Java

[džava] nebo [džáva] – v americkém slangu "kafe"
Sun Microsystems – nyní Oracle

Významné zdroje

– ze stovek/tisíců dostupných

`http://java.sun.com`	Java
`http://dione.zcu.cz/java`	výborný rozcestník

Historie

– původně Oak (1990) – programovací jazyk Sun pro spotřební elektroniku (embeded software) – čipy se často mění, je jich mnoho druhů, software musí být spolehlivý, pokud možno interpretovaný

– zlom 1993, kdy proráží WWW, na němž je potřeba jazyka nezávislého na architektuře – WWW prohlížeč HotJava, který umí applety („podprogramy pro HTML“) – statické HTML má dynamické možnosti, které nezáleží na architektuře

– Sun si uvědomuje „zbraň“ proti Microsoftu a nalézá nadšené spojence, dává zdarma základní vývojové nástroje (JDK) pro nejpoužívanější platformy

– v květnu roku 1995 byla Java firmou Sun oficiálně představena

– Netscape od 2.0 podporuje applety

– od 1996 prožívá Java neuvěřitelný rozvoj

– giganti IBM, SGI, Oracle a Microsoft (v pořadí nákupů) kupují licence na Javu

– všechny významné firmy s vývojovými prostředky intenzivně připravují vývojové prostředí

– je to nejen jazyk pro přímé použití na WWW, tj. aplety, ale naprosto plnohodnotný programovací jazyk pro samostatné aplikace

– napiš a přelož program jednou včetně GUI a spusť bez dalších překladů a akcí na libovolné platformě (procesor + OS) – úděsná představa pro MS

Zkratky

API – Application Programming Interface – Java Core API
JDK – Java Development Kit, – SDK
JIT – Just-In-Time
JVM – Java Virtual Machine (interpreter Javy a prostředí run-time)
RAD – Rapid Application Development
GUI – Graphical User Interface
AWT – Abstract Windowing Toolkit
JFC – Java Foundation Clases
Swing – modernější knihovny pro GUI

Charakteristika jazyka dle Sun „The Java Language: A White Paper“ (viz dione)

Jednoduchý

- málo jazykových konstrukcí, ideální pro přechod z C
- základ je „vykuchané C a C++“, ze kterých vzali osvědčené věci a ty rozvinuli a ubrali problematické (např. ukazatele z C a vícenásobnou dědičnost z C++)

Objektově orientovaný

- OOP je jedním z osvědčených směrů v programování s mnoha výhodami, např. reusable – zvovuvyužitelnost SW, kdy Java má v systému standardních balíků „již vše hotovo“ a stačí to jen použít
- Java je objektová již od počátku – vyvarovala se problémů C++

Interpretovaný

- překlad je do bajtového kódu (byte code), nedochází k sestavování
- to umožňuje nezávislost na architektuře a současně i ochranu zdrojového kódu

Robustní

- původně navržena pro tvorbu vysoce spolehlivého SW
- silně typovaný
- neexistují ukazatele a o správu paměti se stará interpret sám (statistiky říkají, že toto zjednodušení vede až k 50% redukci výskytu chyb běžných při použití tradičních OOP jazyků)
- rozsáhlá kontrola chyb jak při kompilaci, tak i při vlastním běhu programu, např. mezí polí, prázdných odkazů

- systém zpracování výjimek, kdy je ošetření chyb v jednom místě programu
- systém balíků (package) podporuje velké projekty
- knihovny a aplikace v JAR

Přizpůsobený národním zvláštnostem
- abecedy neanglických jazyků jsou podporovány pomocí znakové sady Unicode
- lokalizace součástí knihoven

Distribuovaný
- všemožně podporuje aplikace v sítích
- vestavěná podpora protokolu TCP/IP

Bezpečný
- filosofie interpretu umožňuje provést před spuštěním množství kontrol a zabraňuje znalosti „paměťové mapy“, použitelné jako vstupní brána

Vysoce výkonný
- v počátcích (1996) byla interpretovaná Java z JDK 20krát pomalejší, než kompilované C
- JIT kompilátory – v době zavádění přeloží bajtový kód do strojového kódu platformy (v návrhu bajtového kódu s tím bylo počítáno, takže je to efektivní) dle Sun se pak rychlost vyrovná C++
- Hot-spot – optimalizace za běhu
- Java kompilátory – překlad bajtového kódu do „.EXE“ souboru – ztráta přenositelnosti, ale nepotřebuje interpret

Přenosný
- explicitně určuje velikost všech primitivních datových typů
- hardwarové rozdíly zastřešuje tzv. Java Platforma, která obsahuje dvě základní části:
  - Abstraktní počítač – JVM, který sestává z runtime systému, což je část realizující vazbu na hardware, a interpretu, který vykonává bytový kód
  - Aplikační programové rozhraní (Java Core API) - což jsou základní knihovny pro psaní programů. Výhodou je, že tyto knihovny nemusí být s programem distribuovány, neboť jsou součástí Java Platformy.

