Algoritmizace

- Cíle předmětu
  - naučit se sestavovat algoritmy řešení jednoduchých problémů a zapisovat je v jazyku Java
- Organizace předmětu
  - přednášky (učast nepovinná, ale doporučená)
  - cvičení střídavě u tabule a v počítačové učebně
  - zakončení: klasifikovaný zápočet
    - písemný test 30 b
    - test u počítače 30 b
    - aktivita na cvičeních 20 b
    - semestrální práce 20 b
- Vývojové prostředky
  - JBuilder
- Podklady k přednáškám a cvičení:
  `http://service.felk.cvut.cz/courses/X36ALG`

Algoritmizace

- Doporučená literatura
  - Virius, M.: Java pro zelenáče, Neocortex, 2001
  - Herout, P.: Učebnice jazyka JAVA, Kopp, 2000
  - Herout, P.: JAVA, grafické uživatelské prostředí a čeština, Kopp, 2001
  - Eckel, B.: Myslíme v jazyku Java, Grada, 2000, I + II
- Další publikace v češtině:
  - Chapman, S., J.: Začínáme programovat v jazyce JAVA, Computer Press, 2001
  - Pitner,T.: Java, začínáme programovat, Grada, 2002
  - Hawlitzek, JAVA2, příručka programátora, Grada, 2000
  - Shildt, H.: Java 2, Příručka programátora, Softpress, 2001
- Knihy o algoritmech pro pokročilé
  - Weis, M. A.: Data Structures and Problem Solving using Java, Addison-Wesley, 2002
  - Baase,S., van Geodet, A.: Computer Algorithms, Addison-Wesley, 2000
  - Rowe, G., W.: An Itroduction to Data Structures and Algorithms with Java, Prentice Hall, 1998

Algoritmy

- Úloha: najděte největšího společného dělitele čísel 6 a 15
- Řešení: 3
- Popišme postup, jak jsme k řešení dospěli, a to tak, aby byl použitelný pro dvě libovolná přirozená čísla:
  - označme zadaná čísla *x* a *y* a menší z nich *d*
  - není-li *d* společným dělitelem *x* a *y*, pak zmenšíme *d* o 1, test opakujeme a skončíme, až *d* bude společným dělitelem *x* a *y*
- Přesnější popis:

  Vstup: přirozená čísla *x* a *y*

  Výstup: *nsd*(*x,y*)

  Postup:

  1. Je-li *x*<*y*, pak *d* má hodnotu *x*, jinak *d* má hodnotu *y*
  2. Opakuj krok 3, pokud *d* není dělitelem *x* nebo *d* není dělitelem *y*
     3. Zmenši *d* o 1
  4. Výsledkem je hodnota *d*
- Sestavili jsme algoritmus pro výpočet největšího společného dělitele dvou přirozených čísel

Algoritmy

- Proveďme výpočet podle předchozího algoritmu pro čísla 6 a 15
- Předtím je třeba upřesnit význam symbolů *x*, *y* a *d* použitých v algoritmu: jsou to proměnné (paměťová místa), ve kterých je uložena nějaká hodnota, která se může v průběhu výpočtu měnit
- Hodnoty proměnných *x*, *y* a *d* po provedení jednotlivých kroků budeme zapisovat do tabulky

