Algoritmizace

- **Cíl předmětu** - naučit se sestavovat algorimy řešení základních problémů a zapisovat je v jazyku Java.
- **Jádrem předmětu** jsou data, typy, výrazy a příkazy demonstrované v programovacím jazyce Java, základy programovacích technik a datové abstrakce.
  Předmět se nezabývá hardwarem, telekomunikacemi ani jinými jazyky či asemblery.
- **Algoritmizace je „prerekvizita“ následujících předmětů!!**
- **Navštivte weby našich kateder podílejících se na tomto programu**
  - **Katedra kybernetiky**
  - **Katedra počítačů**
  - **Katedra počítačové grafiky a interakce**
  - **Katedra ekonomiky, manažerství a humanitních věd**
- **Organizace předmětu**
  - přednášky (účast nepovinná, ale doporučená)
  - **cvičení jsou povinná** (dvě absence jsou tolerovány)
- **Stránky předmětu s materiály k přednáškám a cvičením:**
  http://cw.felk.cvut.cz/doku.php/courses/y36alg/start

Zakončení kurzu

- **Zakončení**: zápočet + zkouška
- **Klasifikace** na základě bodového zisku (max. 100 b):
  - ústní zkouška 10 b (-5b)
  - písemný zkouškový test 25 b (min. 10 b)
  - test u počítače na cvičeních 30 b (min. 15 b)
  - aktivita na cvičeních a DÚ 20 b (min. 10 b)
  - semestrální práce 20 b (min. 10 b)
- **Hodnocení**

	body	**číselně**	**slovní**
A	100-90	1	výborně
B	89-80	1,5	velmi dobře
C	79-70	2	dobře
D	69-60	2,5	uspokojivě
E	59-50	3	dostatečně
F	méně než 50 bodů	4	nedostatečně (op.)

Algoritmizace - Témata přednášek

1. Algoritmy, programy, programovací jazyky
2. Proměnné a výrazy
3. Řídicí struktury
4. Funkce
5. Rozklad problému na podproblémy
6. Pole
7. Třídy a objekty I
8. Soubory, textové soubory
9. Složitost algoritmů
10. Třídy a objekty II
11. Spojové struktury
12. Datové abstrakce
13. Rezerva

Témata cvičení a harmonogram testů

1. Seznámení s prostředím NetBeans
2. Proměnné, výrazy, přiřazení, vstup a výstup
3. Větvení
4. Cykly
5. Funkce **+ zadání semestrální práce**
6. Pole
7. Objekty
8. Rekurze
9. Soubory
10. Složitost algoritmů **+ TEST**
11. Třídy **+ odevzdání semestrální práce**
12. Datové struktury **+ prezentace semestrálních prací**
13. **Prezentace semestrálních prací + zápočet**

Změny ve výuce

Svátky: 28. 9. (PO), 28. 10. (ST) a 17. 11. (ÚT)

Přesun výuky: 20. 11. (PÁ), výuka jako PO téhož týdne

Rektorské volno: 14. 10. (ST) od 14,00 hod.

Děkanský den: 16. 11. (PO)

Zdroje pro studium

- **Základní příručky:**
  - **Herout, P.: Učebnice jazyka JAVA, Kopp, 2000**
  - Virius, M.: Java pro zelenáče, Neocortex, 2001
- **Další zdroje**
  - Eckel, B.: Myslíme v jazyku Java, Grada, 2000, I + II
  - 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
  - Herout, P.: JAVA, grafické uživatelské prostředí a čeština, Kopp, 2001

Algoritmy: Motivační příklad

- **Úloha: najděte největšího společného dělitele čísel 6 a 15**
- **Obecný postup**:

  Popišme postup tak, aby byl použitelný pro dvě libovolná přirozená čísla, nejen pro 6 a 15:

  - 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

Algoritmy: Motivační příklad

- **Úloha: najděte největšího společného dělitele čísel 6 a 15**
- **Obecný postup**:

  Popišme postup tak, aby byl použitelný pro dvě libovolná přirozená čísla, nejen pro 6 a 15:

  - 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*)

  Metoda:

  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*

Algoritmy: Motivační příklad

- Proveďme výpočet podle předchozího algoritmu pro čísla **6** a **15**.

