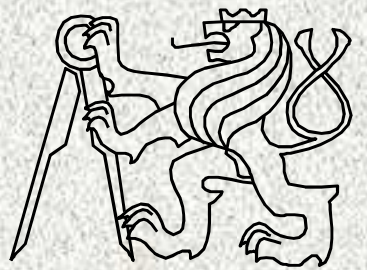
Programování 1 - úvod

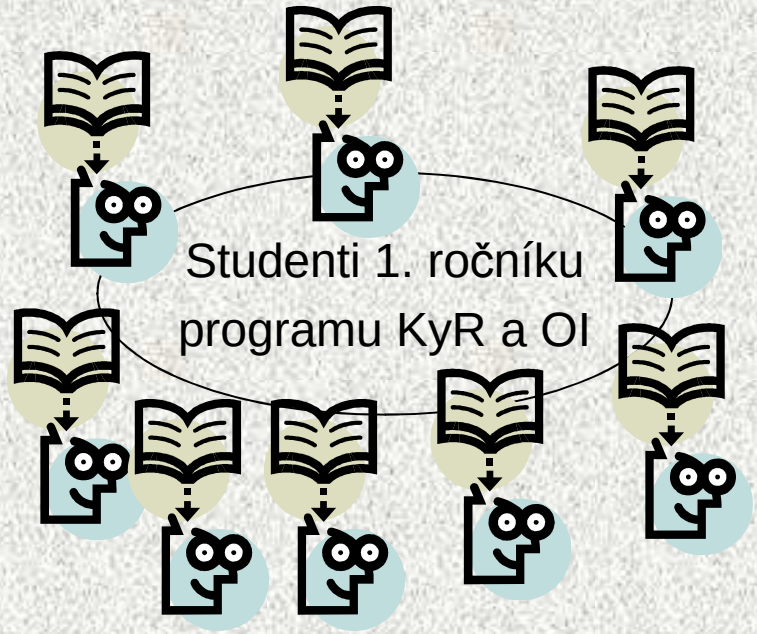


A0B36PR1-Programování 1

Fakulta elektrotechnická

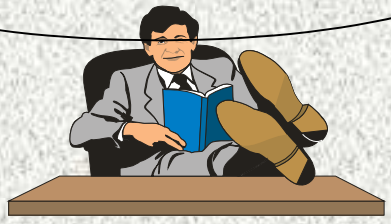
České vysoké učení technické

Vítejte, představme se … ☺



Přednášející
Doc. Ing. Ivan Jelínek, CSc.

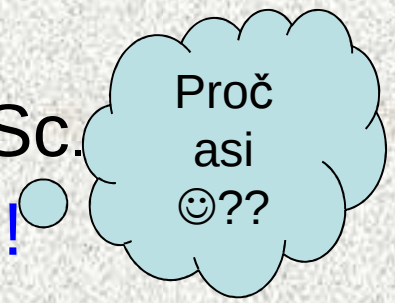
Cvičící:….



- Jste STUDENTI ČVUT FEL!
- Jste členy akademické obce!
- Děkan, senát, vědecká rada
- Zaměstnanci (tituly …), doktorandi
- Katedry

Kdo je kdo v Programování 1

1. Vy – studenti
   - Různé zkušenosti s různými programovacími jazyky
2. Přednášející:
   - Doc.Ing. Jelínek Ivan CSc.
3. Nejdůležitější učitelé: cvičící!!
   - Ing. Buk Zdeněk
   - Ing. M. Filipský
   - Ing. M. Balík
   - Ing. M. Mudroch
   - Ing. J. Tožička
- Garant: Doc.Ing. Jelínek Ivan CSc.



Organizace a hodnocení předmětu

- A0B36PR1 Programování 1
- Rozsah: 2p+2c
- Zakončení Z,ZK
- Kredity 6
  - po prvním semestru je nutné získat alespoň 15
  - výsledná známka ovlivní možnost tvorby rozvrhu v letním semestru

(*)Způsoby zakončení předmětu:
- zápočet
- klasifikovaný zápočet
- zápočet / zkouška

Cíl předmětu

- Programování 1

- prerekvizita Programováni 2
- prerekvizita Algoritmizace

- *reprezentace dat v počítači*
- *reprezentace čísel*
- základní datové a řídící struktury jazyka Java
- cykly
- jednoduché programy v Javě
- ladění programů
- procedurální vs. objektový přístup
- objekty, třídy
- soubory a proudy, kolekce,
- …

Osnova přednášek

1. **Základní pojmy výpočetní techniky, operační systém, software, překladač, interpret, programovací jazyky, syntaxe, sémantika**
2. **Koncepce Javy, základní vlastnosti, současná podoba a vývoj, úvod do jazyka, zpracování programu, vnitřní forma, vývojová prostředí,**
3. **Struktura programu, vývoj programu, ladění programu, vývojová prostředí,proměnné, výrazy, typy, logické a číselné operátory, první program.**
4. **Vstup a výstup, řídící konstrukce, větvení, cykly**
5. **Funkce, procedury, parametry, statické proměnné, lokální proměnné, blok, princip přidělování paměti proměnným, halda, zásobník**
6. **Pole, referenční proměnná typu pole, pole jako parametr, funkce typu pole.**
7. **Třídy 1 - Principy objektového přístupu, třídy, třída jako programová jednotka, třída jako zdroj funkcí, třída jako datový typ, statické a instanční metody,**
8. **Třídy 2 - zapouzdření, setry, getry, metody třídy Object -equals, toString, hashCode**
9. **Rozklad problému na podproblémy, princip rekurze a iterace**
10. **Spojové struktury, zásobník, fronta, stromy, zásobník, fronta.**
11. **Úvod do ADT, definice ADT, ADT množina, zásobník, fronta, tabulka. Implementace pomocí polí, kolekce**
12. **Soubory a proudy, soubor jako posloupnost bytů,úvod do zpracování výjimek, ukládání/čtení primitivních typů, primitivních typů a objektů (řetězců), objektů do souboru – serializace;**
13. **Základní principy vyhledávání, řazení**
14. **Rezerva**

