

Java pro programátory .NET

Lukáš Zapletal

lukas.zapletal zavináč liberix.cz

Představení jazyka

Co je to Java?

[džáva]

Java je objektově orientovaný jazyk stvořený pro vestavěná zařízení, který se uchytil v mnoha oblastech od desktopy až po mobilní zařízení a Internet.

Největší úspěchy zažívá na serverech.

J2SE – standardní edice

J2EE – komponent. technologie pro obchod

J2ME - „ořezaná“ J2SE (mobily, TV, PDA)

Letem světem...

JVM = CLR

Bytecode = IL

Java, JPython... = C#, VB, Managed C++

JAR = DLL/EXE

JDK = .NET Framework SDK

javac = cs

ant = make

Servlet/JSP/JSF = ASP.NET

Eclipse, Netbeans = MS Visual Studio

(www.eclipse.org, www.netbeans.org)

Kde ji najít...

java.sun.com – vývojáři

java.com – uživatelé

JRE – nutné pro běh (14MB)

JDK – překladač, nástroje (50MB)

JDK + Netbeans bundle (100MB)

dokumentace – zvlášť (50MB)

(rozbalit do ./doc)

Jak se ji naučit...

- B. Eckels: Thinking In Java (www, vyšla v překladu – Grada 2003)
čeští autoři – M. Virius, P. Herout
- J. Bloch – Effective Java (překlad, Grada)

další zdroje:

- jakarta.apache.org - Ant
- www.onjava.com - články
- www.theserverside.com - J2EE

Co je stejné...

- ◆ jednoduchá dědičnost (jeden rodič)
- ◆ operátor new, sběr smetí
- ◆ přístup ke členům pomocí tečky
- ◆ abstraktní metody a jejich princip
- ◆ rozhraní, dědičnost rozhraní
- ◆ konstruktory, implicitní konstruktory

Co je jiné: způsob zápisu

balíčky: `cz.upol.zapletal`
třídy: `Trida`, `MojeTrida`, `SuperDlouhyNazev`
metody: `init`, `doUpdate`, `getSize`, `setSize`
členské proměnné: stejné jako metody
konstanty: `K1`, `KONSTANTA_JEDNA`

Co je jiné: základy

- ◆ JAVA
 - ◆ `java.lang.Object`
 - ◆ balíčky
 - ◆ výchozí: `package`
 - ◆ metody
 - automaticky
 - virtuální
 - ◆ nemá destruktork (pouze finalizaci)
 - ◆ anonymní třídy
- ◆ C#
 - ◆ `System.Object`
 - ◆ namespaces
 - ◆ výchozí: `private`
 - ◆ nutno použít
 - `virtual/override`
 - ◆ konstruktory jako v C++
 - ◆ pouze anonymní metody

Co je jiné: data a operátory

- ◆ nemá unsigned
- ◆ nemá ukazatele (záměr)
- ◆ boxing/unboxing až od verze 5
- ◆ enum až od 5
- ◆ struct = class
- ◆ operátory stejné
- ◆ instance of
- ◆ final
- ◆ unsigned int/short...
- ◆ základní podpora pro ukazatele
- ◆ má (un)boxing
Integer a = 5;
- ◆ má enum
- ◆ má struct
- ◆ operátory stejné
- ◆ is
- ◆ sealed

Co je jiné: řízení toku

- ◆ if, while, do, for, break, continue
- ◆ switch – jen int
- ◆ nemá goto (záměr)
- ◆ foreach až od verze jazyka 5:
for (Object c: col)
- ◆ nemá přetěžování operátorů (záměr)
- ◆ úplně stejné jako v Javě
- ◆ switch – i řetězce
- ◆ goto lze použít
- ◆ foreach (object o in collection)
- ◆ C# umí přetěžovat některé operátory

Co je jiné: výjimky

- ◆ `java.lang.Throwable`
- ◆ dvě úrovně:
`Error`, `Exception`
- ◆ rozlišuje mezi povinnými a nepovinnými výjimkami
- ◆ `throws` v metodách
- ◆ těžko se dá programovat špatně
- ◆ zásadně se netestuje výsledek op. na `null`
- ◆ `System.Exception`
- ◆ má pouze výjimky
- ◆ všechny výjimky jsou nepovinné (jejich odchyt není nutný)
- ◆ v `.NETu` lze snadno programovat „nebezpečně“ - tedy nezachytávat výjimky
- ◆ testování na `null` (někdy)