  Poznámka: Symboly *x*, *y* a *d* použité v algoritmu jsou **proměnné** (paměťová místa), ve kterých je uložena nějaká hodnota, která se může v průběhu výpočtu měnit.

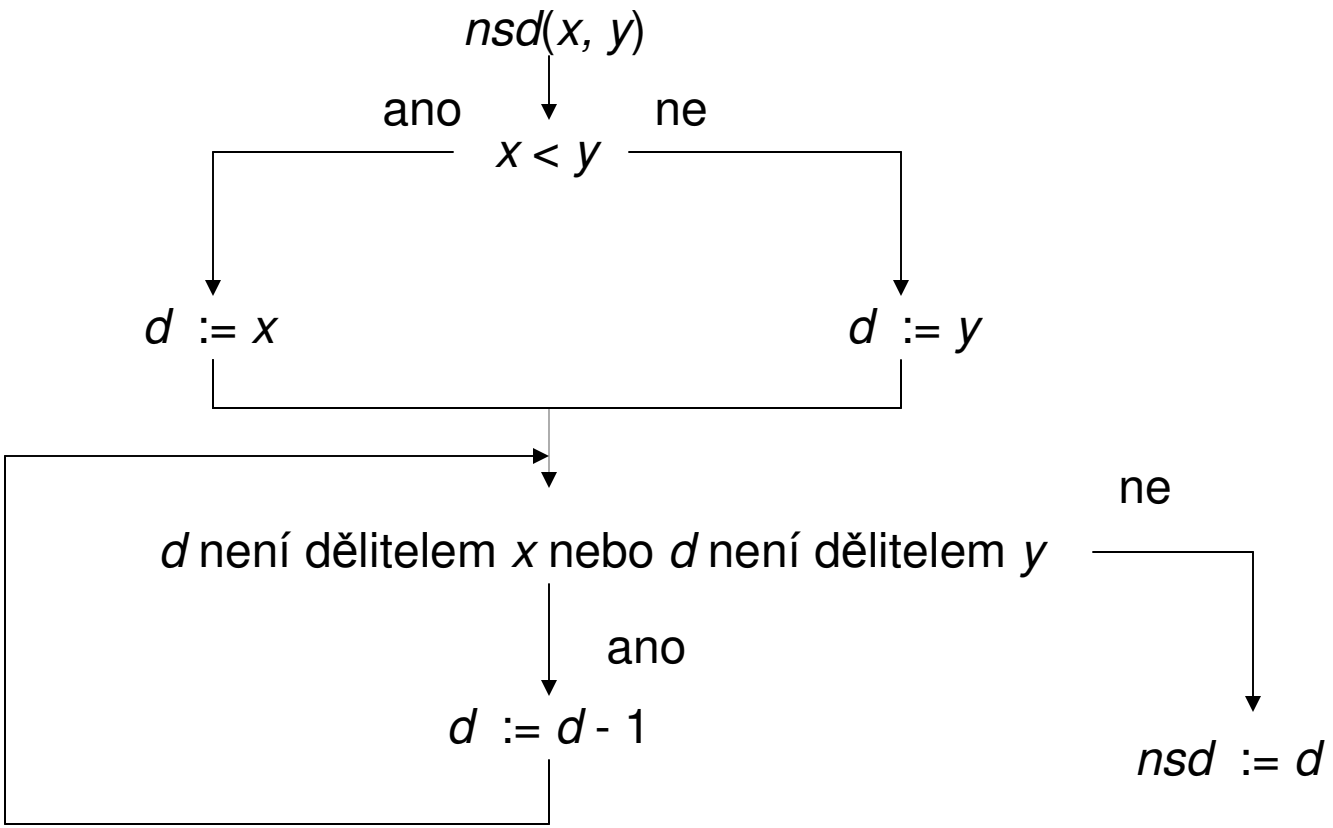
krok		**x**	**y**	**d**	**poznámka**
	6	**15**	**?**		**zadání vstupních dat**
1	6	15	6		
2	6	**15**	**6**		**d není dělitelem y**, proveď krok 3
3	6	15	5		
2	**6**	15	**5**		**d není dělitelem x,** proveď krok 3
3	6	15	4		
2	**6**	**15**	**4**		**d není dělitelem x ani y,** proveď krok 3
3	6	15	3		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