Osnova cvičení

1. Seznámení s počítačovou učebnou a výpočetním prostředím
2. Seznámení s vývojovým prostředím pro programování,
3. Struktura programu v jazyku Java, zadání semestrální práce
4. Sekvence, vstup, výstup, větvení
5. Cykly, odladění triviálních úloh ve vývojovém prostředí
6. Řešení složitější úlohy, rozklad na podproblémy, procedury a funkc
7. Pole
8. Třída jako datový typ
9. Třídy a dědičnost
10. Rekurze
11. Spojové struktury a ADT
12. Test + spojové struktury
13. Soubory a proudy
14. Zápočet

Doporučená literatura

Základní zdroje:

- Poznámky z přednášek a cvičení
- Slidy z přednášek http://eduweb.fel.cvut.cz/courses/A0B36PR1
- Základní příručky jazyka Java:
  - Zakhour, S: Java 6, výukový kurz, CPress, Brno, 2007
  - Herout, P.: Učebnice jazyka Java, Kopp, 2007
  - Keogh, J.:Java bez předchozích znalostí, Computer Press, 2005
  - Virius, M.: Java pro zelenáče, Neocortex, 2001

Další zdroje (publikace v češtině**)**:

- 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

Hodnocení a zkouška

Zdroje bodů	max. bodů	min. bodů
4 domácí úlohy	30	20
semestrální práce	24	12
test na cvičeních	16	8
zkouškový test	20	10
ústní zkouška	20	-10b

Minimální počet bodů pro zápočet je 40 bodů

Body ze cvičení, maximálně 70. 60 a více bodů → možnost A, B, C podle prémiového testu

Možnost nechat si zapsat známku nebo jít k ústní zkoušce – odečte se 10 bodů

Klasifikace *na základě bodového hodnocení)*				
klasifikace	***počet bodů***	**číselně**	**slovně**	
A	**90 - 100**	**1**	**výborně**	
B	**80 - 89**	**1,5**	**velmi dobře**	
C	**70 - 79**	**2**	**dobře**	
D	**60 – 69**	**2,5**	**uspokojivě**	
E	**50 - 59**	**3**	**dostatečně**	
F	**< 50**	**4**	**nedostatečně**	

Zakončení předmětu: zápočet, **zkouška** *(na základě bodového hodnocení)*

Začínáme

Šest zákonů programování

1. V každém programu je alespoň jedna chyba
2. Každý program lze zkrátit alespoň o jeden řádek
3. Nejjednodušší chyby se nejhůře hledají
4. Každou opravou se do programu zanese nová chyba
5. Když už se zdá, že program je v pořádku, určitě jste něco přehlédli
6. Programátor dělá to co umí, počítač si dělá, co chce

Začínáme doopravdy

Algoritmus

- 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
- Algoritmus – syntetický model postupu řešení obecných úloh

- Prostředky pro zápis algoritmu
  - přirozený jazyk, vývojové diagramy, struktogramy, pseudojazyk, programovací jazyk

Algoritmus, slovní popis, analýza

- Úloha:

  Najděte největšího společného dělitele čísel 6 a 15

- Řešení:

  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*

- Poznámka:

  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

Algoritmus – „průběh výpočtu“

Úloha: najděte největšího společného dělitele čísel 6 a 15
Průběh řešení:

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

Algoritmus - „zápis“

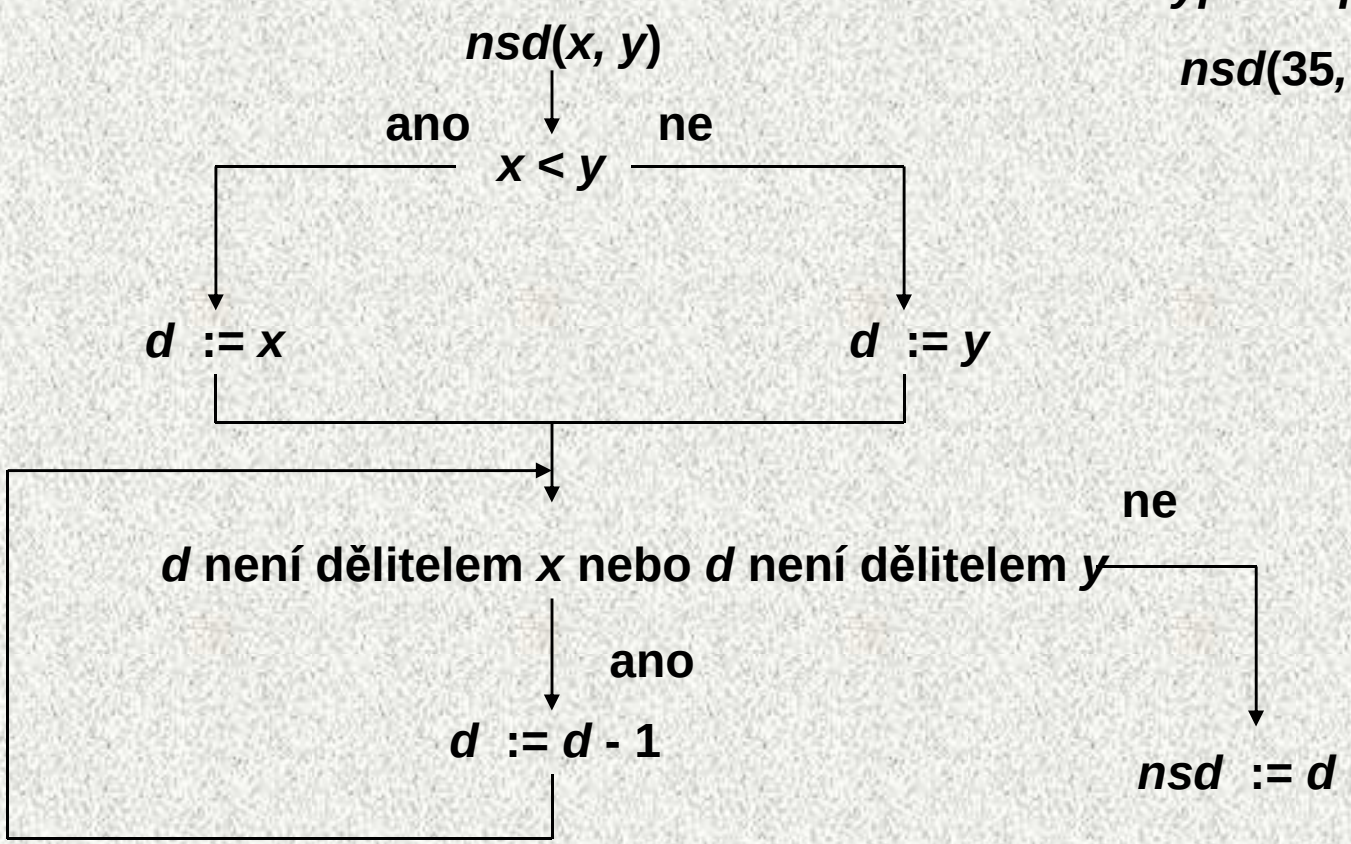
- Úloha:
  najděte největšího společného dělitele