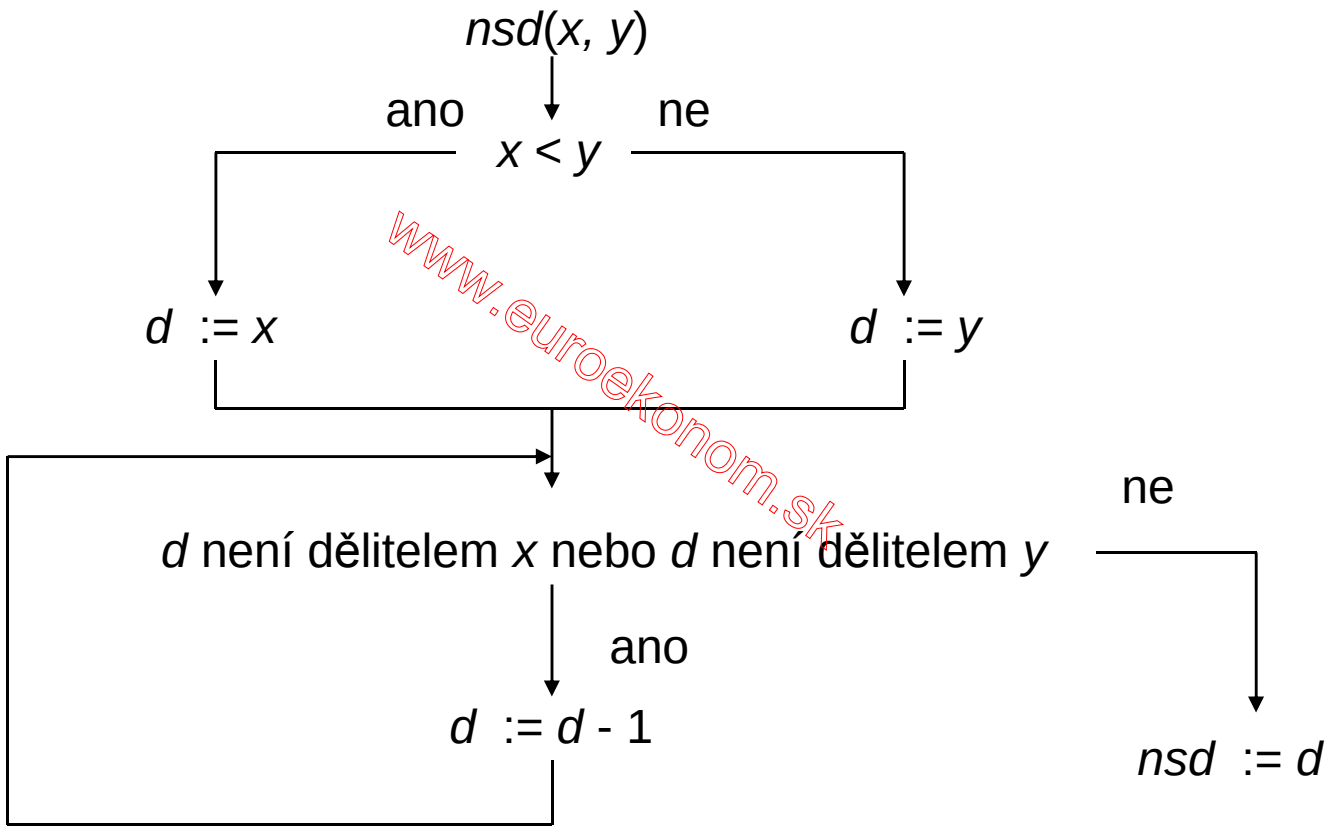
krok	x	y	d	poznámka
	6	15	?	zadání vstupních dat
1	6	15	6	
2	6	15	6	d není dělitelem y, opakuj krok 3
3	6	15	5	
2	6	15	5	d není dělitelem y, opakuj krok 3
3	6	15	4	
2	6	15	4	d není dělitelem y, opakuj krok 3
3	6	15	3	
2	6	15	3	d je dělitelem x i y
4	6	15	3	výsledek je 3

Algoritmy

- Algoritmus
  - postup při řešení určité třídy úloh, který je tvořen seznamem jednoznačně definovaných příkazů a zaručuje, že pro každou přípustnou kombinaci vstupních dat se po provedení konečného počtu kroků dospěje k požadovaným výsledkům
- Vlastnosti algoritmu:
  - hromadnost
    měnitelná vstupní data
  - determinovanost
    každý krok je jednoznačně definován
  - konečnost a resultativnost
    pro přípustná vstupní data se po provedení konečného počtu kroků dojde k požadovaným výsledkům
- Prostředky pro zápis algoritmu
  - přirozený jazyk, vývojové diagramy, struktogramy, pseudojazyk, programovací jazyk