Algoritmy

- **Algoritmus** - postup pro 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
- Algoritmus – syntetický model postupu řešení obecných úloh
- **Prostředky pro zápis algoritmu**
  - přirozený jazyk, vývojové diagramy, struktogramy, pseudojazyk, programovací jazyk

Algoritmy

- **Vývojový diagram**



Algoritmy

- **Zápis algoritmu v pseudokódu**

```
nsd(x,y):
if x<y then d:=x else d:=y;
while d „není dělitelem" x or 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

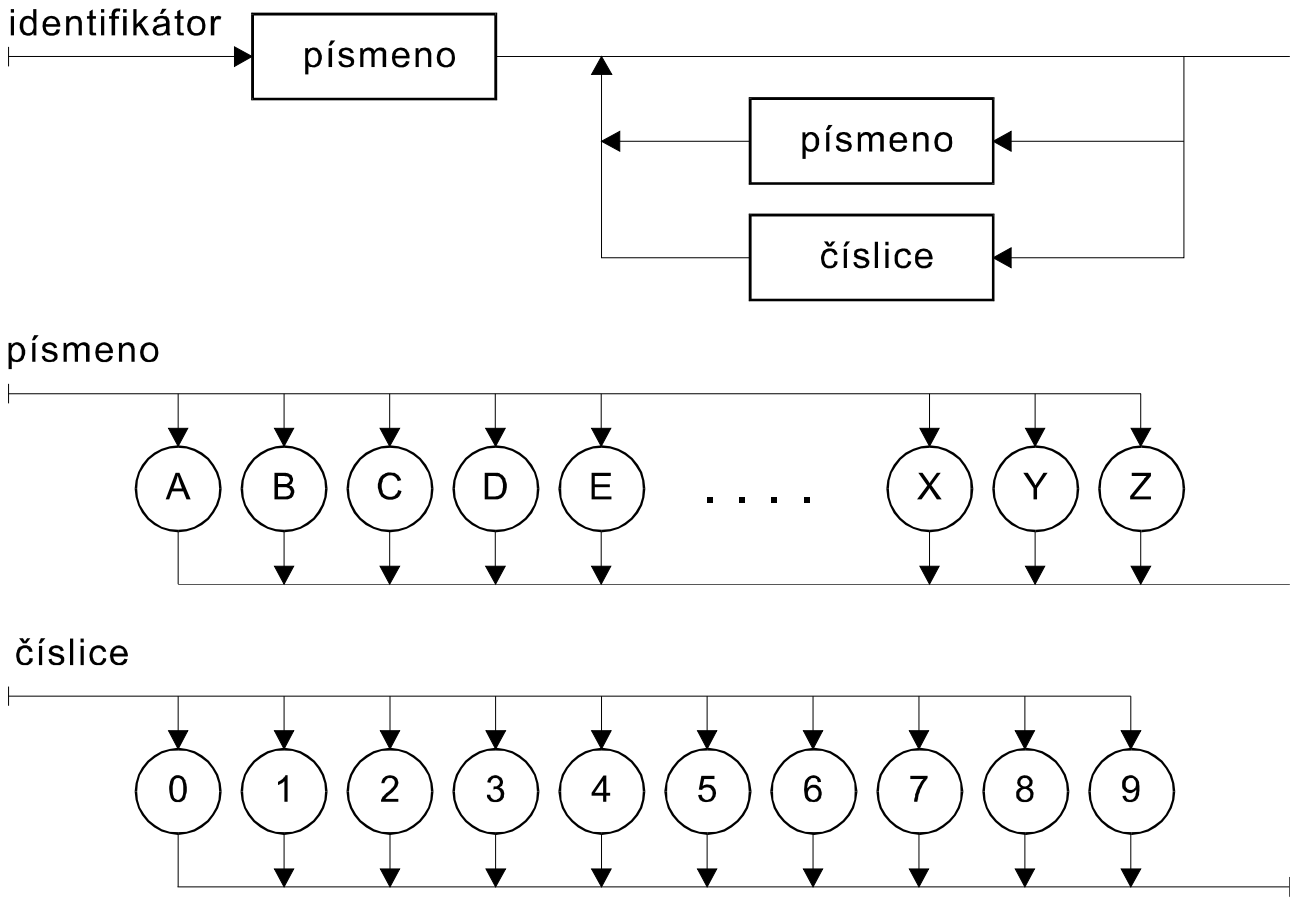
- Program je předpis (zápis algoritmu) pro provedení určitých akcí počítačem zapsaný v programovacím jazyku.
- Programovací jazyky
  - strojově orientované
    - strojový jazyk = jazyk fyzického procesoru
    - asembler (jazyk symbolických adres)
  - vyšší jazyky
    - **imperativní** (příkazové, procedurální)