- Obecný 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, vývojový diagram

- Vývojový diagram



Výpočet pro

***nsd*(35, *7*)**

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 datovými objekty nebo způsob řízení výpočtu

Algoritmy a progra

- Zápis algoritmu v pseudojazyku

```
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;
}
```

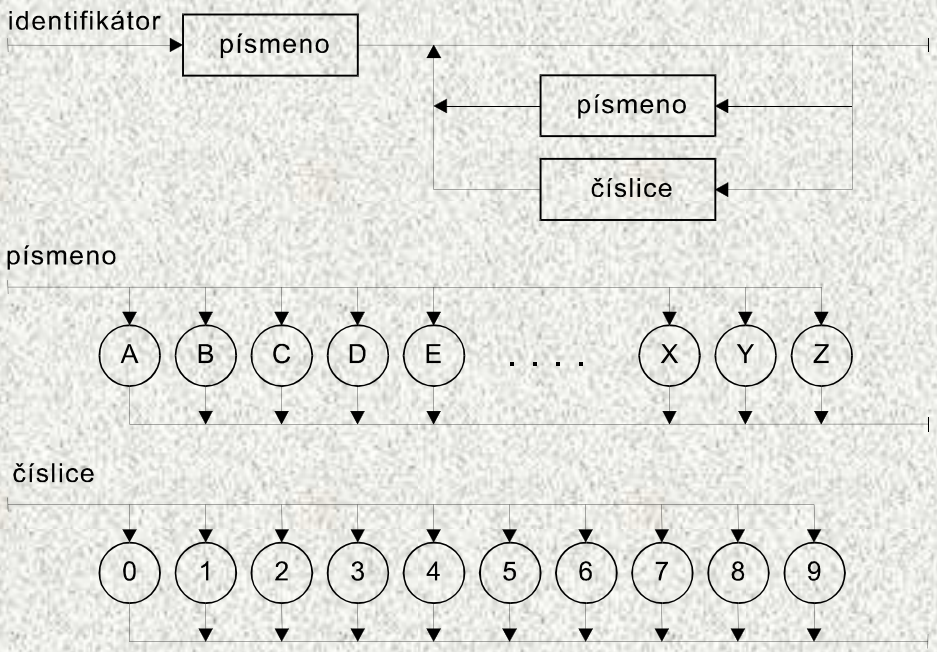
int .. celé číslo, proměnné x,y,d a výsledek budou typu **int**

d--; příkaz, který sníží hodnotu uloženou v proměnné d o jedničku

x%d … zbytek po dělení čísla x číslem d,
!= … není rovno,
|| … nebo

Vlastnosti programovacích jazyků

- Syntaxe
  - souhrn pravidel udávajících přípustné tvary dílčích konstrukcí a celého programu
  - syntaktické diagramy



- Sémantika
  - udává význam jednotlivých konstrukcí

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:

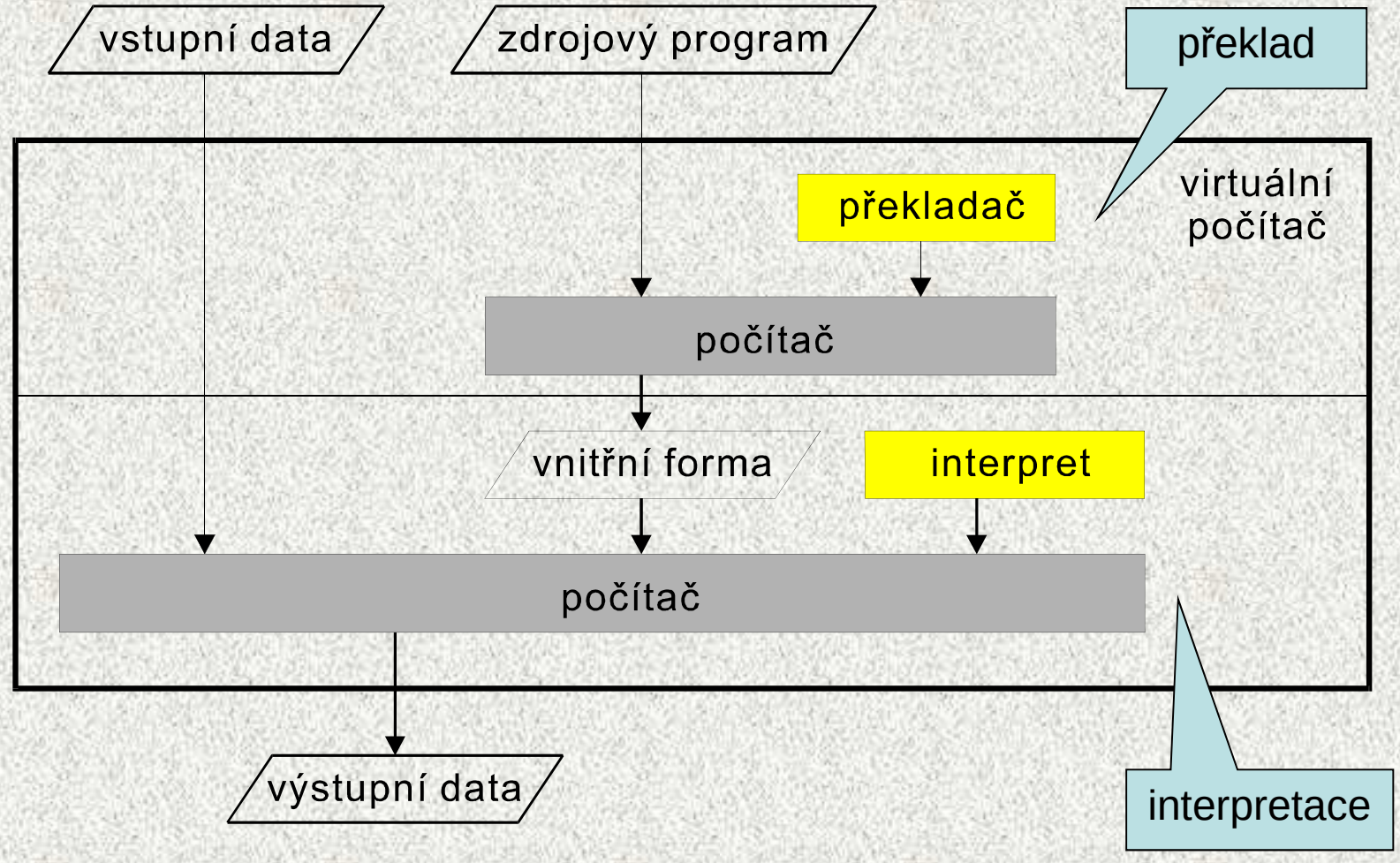
  {*x*} žádný nebo několik výskytů *x*

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

  [*x*] žádný nebo jeden výskyt *x*

Implementace programovacích jazyků

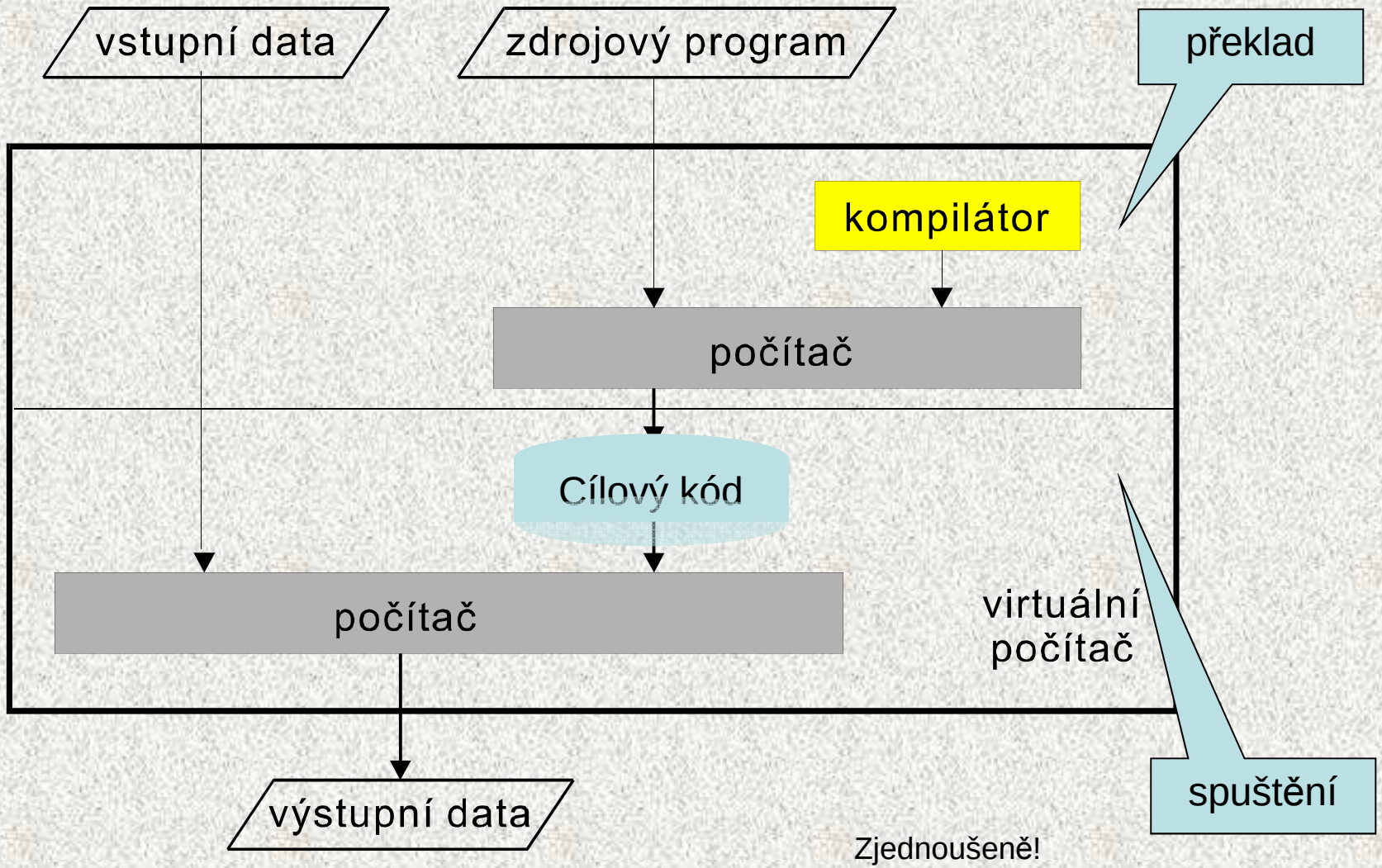
- Interpretační metoda:



Zjednoušeně!