Algoritmy

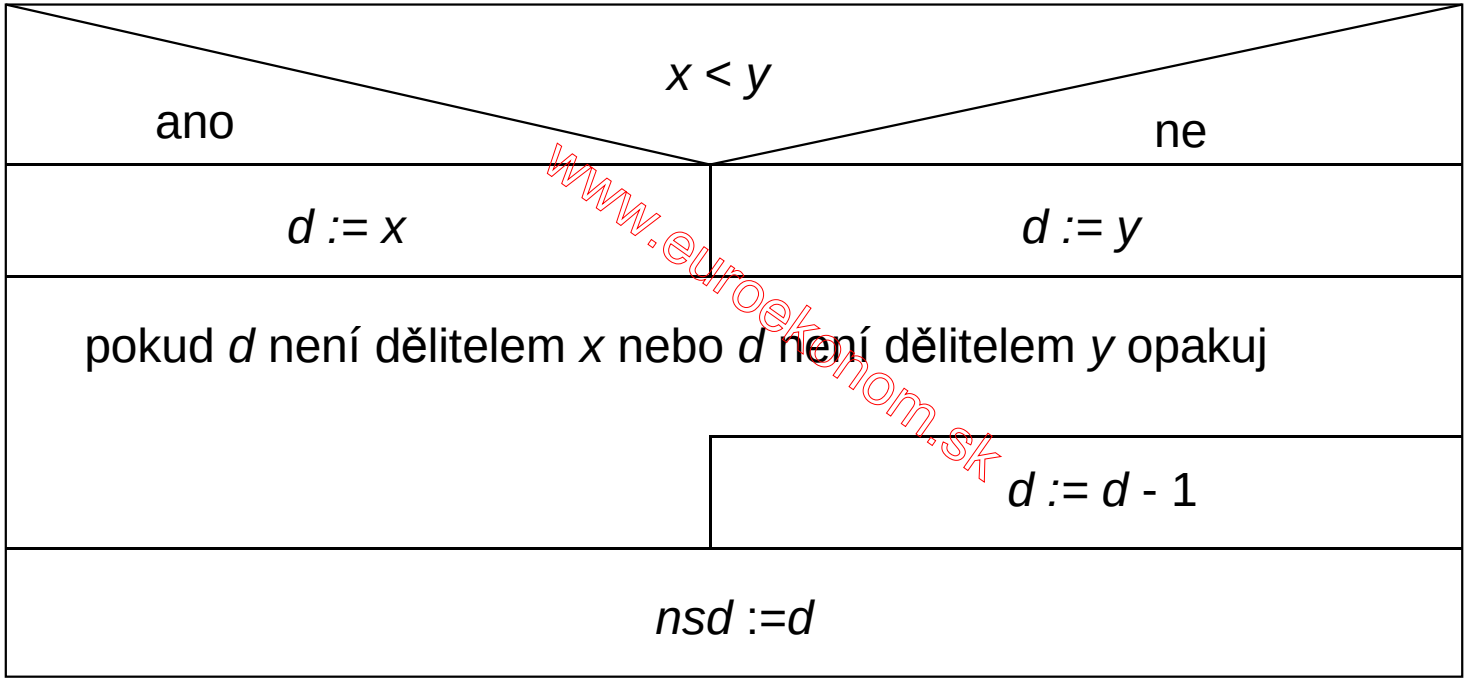
- Vývojový diagram



Algoritmy

- Struktogram

  nsd(*x*, *y*)



Algoritmy

- Zápis algoritmu v pseudojazyku

```
nsd(x,y):
if x<y then d:=x else d:=y;
while d není dělitelem x and d není dělitelem y do
  d:=d-1;
nsd:=d;
```

- Zápis algoritmu v programovacím jazyku

```
int nsd(int x, int y)
{
  int d;
  if (x<y) d=x; else d=y;
  while (x%d!=0 || y%d!=0) d--;
  return d;
}
```