    - 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 dat. objekty nebo způsob řízení výpočtu

Vlastnosti programovacích jazyků

- **Syntaxe** - souhrn pravidel udávajících přípustné tvary dílčích konstrukcí a celého programu.
  - syntaktické diagramy
  - různé formy Backus-Naurovy formy
- **Sémantika** - udává význam jednotlivých konstrukcí.
  - obvykle popsána slovně

Syntaktické diagramy

- Příklad: identifikátor je posloupnost písmen a číslic začínající písmenem



Rozšířená Backus-Naurova forma

- 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:

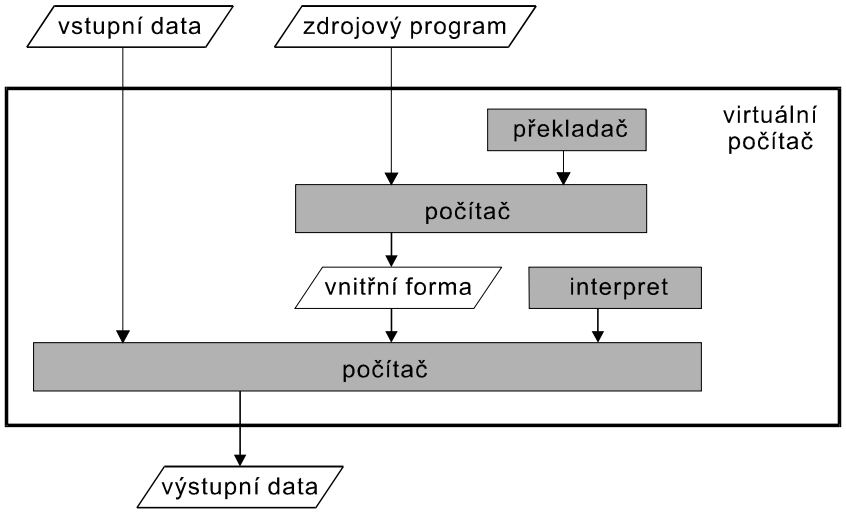
  {*x*} ... žádný nebo několik výskytů *x*

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

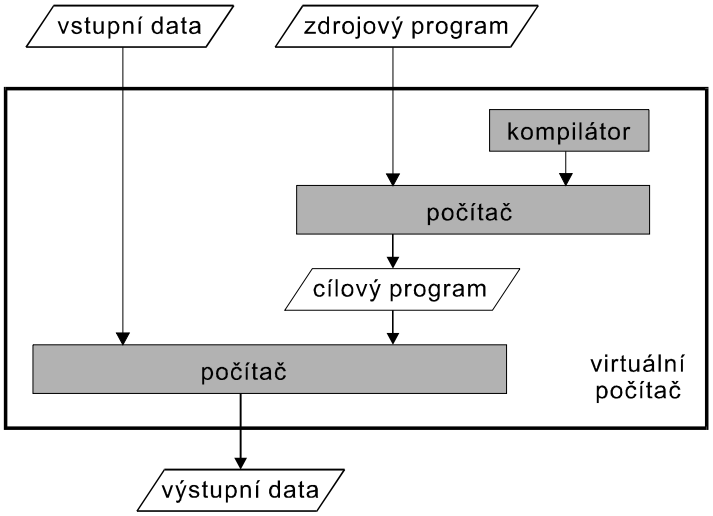
pro zájemce

Implementace programovacích jazyků

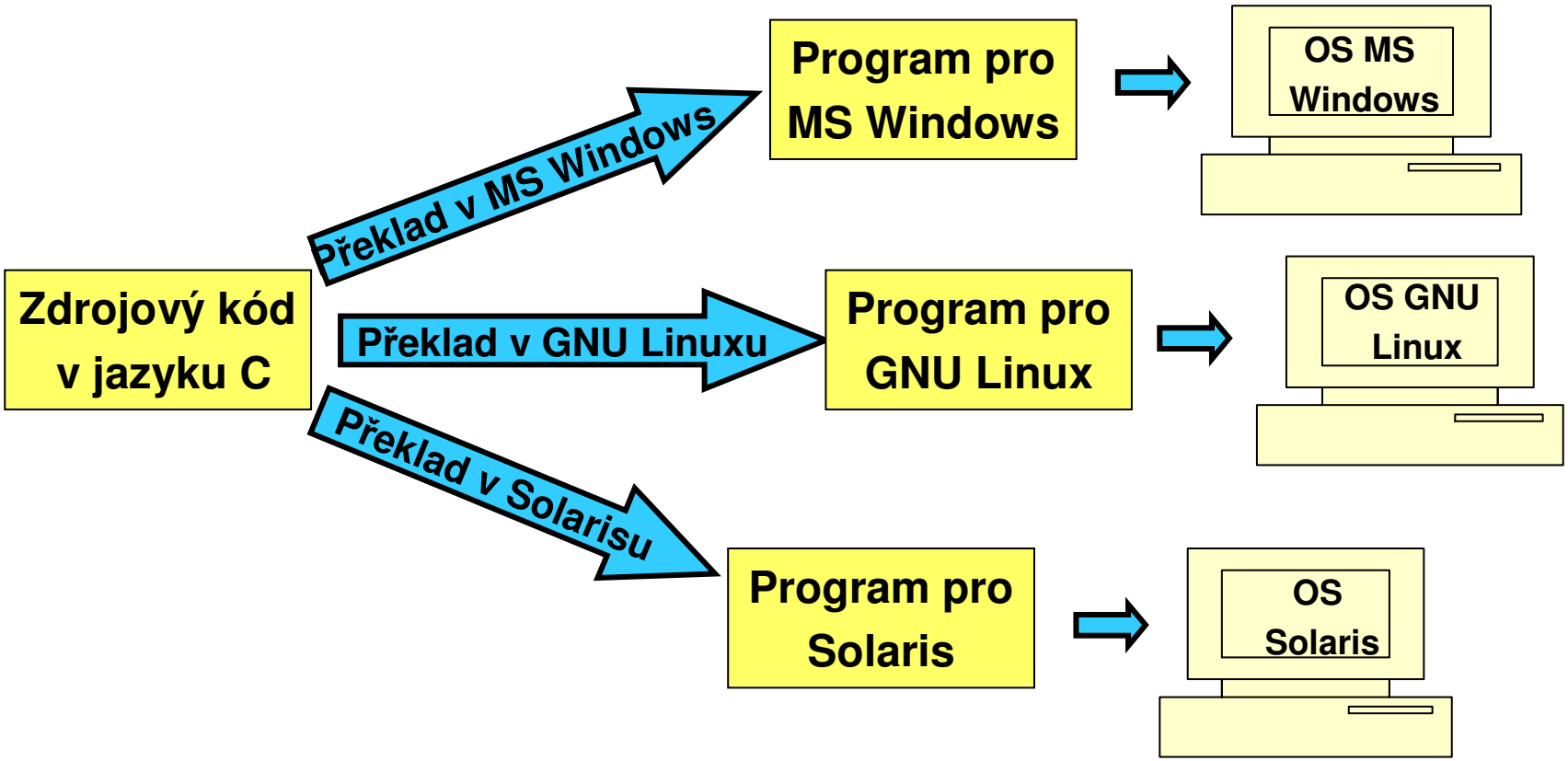
- **Interpretační metoda**



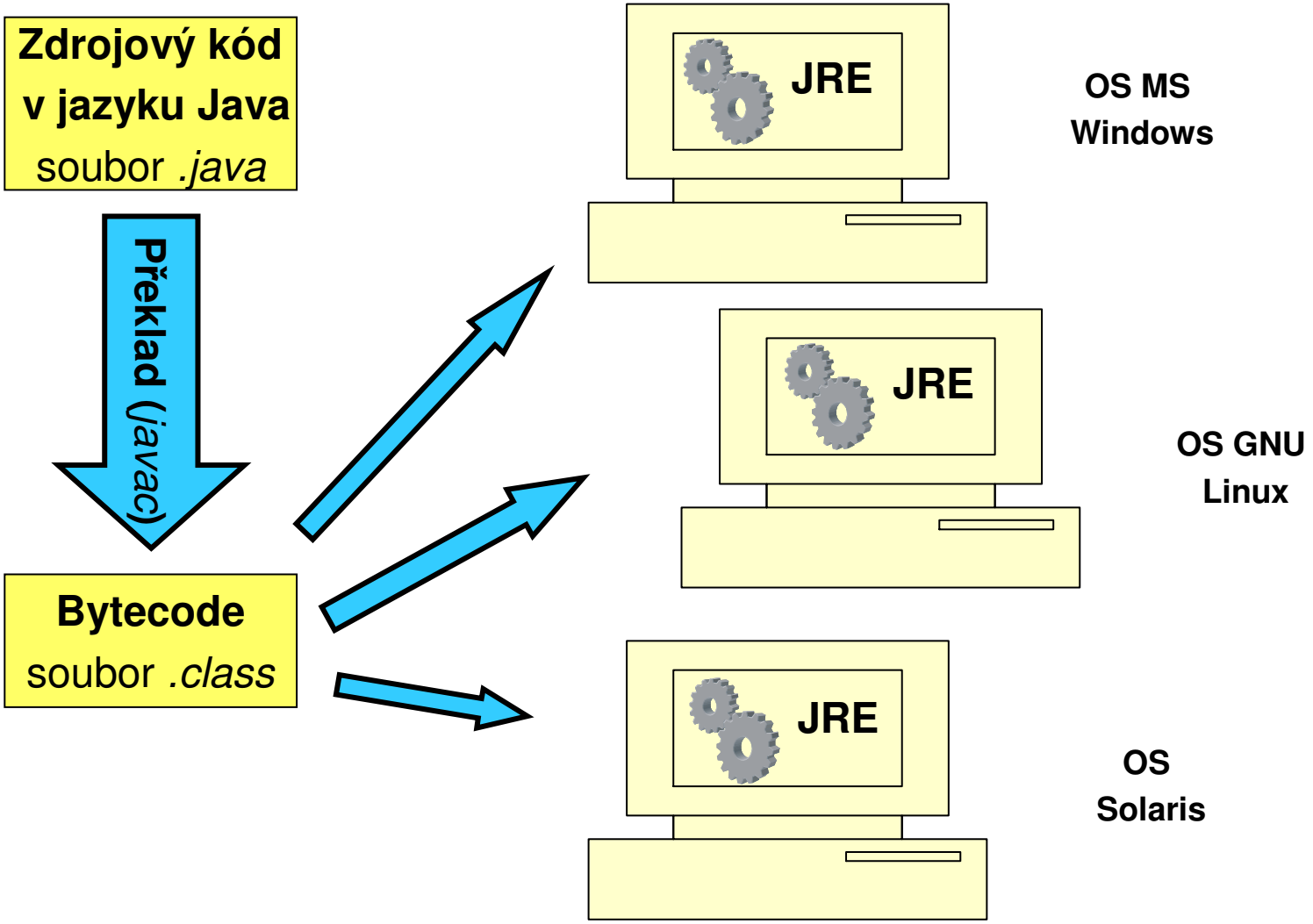
- **Kompilační metoda**