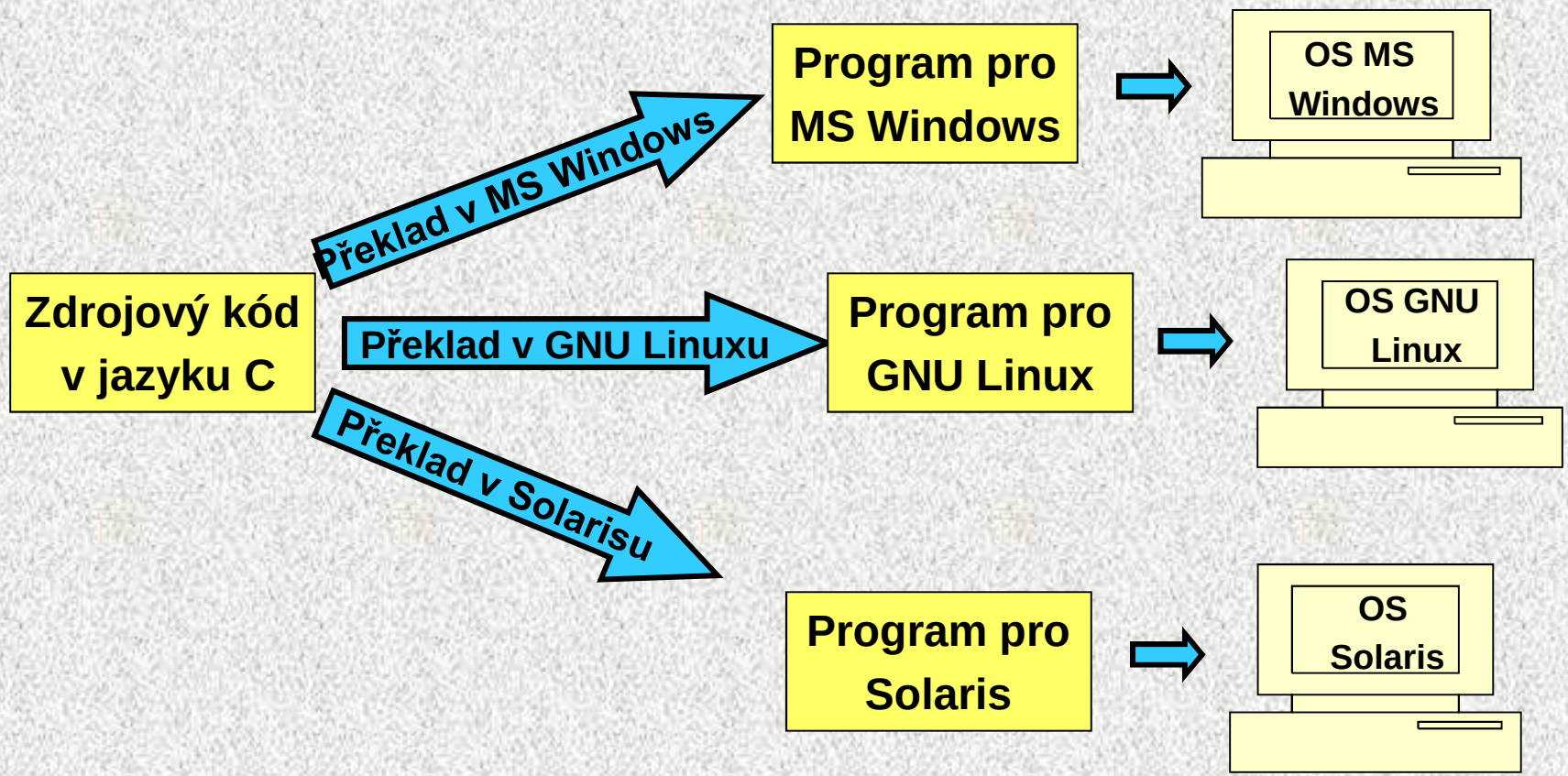
Implementace programovacích jazyků

- Kompilační metoda:

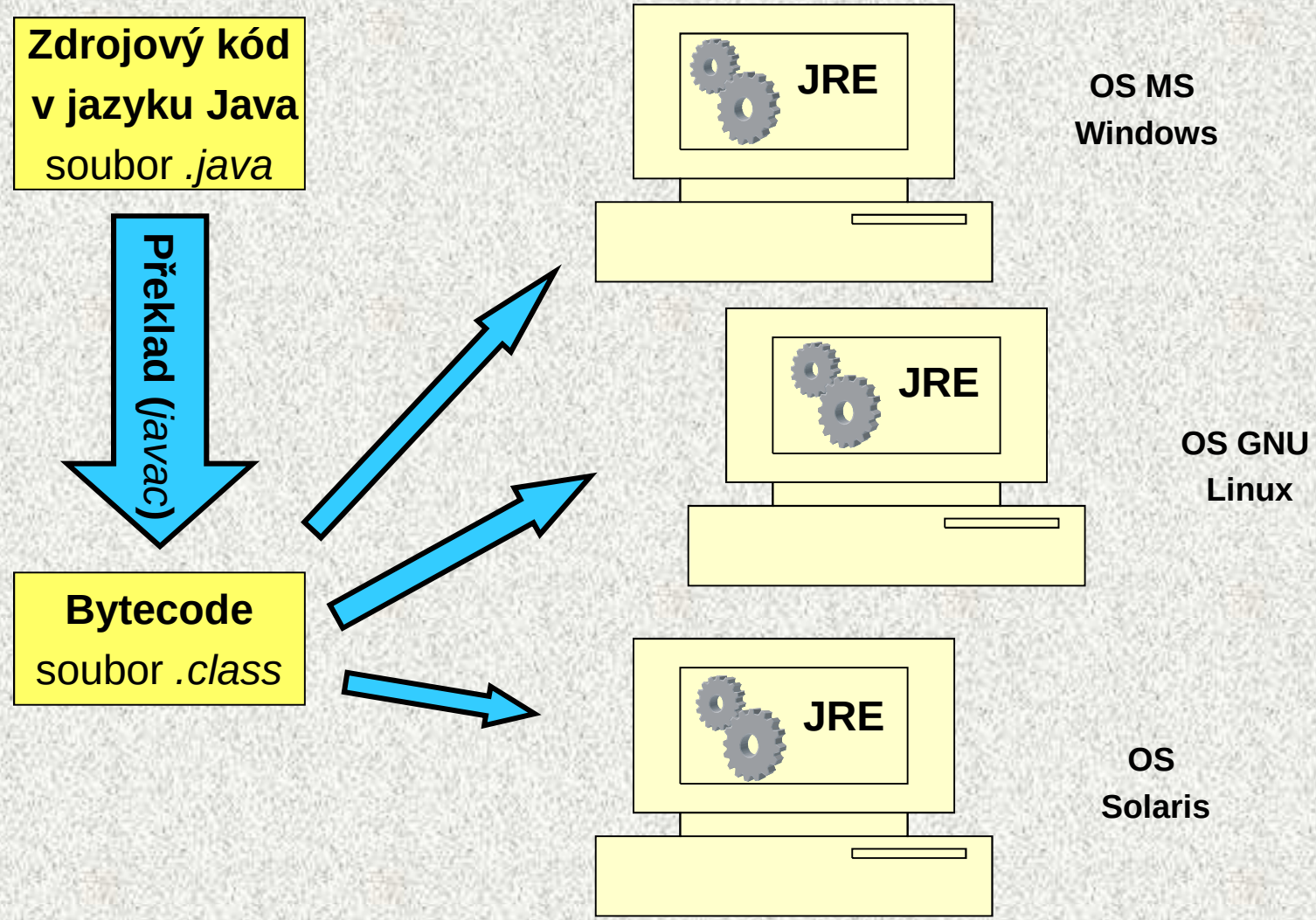


Zjednoušeně!