Co je jiné: Object

- ◆ toString()
 - ◆ equals(Object o)
 - ◆ hashCode()
 - ◆ getClass()
 - ◆ finalize()
 - ◆ clone()
- ◆ ToString()
 - ◆ Equals(Object o)
 - ◆ GetHashCode()
 - ◆ GetClass()
 - ◆ Finalize()
 - ◆ MemberWiseClone()

Co je jiné: kolekce

- ◆ bohatší
- ◆ dvě verze
- ◆ má vláknově-zabezpečené i nezabezpečené
- ◆ množina (Set)
- ◆ `Collections.sort(...)`
- ◆ nemá [...]
- ◆ problém přetypování (Commons Primitives)
- ◆ funkčně chudší
- ◆ pouze jedna verze, dobré pojmenování
- ◆ pouze vláknově nezabezpečené
- ◆ nemá množiny
- ◆ `kolekce.sort()`
- ◆ možno použít [...]
- ◆ problém přetypování (boxing/unboxing)

Co je jiné: I/O, vlákna

- ◆ dobrá knihovna proudů + NIO
 - ◆ serializace (Serializable)
 - ◆ Reader/Writer
 - ◆ java.lang.Thread
 - ◆ ThreadPool jako ext. knihovna
- ◆ dobrá knihovna proudů
 - ◆ serializace (Serializable)
 - ◆ Reader/Writer
 - ◆ System.Threading
 - ◆ ThreadPool přímo v knihovně

Co je jiné: vlastnosti

◆ get/set

```
int getSize() {  
    return size;  
}  
  
void setSize(int s) {  
    size = s;  
}
```

◆ properties:

```
int Size {  
    get {  
        return size;  
    }  
    set {  
        size = value;  
    }  
}
```

Co je jiné: řetězce

- ◆ `java.lang.String`
- ◆ `String a = „xxx“;`
- ◆ `a.equals(„b“)` nebo `„b“.equals(a)`
- ◆ těžší zápis, výhody toho, že `String` má své místo v hierarchii
- ◆ spojování pomocí `+`
- ◆ `StringBuffer`
- ◆ regulární výrazy
- ◆ `java.text.Format`
- ◆ `printf` až od verze 5
- ◆ prim. typ: `string`
- ◆ `string a = „xxx“;`
- ◆ `a == b` nebo `b == a`
- ◆ snadnější zápis, `string` stojí mimo API – nutno s ním nakládat zvlášť
- ◆ spojování pomocí `+`
- ◆ `StringBuilder`
- ◆ regulární výrazy
- ◆ `IFormatProvider`
- ◆ `printf` metody

Co je jiné: modifikátory

- ◆ public/private stejné
- ◆ protected to samé
- ◆ package – pouze z balíčku
- ◆ public/private
- ◆ protected
- ◆ internal – pouze assembly
- ◆ nutno používat kl. slovo override při změně

Co je jiné: dědičnost

- ◆ `class c extends p {`
- ◆ `class c implements`
`int1, int1... {`
- ◆ `c instanceof int1`
- ◆ `class c : p {`
- ◆ `class c : int1, int2 {`
`}`
- ◆ `c is int1`

Co je jiné: primitivní typy

- ◆ byte/int/short/long
- ◆ nemá unsigned
- ◆ java.lang.Byte
 - Short
 - Integer
 - Character
- ◆ třída BigDecimal (pro finanční data)
- ◆ nemá ukazatele
- ◆ stejné
- ◆ navíc ubyte/uint...
- ◆ System.Byte
 - Int16
 - Int32
 - Char
- ◆ prim. typ decimal
- ◆ ukazatele:
 - > * &