Programy a programovací jazyky

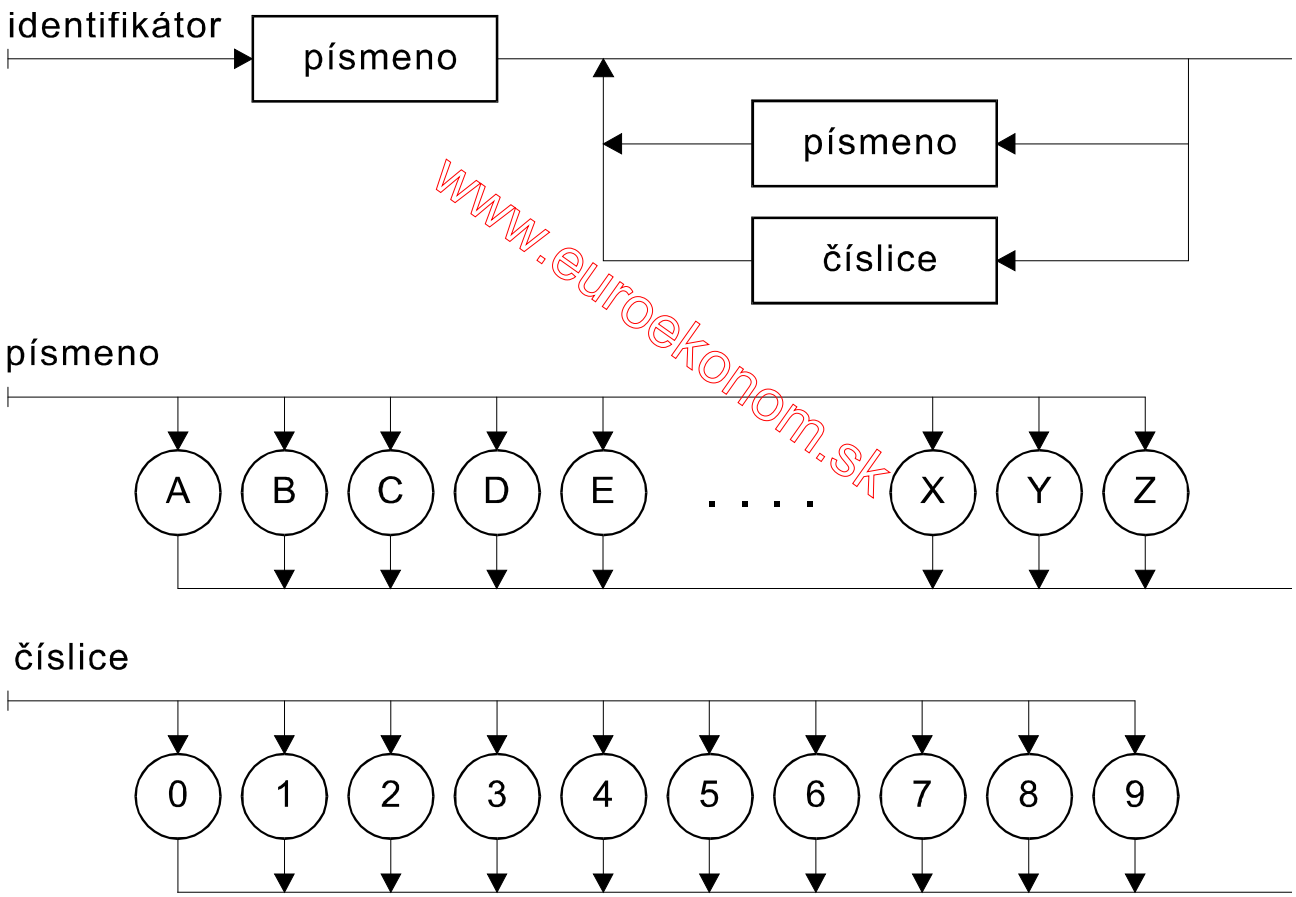
- Program je předpis pro provedení určitých akcí počítačem zapsaný v programovacím jazyku
- Programovací jazyky
  - strojově orientované
    - strojový jazyk = jazyk fyzického procesoru
    - asembler (jazyk symbolických adres)
  - vyšší jazyky
    - **imperativní** (příkazové, procedurální)
    - neimperativní (např. funkcionální)
- Hlavní rysy imperativních jazyků (např. C, C++, Java, Pascal, Basic, ...)
  - zpracovávané údaje mají formu datových objektů různých typů, které jsou v programu reprezentovány pomocí proměnných resp. konstant
  - program obsahuje deklarace a příkazy
  - deklarace definují význam jmen (identifikátorů)
  - příkazy předepisují akce s datovými objekty nebo způsob řízení výpočtu

Syntaxe a sémantika programovacích jazyků

- Syntaxe
  - souhrn pravidel udávajících přípustné tvary dílčích konstrukcí a celého programu
- Sémantika
  - udává význam jednotlivých konstrukcí
- Prostředky pro popis syntaxe
  - syntaktické diagramy
  - různé formy Backus-Naurovy formy
- Sémantika je obvykle popsána slovně

Syntaktické diagramy

- Príklad: identifikátor je posloupnost písmen a číslic začínající písmenem



Rozšířená BNF

- Rozšířená Backus-Naurova forma – EBNF
- Příklad: identifikátor

  identifikátor = písmeno {písmeno | číslice}

  písmeno = 'A' | 'B' | 'C' | 'D' | ... | 'X' | 'Y' | 'Z'

  číslice = '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '9'
- Neterminály:

  identifikátor, písmeno, číslice
- Terminály:

  ‘A’, ‘B’, ...
- Význam metasymbolů:

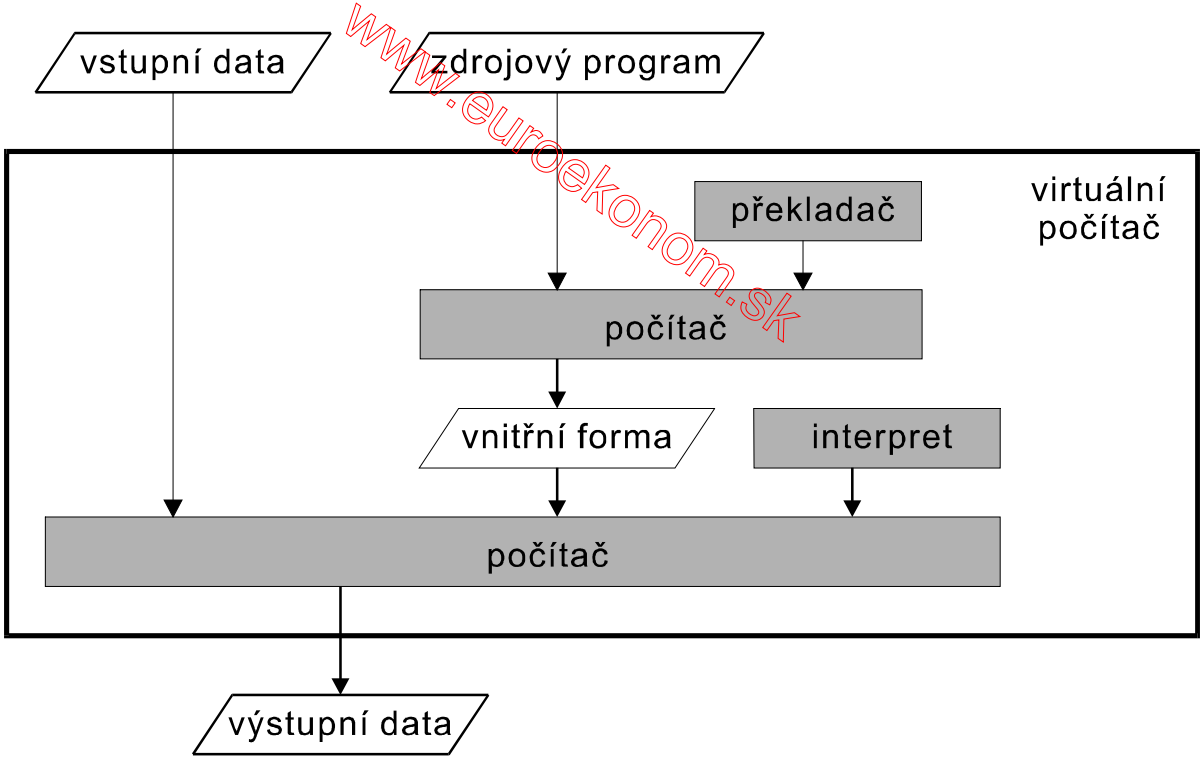
  {*x*} žádný nebo několik výskytů *x*

  x | *y* *x* nebo *y*

  [*x*] žádný nebo jeden výskyt *x*

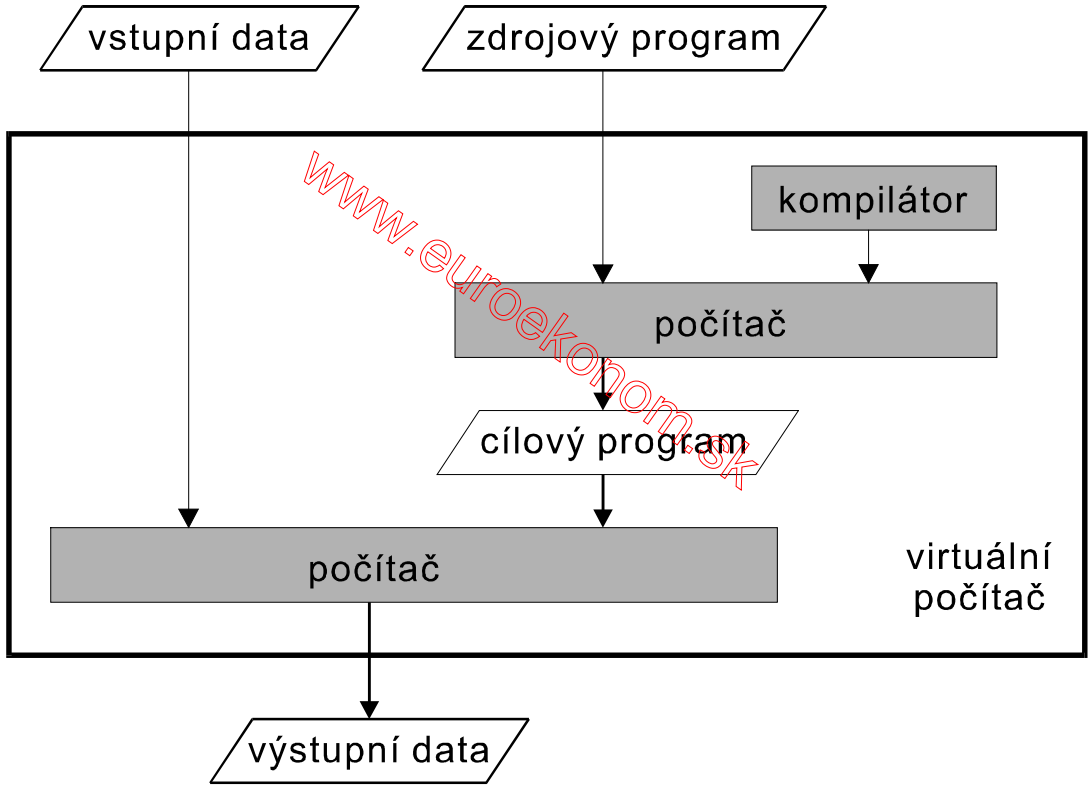
Implementace programovacích jazyků

- Dvě základní metody:
  - interpretační
  - kompilační
- Interpretační metoda:



Implementace programovacích jazyků

- Kompilační metoda:



Úvod do jazyka Java

- Pro prezentaci, návrh a ověřování algoritmů použijeme jazyk Java
- Proč?
  - jde o vyšší, obecně použitelný programovací jazyk s vysokým stupněm zabezpečení
  - je objektově orientovaný, umožňuje však i klasické procedurální programování
  - syntaxe výrazů a příkazů vychází z jazyka C; přechod z Javy na C nebo C++ je tedy jednodušší, než přechod z Pascalu
  - základní implementaci (JDK – Java Development Kit) firmy Sun lze pro prostředí Windows i Unix stáhnout ze stránek firmy Sun: *http://java.sun.com*
  - interaktivní grafické vývojové prostředí JBuilder firmy Borland lze pro prostředí Windows i Unix stáhnout ze stránek firmy Borland: *http://www.borland.cz*
  - vytvořené programy jsou zcela portabilní (program vytvořený pod Windows bez problémů funguje pod Unixem a naopak)

Úvod do jazyka Java

- Jazyk Java je implementován interpretačním způsobem
  - program je tvořen jedním nebo několika zdrojovými soubory s příponou *.java*
  - zdrojové soubory se přeloží překladačem *javac* (v terminologii firmy Sun to je kompilátor) do vnitřní formy (byte code, bajt-kód), která je počítačově 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, které je třeba mít k dispozici jak při překladu, tak při interpretaci