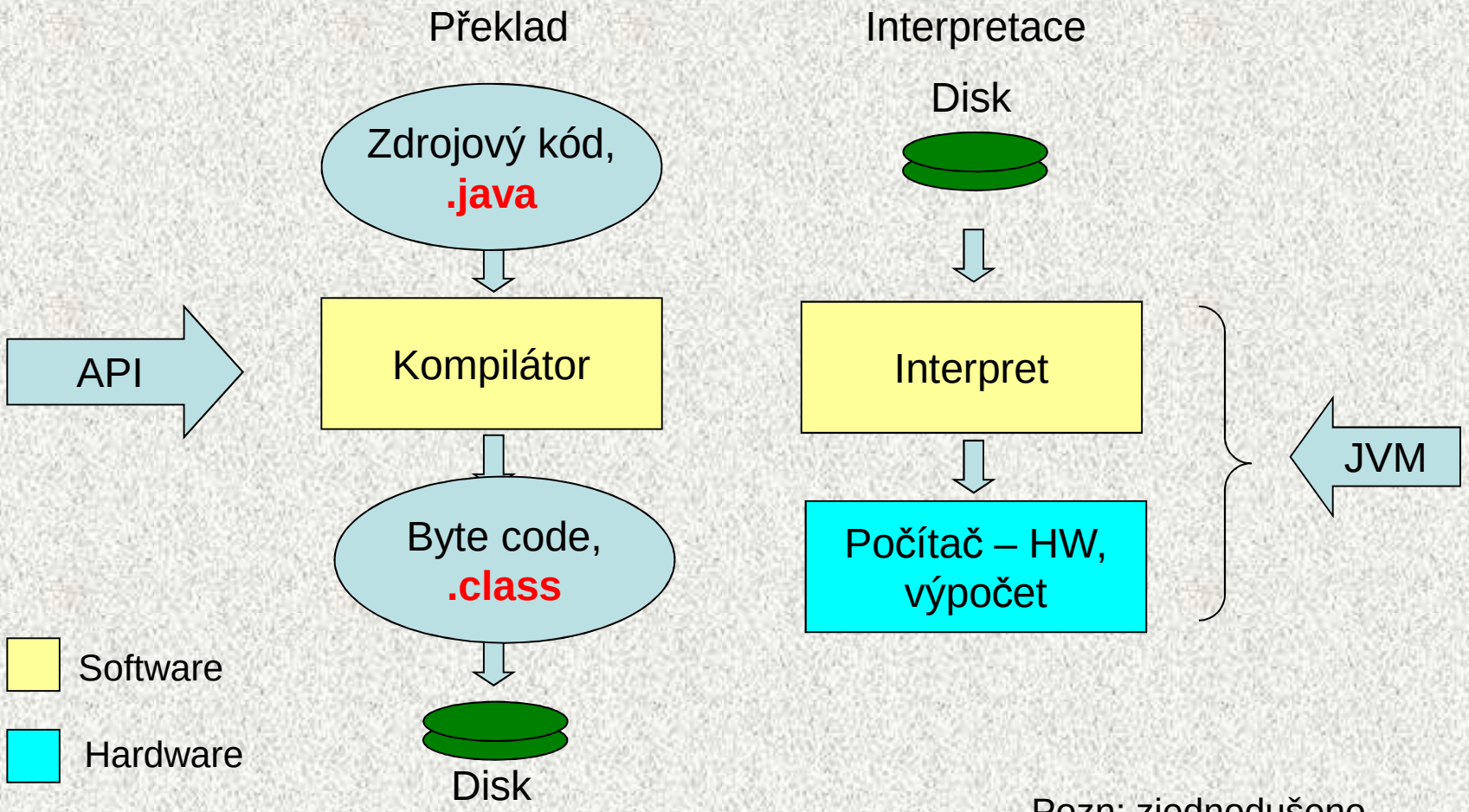
Kompilační metoda - jazyk C, C++



Interpretační metoda - jazyk Java



Java Platforma (JRE) = Java Core API + JVM



Pozn: zjednodušeno

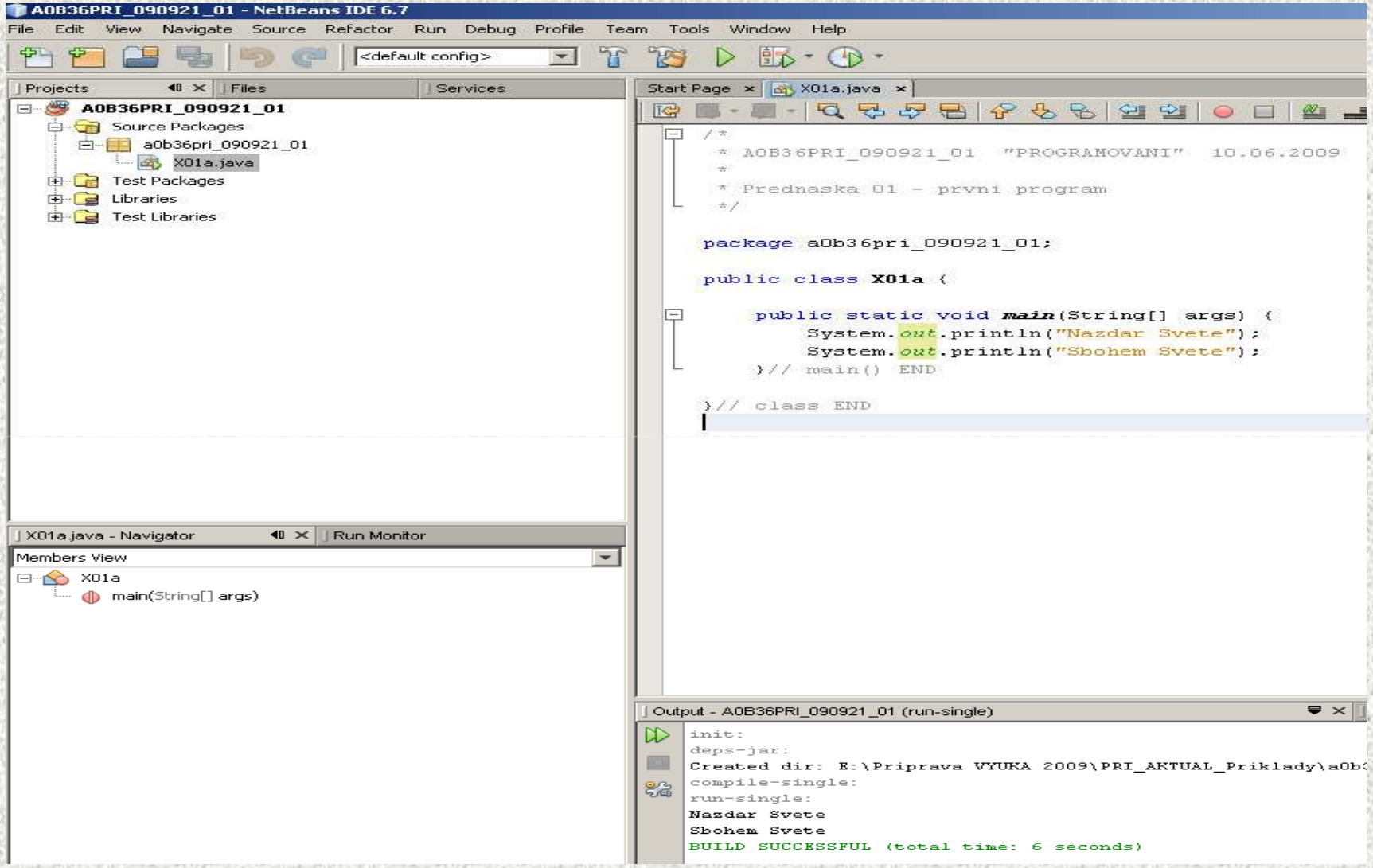
IDE – vývojový nástroj

- Pro vývoj programů se používá vývojový nástroj nazývaný IDE (Integrated Development Environment)
- IDE je v dnešní době k dispozici pro všechny běžné používané programovací jazyky, často od různých výrobců software
- V předmětu A0B36PRI budeme pro vývoj programů v Javě používat IDE NetBeans
- IDE NetBeans je možné zdarma stáhnout na webu a nainstalovat na platformě Windows nebo Unix (viz http://www.netbeans.org)
- NetBeans obsahují (nebo využívají) všechny nezbytné části pro vývoj programu v Javě (textový editor, kompilátor, ladící prostředky a další části)
- S používáním NetBeans se seznámíte na cvičeních

Jazyk JAVA - úvod

- Pro prezentaci, návrh a ověřování algoritmů a výuku základních programovacích technik použijeme jazyk Java
- Proč?
  - jde o vyšší, obecně použitelný programovací jazyk s vysokým stupněm zabezpečení
  - je objektově orientovaný, umožňuje však i klasické procedurální programování, kterému se budeme především věnovat
  - vytvořené programy jsou zcela přenositelné (portable) mezi různými platformami (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 prostředky pro vývoj programů v Javě jsou k dispozici zdarma (viz IDE NetBeans).
    - Studenti si tedy mohou snadno vývojový nástroj instalovat na svých domácích počítačích a mohou se učit programovat i mimo počítačové učebny školy