Co je jiné: pole

- ◆ `int[][] p = new [3][2];`
 - ◆ neobdélníková pole stejné jako v .NET
 - ◆ `System.arraycopy()`
 - ◆ pole lze procházet pomocí `for` (od verze 5.0)
- ◆ `int[,] p = new [3,2];`
 - ◆ neobdélníková pole (jagged arrays)
 - ◆ `Array.Copy()`
 - ◆ pole lze procházet pomocí `foreach`

Co je jiné: kolekce

- ◆ Collection
- ◆ List
- ◆ Comparator
- ◆ Set
- ◆ Enumeration
- ◆ nemá
- ◆ ArrayList
- ◆ HashMap
- ◆ TreeMap
- ◆ LinkedList
- ◆ ICollection
- ◆ IList
- ◆ IComparer
- ◆ nemá
- ◆ IEnumerable
- ◆ IHashCodeProvider
- ◆ ArrayList
- ◆ Hashtable (pozor)
- ◆ SortedList
- ◆ Queue

Co je jiné: delegáti

◆ Java nemá delegáty:

```
this.addListener(  
    new MouseListener() {  
  
        void onClick(Event e) {  
            // dělej něco  
        }  
  
    });
```

◆ delegate:

```
delegate MouseHandler  
    xy;  
  
this.OnClick +=  
    new MouseHandler(  
        MyMouseHandler);  
  
public MyMouseHandler...  
    {}
```