- Programy v jazyku Java budeme vytvářet pomocí vývojového systému JBuilder, který přípravu programu, jeho překlad a provedení zjednodušuje
- Se systémem JBuilder se seznámíte na 1. cvičení v počítačové učebně

První program

- Příklad programu, který vypíše daný text na obrazovku:

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

- Po překladu a spuštění se na obrazovku vypíše

```
Nazdar, toto je prvni program
```

- Nejjednodušší zdrojový program v jazyku Java je tvořen jedním souborem, který obsahuje deklaraci veřejné třídy (public class), ve které je deklarována hlavní funkce *main* (veřejná statická metoda, public static method)
- Soubor musí mít jméno shodné se jménem veřejné třídy a příponu *.java*
- První řádek deklarace hlavní funkce *main* (hlavička funkce) obsahuje:
  - klíčové slovo *void*, které vyjadřuje, že funkce nevrací žádnou hodnotu (jde o proceduru)
  - v závorkách specifikaci parametru, který zpočátku nevyužijeme
- Konvence: jména tříd se píší s prvním velkým písmenem

Příklady k přednáškám

- První program a všechny další, které budou prezentovány na přednáškách, jsou v adresáři *alg_p*
- Struktura adresáře:

alg_p
- src
  - alg1 — zdrojové programy k tématu alg1
  - alg2 — zdrojové programy k tématu alg2
  - ...
- classes
  - ...

- Podadresáře *alg1*, *alg2* atd. představují tzv. balíky (packages)
- Každý zdrojový soubor umístěný v balíku začíná specifikací balíku

```
package alg1;
public class PrvniProgram {
  public static void main(String[] args) {
    System.out.println("Nazdar, toto je prvni program");
  }
}
```