
- Proměnné těchto typů reprezentují reference na objekty
- Deklarace způsobí
  - rezervování paměťového prostoru na *referenci*!
  - vlastní objekt (instance třídy) nevzniká!

Vytvoření instance

- operátor `new`
- rezervuje paměťový prostor pro objekt (instanci dané třídy)

Proměnné objektového typu

```
public class Banka {
    public static void main(String[] args) {
        Ucet ucet = new Ucet();
        Ucet jinyUcet = new Ucet();
        Ucet uplneJinyUcet = ucet;
    }
}
```



Volání metod

Nad existujícími (vytvořenými) objekty můžeme volat jejich metody

- samotnou deklarací (napsáním kódu) metody se žádný kód neprovede
- chceme-li vykonat kód metody, musíme ji zavolat.
- volání se realizuje "tečkovou notací"
- volání lze provést, jen je-li metoda z místa volání přístupná
- přístupnost regulují modifikátory přístupu

Volání metod

```
public class Banka {
    public static void main(String[] args) {
        Ucet ucet = new Ucet();
        ucet.vypisZustatek();
        ucet.pridej(100.50);
        ucet.vypisZustatek();
        ucet.uber(0.50);
        ucet.vypisZustatek();
    }
}
```

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.

Komentáře

```
// řádkový komentář

/*
   blokový
   (víceřádkový) komentář
*/

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

Generování dokumentace

Dokumentace

- je generována nástrojem `javadoc`
  - z dokumentačních komentářů
  - a ze samotného zdrojového textu
- *je tedy možné dokumentovat (základním způsobem) i program bez vložených komentářů!*
- má standardně podobu HTML stránek (s rámy i bez)
- chování `javadoc` můžeme změnit volbami (options) při spuštění

Dokumentační komentáře uvádíme:

- před hlavičkou třídy (komentuje třídu jako celek)
- před hlavičkou metody nebo proměnné (komentuje příslušnou metodu nebo proměnnou)

Značky pro dokumentační komentáře

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

@author	specifikuje autora API/programu
@version	označuje verzi API, např. "1.4.2"
@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 použití dokumentačních komentářů

```
package ija1.ucty;

/**
 * Trida ucet
 * @author R. Koci
**/
public class Ucet {
    /** Majitel uctu */
    protected String majitel;
    protected double zustatek;

    public void pridej(double castka) {
        zustatek += castka;
    }
    ...
}
```