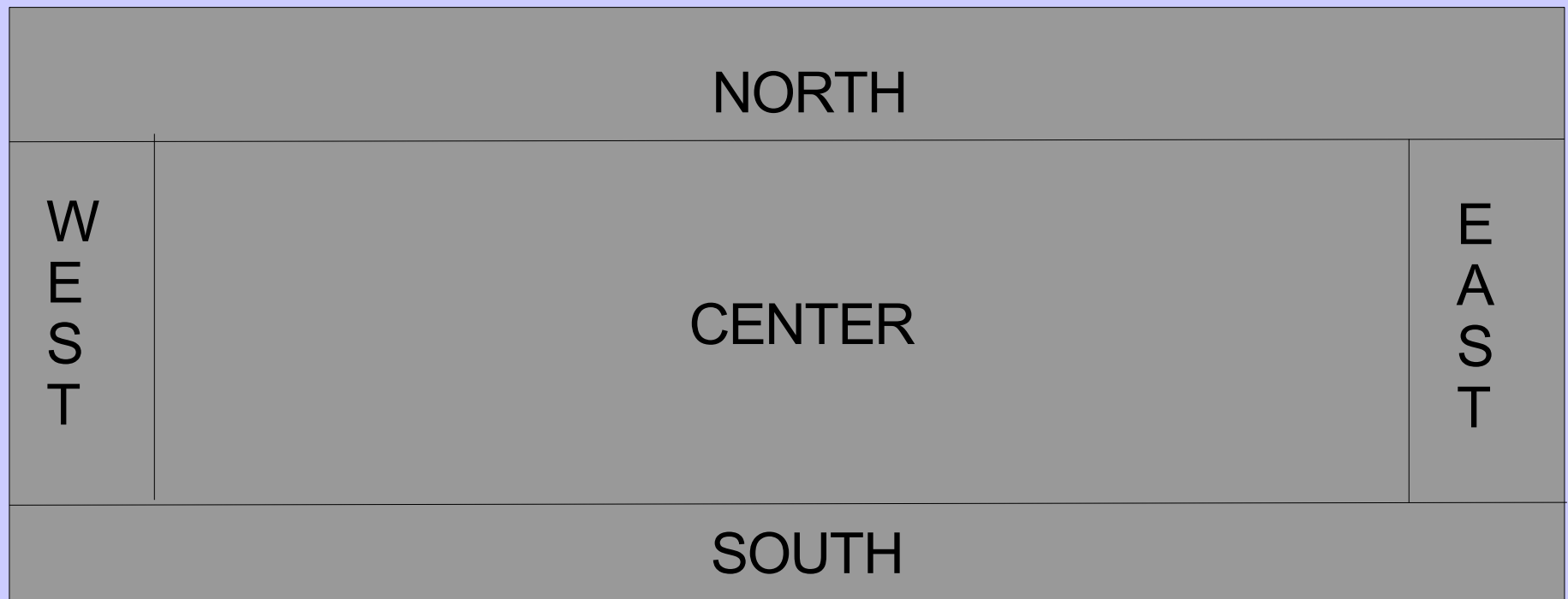
Tvorba GUI: okna a okna

- první verze Javy = AWT
 - od verze 1.2 je to SWING (JFC)
- Swing je rozšířením AWT (Frame x JFrame)
- přenositelná knihovna s modulární arch. a s podporou skinů (look-and-feel)
 - návrhové vzory skládání a dekorátor
- existují i jiné (SWT, QT, Gnome-Java...)
 - okno = frame (JFrame)
- `static void main(String[] args) { ... }`

Tvorba GUI: Layout Managers

- ◆ JFrame = okno
- ◆ obsah = contentPane + menuBar
- ◆ největší problém, v Javě se nepoužívají XY managery kvůli přenositelnosti
- ◆ vytvoří se panely (JPanel) a na ně se rozmístí komponenty, to se vrství
- ◆ panelům se přidělují layout managery: BorderLayout, GridLayout, FlowLayout, CardLayout a BoxLayout – pouze do řady nebo sloupce
- ◆ pro GUI buildery: GridbagLayout, SpringLayout
- ◆ např. Netbeans využívá GBLayout + přidává (nestandardní) XYLayout
- ◆ profi programy se stejně takto píší...

Tvorba GUI: BorderLayout



Je implicitním managerem po vytvoření JPanelu, často se používá, jedno okno často obsahuje až několik (5) vnořených panelů.

Tvorba GUI: zprávy

- ◆ základem jsou rozhraní NěcoListener
- ◆ každá komponenta poskytuje metody `addNějakýListener(...)`
- ◆ časté využití anonymních tříd
- ◆ pro usnadnění je k dispozici k některým rozsáhlým rozhraním také abstraktní implementace (třídy) `AbstractNěcoListener`
- ◆ při reakcích je nutno „jednat rychle“ nebo spustit nové vlákno (aby aplikace nepřestala reagovat)

Tvorba GUI: Applety

- ◆ Applety jsou komponenty spouštěné v prohlížeči, proto mají omezená práva
- ◆ umožnění některých věcí (čtení/zápis na disk, k síti) = dig. podepisování
- ◆ v SWT = Applet (Java 1.1, MSIE)
- ◆ od verze 1.2 = JApplet
- ◆ nutno mít nainstalován plug-in (instaluje se automaticky s JRE)
- ◆ chová se podobně jako JFrame, ale má jinou inicializaci (init, destroy)

Balíčky a adresáře

- ◆ každá třída (soubor class) musí být ve stejném adresáři jako balíčku
- ◆ např. `cz.upol.zapletal.Curve` musí být v: `/někde/cz/upol/zapletal/Curve.class`
- ◆ ve stejné hierarchii jsou soubory `.java`
- ◆ překladač i interpret si to vynutí
- ◆ vše je průhledné, velmi snadné to ovšem mají crackeři... proto obfuscating

Překlad a spouštění aplikací

- ◆ CLASSPATH = nejdůležitější prvek
- ◆ CP určuje cestu k adresářům se soubory class nebo k souborům JAR
- ◆ java = interpret (dnes již JIT)
- ◆ java -cp .:bin:lib/knihovna.jar cz.upol.Test
- ◆ na Windows středníky, jinde dvojtečky
- ◆ javac = překladač (napsán v Javě)
- ◆ javac -cp lib/knihovna.jar *.java
- ◆ moc možností nemá, časté využití nástroje Ant nebo prostředí IDE (ty Ant používají)

Spouštění appletů

- ◆ applety se musejí zkompilovat a eventuálně zabalit do JAR archivu
- ◆ v HTML se použije `<applet>` nebo `<object>`
- ◆ ladění pomocí aplikace `appletviewer` je těžkopádné
- ◆ doporučení: implementovat metodu `main` a ladit applet jako obyčejnou aplikaci
- ◆ pozor však pak na omezená práva (nepodepsaný applet toho moc nemůže...)
- ◆ v prohlížeči – Java Concole (trayikona)

d :=]

gl