Víceprocesní

- vestavěnou jazykovou podporou plně podporuje multithreading (=multitasking) více vláken (light-weigth procesy), tj. paralelní běh částí programu a jejich synchronizaci

Dynamický

- lze využívat části, které jsou kdekoliv jinde a při jejich změně se s původním programem nemusí vůbec nic dělat
- tyto části jsou dynamicky linkovány za běhu programu

Dokumentovaný

- do praktické použitelnosti dotažena idea automatické generace dokumentace z kódu
- jen knihovny mají 220 MB dokumentace
- Java Tutorial

Podpora GUI od počátku

- knihovna Swing – GUI vypadá na všech platformách stejně
- tato výhoda se ukázala jako nevýhoda – Java neprorazila na desktopech

Evoluce Javy

- JDK 1.0.2 – původní verze Javy z 1995
- JDK 1.1.8 – proti JDK 1.0 došlo ke změnám jazyka a k podstatným změnám (zejména rozšíření) API.
  od této verze je stabilní jazyk (1997)
- JDK 1.2.2 – podstatnou změnou bylo zahrnutí některých nových knihoven (např. JFC Swing) jako součásti JDK.
- JDK 1.3.0 – změnou proti JDK 1.2. je zrychlení části API. Změny v jazyce nenastaly žádné, změny v API jsou pouze rozšíření a vylepšení
- JDK 1.4.0 – výrazné zrychlení, v API jsou pouze rozšíření a vylepšení
- JDK 1.5.0 – značné doplnění jazyka (C#) o generické typy (Java2, Tiger)
- JDK 1.6.18 – současná verze (Java 6)

Existují tři distribuční verze
SE – Standard Edition – běžné použití
ME – Micro Edition – použití v malých zařízeních (úsporná ořezaná verze)
EE – Enterprise Edition – použití pro velké informační systémy (rozšířená verze)

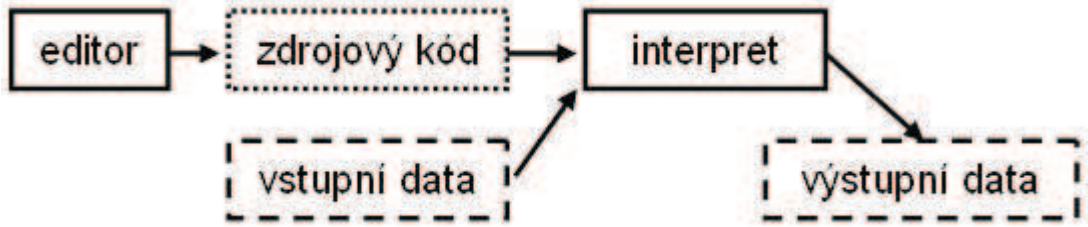
Jak javu nainstalovat

Ukázka práce v moderním RAD

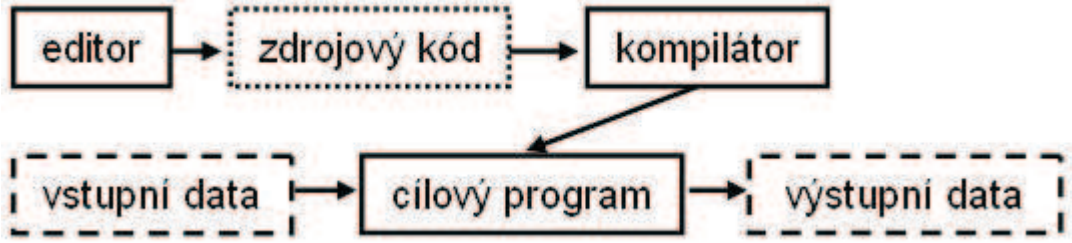
1.4.2. Implementace programovacích jazyků

Dvě základní metody:

- **interpretační**
  - větší přenositelnost programů, ale menší rychlost, syntaktické chyby odhaleny až při spuštění



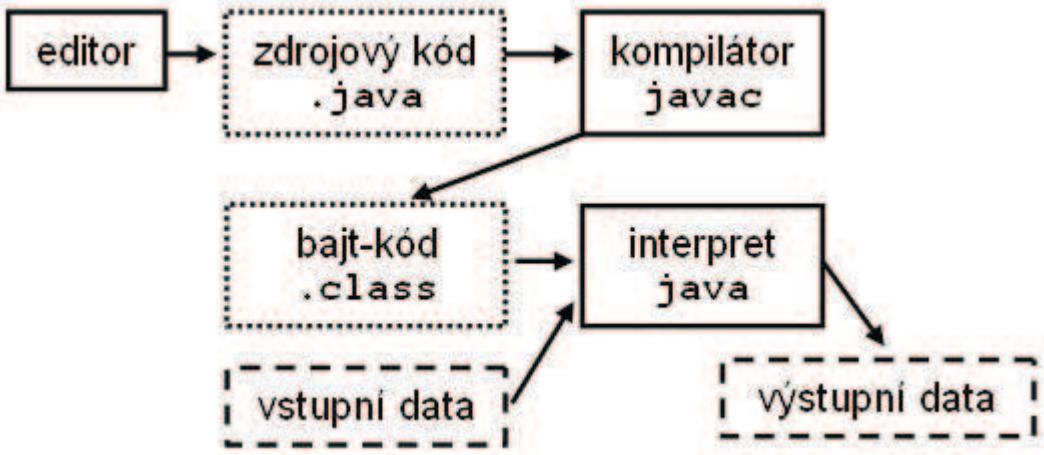
- **kompilační**
  - nulová přenositelnost mezi **platformami**, větší rychlost, syntaktické chyby odhaleny při překladu (platforma = procesor + operační systém)