Kompilační metoda - jazyk C



Interpretační metoda - jazyk Java



Fáze řešení algoritmické úlohy

1. Zadání.
2. Rozbor, analýza – specifikace vstupů, výstupů.
3. Algoritmus – pseudokód, vývojový diagram.
4. Testování.
5. Kódování.
6. Laděné – odstranění chyb v programu.
7. Optimalizace kódu – paměťová a časová efektivita.
8. Dokumentace.

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
  - vytvořené programy jsou zcela portabilní (program vytvořený pod Windows bez problémů funguje pod Unixem a naopak)
  - 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*
  - my používáme vývojové prostředí NetBeans 6.7, fy. Sun Microsystem. *http://www.netbeans.org/*

Úvod do jazyka Java

- **Jazyk Java** je implementován interpretačním způsobem
  - program je tvořen jedním nebo několika **zdrojovými soubory** s příponou **`.java`**:

    `Program.java`
  - zdrojové soubory se přeloží **překladačem(*)** ***javac*** do vnitřní formy (**byte code**, bajt-kód) s příponou **`.class`**

    `Program.java > javac > Program.class`
  - **interpretaci** vnitřní formy provede program ***java*** *(v prostředí* JVM – Java Virtual Machine) a provede výpočet

    `Program.class > java > „výpočet“`

  Poznámka: (*) v terminologii firmy Sun to je kompilátor.
- Program obvykle **využívá řadu knihoven**, které je třeba mít k dispozici jak při překladu, tak při interpretaci!!!

Zpracování programu v jazyku JAVA

Zdrojový soubor: **`Program.java`**

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

Spuštění překladače do byte-code: **`javac Program.java`**

Vznikne: **`Program.class`**

Spuštění interpretru: **`java Program`**

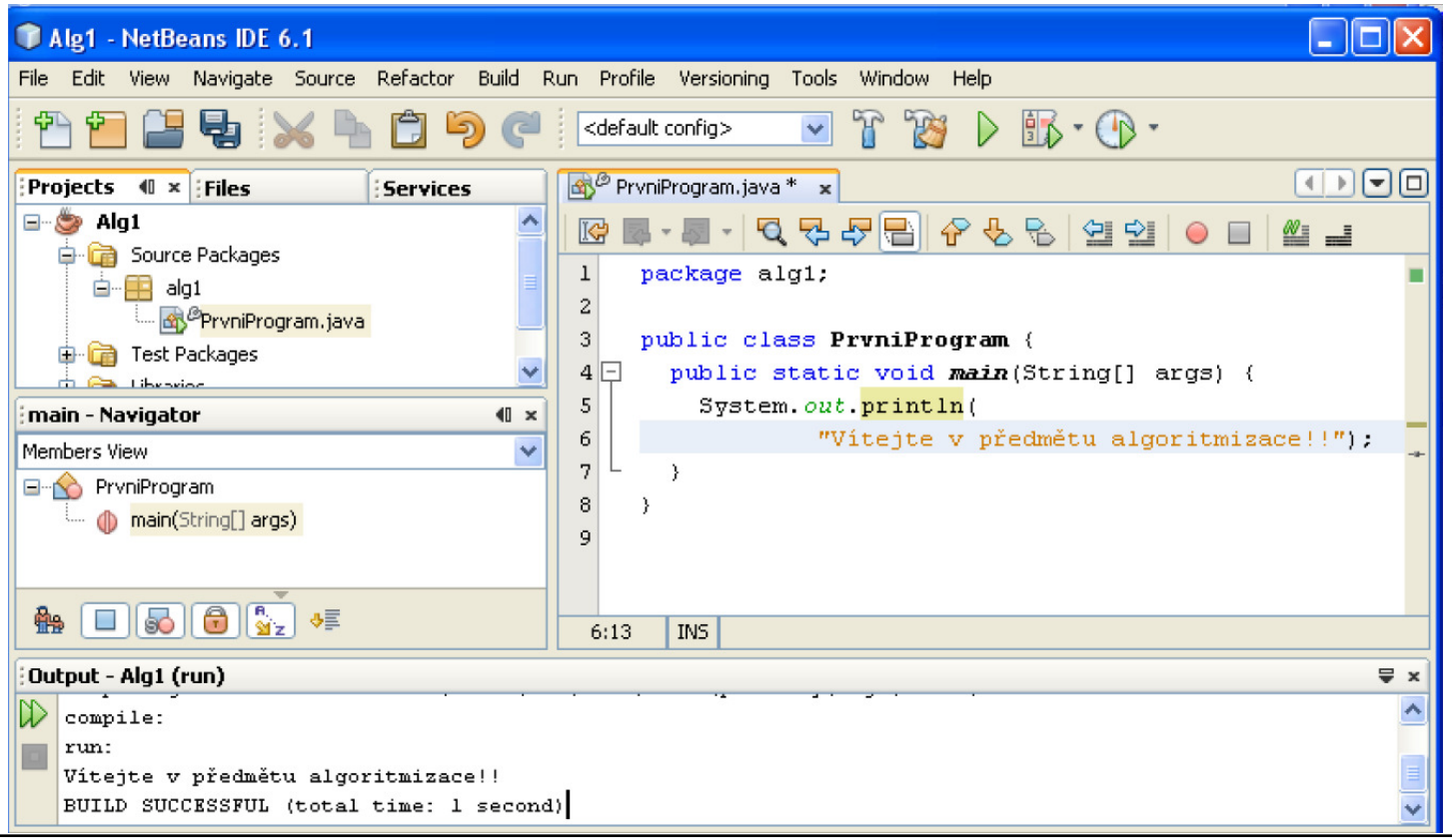
Výstup programu: Nazdar, toto je prvni program

První program v jazyku Java