- obě metody se dají kombinovat

1.5. Úvod do jazyka Java

- jazyk Java je příkladem kombinace kompilační a interpretační metody



  - program je tvořen jedním nebo několika zdrojovými soubory s příponou `.java`
  - zdrojové soubory se přeloží překladačem `javac` (v terminologii firmy Sun je to kompilátor) do vnitřní formy (*byte code*, **bajt-kód**), která je platformově nezávislá
    - překladem souboru `Jmeno.java` vznikne soubor s názvem `Jmeno.class`

- interpretaci vnitřní formy provede program `java` (JVM – *Java Virtual Machine*)
- program obvykle využívá řadu knihoven (*Java Core API – Application Programming Interface*), které je třeba mít k dispozici jak při překladu, tak při interpretaci
  - ♦ JVM + Java Core API = Java platforma
- rychlost interpretované Javy a kompilovaného jazyka (např. C) je srovnatelná
  - ♦ technologie JIT (*Just In Time*) a Hot-Spot

1.5.1. Vývojové prostředky

doporučené (tj. nainstalované v počítačové učebně) jsou dva prostředky

oba jsou volně šiřitelné a jsou dostupné na poskytnutém CD

1. **JDK, SciTe, příkazová řádka** – budeme používat při několika prvních cvičeních
   - JDK (*Java Development Kit*) – základní programy poskytované firmou Sun

     k dispozici na `http://java.sun.com`
   - SciTe – jednoduchý editor se základní podporou Javy

     k dispozici na `http://www.scintilla.org`
   - **Výhody**:
     - naprostá kontrola nad zdrojovým souborem, jednoduché pro naučení, minimální systémové nároky, editor je vhodný i pro operace s jinými typy souborů
   - **Nevýhody**:
     - pro složitější programy těžkopádné, přicházíme o výhodu podpory moderních vývojových prostředků, méně komfortní programování
2. **Eclipse** – budeme používat od zhruba druhé třetiny semestru
   - RAD (*Rapid Application Development*) nástroj špičkové profesionální kvality

     k dispozici na `http://www.eclipse.org/platform`
   - **Výhody**:
     - po zacvičení a poznání i základních možností výrazně zvyšuje produktivitu a komfort programování
       - ♦ významná pomoc při ladění (*debug*)
     - odhady založené na zkušenosti říkají, že pro zkušeného programátora použitím (kvalitního) RAD se efektivita zvyšuje 2 až 3krát
   - **Nevýhody**:
     - pro začátečníka komplikované prostředí

    - ♦ využijeme pouze omezenou část základních možností
- značné systémové nároky, zejména na operační paměť (min. 256 MB, lépe 512 MB)

1.5.2. První program v Javě

vypíše daný text na obrazovku

```
public class PrvniProgram {
  public static void main(String[] args) {
    System.out.println("Ahoj, toto je prvni program");
  }
}
```

po překladu:

```
javac PrvniProgram.java
```

a spuštění:

```
java PrvniProgram
```

se na obrazovku vypíše:

```
Ahoj, toto je prvni program
```

- nejjednodušší program v jazyku Java je tvořen jedním **zdrojovým souborem**
- obsahuje deklaraci **veřejné třídy** (*public class*) pojmenované `PrvniProgram`
    - Konvence („štábní kultura“):
        - ♦ jména tříd se píší s prvním velkým písmenem
        - ♦ vnořené úseky kódu se odsazují (ideálně dvěma mezerami, ne tabulátorem)
- v ní je deklarována hlavní **metoda** (funkce) `main()`
    - je to **veřejná statická metoda** (*public static method*)
    - její první řádek se nazývá **hlavička metody**
    - klíčové slovo `void` vyjadřuje, že metoda **nevrací žádnou hodnotu** (jde o **proceduru**)
    - v závorkách je specifikace **formálního parametru** (*`String[] args`*), který zpočátku nevyužijeme
- zdrojový soubor **musí** mít jméno shodné se jménem veřejné třídy a příponu `.java`, tedy `PrvniProgram.java`
    - jakýkoliv jiný název, včetně `prvniprogram.java`, je chybný

Výstraha

Java **důsledně** rozlišuje malá a velká písmena (*case sensitive*)