- Nejjednodušší zdrojový program - jeden soubor
  - deklarace veřejné třídy (`public class`),
  - hlavní funkce `main` (veřejná statická metoda, public static method)
- Soubor musí mít jméno shodné se jménem veřejné třídy a příponu *.java*
- Hlavička funkce `main` ():
  - klíčová slova `public static void` (`void` - procedura)
  - `(String[] args)` specifikace vstupních parametrů
- Konvence: jména tříd se píší s prvním velkým písmenem

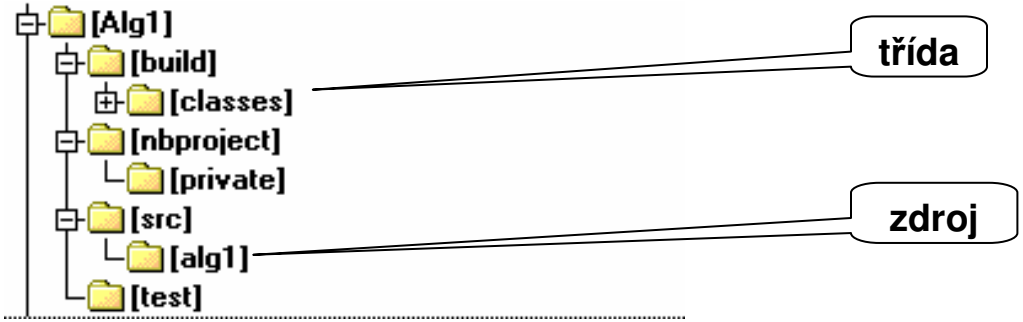
Úvod do jazyka Java, vývojový systém

- Programy v jazyku Java budeme vytvářet pomocí **vývojového prostředí NetBeans**, který proces přípravy programu, jeho překlad a provedení zjednodušuje.
- Se systémem NetBeans se seznámíte na 1. cvičení v počítačové učebně



Příklady k přednáškám

- První program a všechny další, které budou prezentovány na přednáškách, jsou v adresáři *Algp*, kde *p* je číslo přednášky.
- Struktura adresáře pro příklad 1.přednášky (zdroj. programy k tématu alg1):



- 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");
  }
}
```

Jména a konvence

- **package** (balíček) - jen malá písmena, i několik jmen oddělených tečkou,
  např: alg1, java.util
- **class** (třída), **abstraktní třída**, jejich **konstruktory**, **rozhraní** - začínají velkým písmenem,
  např: String, MyFirstClass, Serializable, Comparable

  Výjimky by měly mít sufix Exception,
  např: MySpecialException
- **method** (metoda) - začínají malým písmenem, další slovo začíná velkým písmenem,
  např: setBorder, isEmpty, getNumber
- **variable** (proměnná) - začínají vždy malým písmenem, další slovo začíná velkým písmenem,
  např: diskriminant, totalCount
- **final variable** (konstanta) a **návěští** - jen velká písmena, jednotlivá slova oddělena podtržítkem,
  např. MAX_COUNT, RED