NetBeans


A0B36PRI_090921_01 - NetBeans IDE 6.7
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Projects
Files
Services
A0B36PRI_090921_01
Source Packages
a0b36pri_090921_01
X01a.java
Test Packages
Libraries
Test Libraries
Start Page
X01a.java
/*
* A0B36PRI_090921_01 "PROGRAMOVANI" 10.06.2009
*
* Prednaska 01 - prvni program
*/
package a0b36pri_090921_01;
public class X01a {
public static void main(String[] args) {
System.out.println("Nazdar Svete");
System.out.println("Sbohem Svete");
}// main() END
}// class END
X01a.java - Navigator
Run Monitor
Members View
X01a
main(String[] args)
Output - A0B36PRI_090921_01 (run-single)
init:
deps-jar:
Created dir: E:\Priprava VYUKA 2009\PRI_AKTUAL_Priklady\a0b
compile-single:
run-single:
Nazdar Svete
Sbohem Svete
BUILD SUCCESSFUL (total time: 6 seconds)

Jazyk JAVA - úvod

- 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*** (JVM – Java Virtual Machine v balíčku JRE Java Runtime Environment) a provede výpočet:

    `Program.class > java > „výpočet”`

Poznámky:

(*) v terminologii firmy Sun to je kompilátor

- program obvykle využívá řadu knihoven (Java Core API), které je třeba mít k dispozici jak při překladu, tak při interpretaci!!!

JAVA – první program

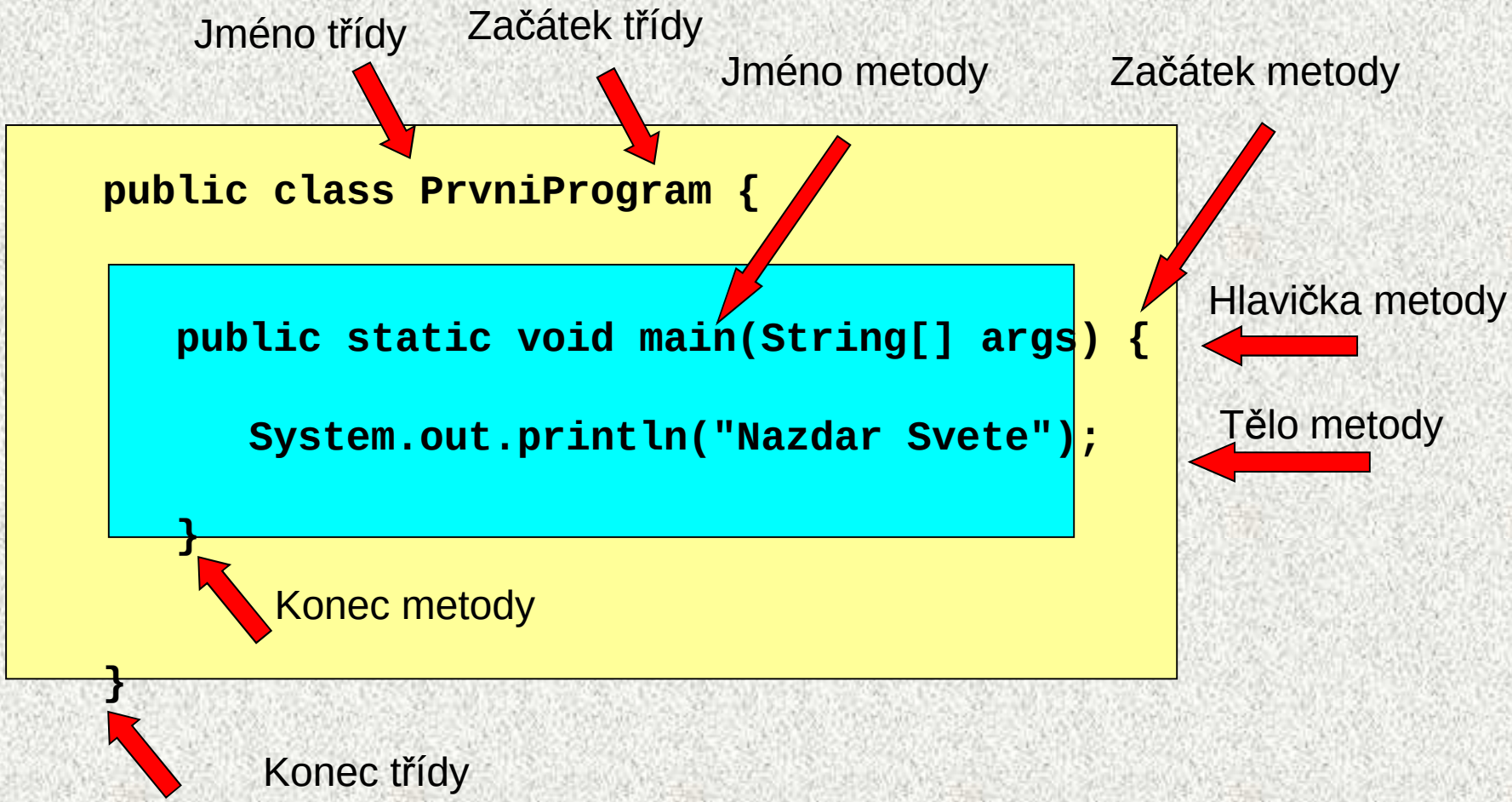
- Příklad: program vypíše daný text na obrazovku:

```
public class PrvniProgram {
  public static void main(String[] args) {
    System.out.println("Nazdar Svete");
  }
}
```

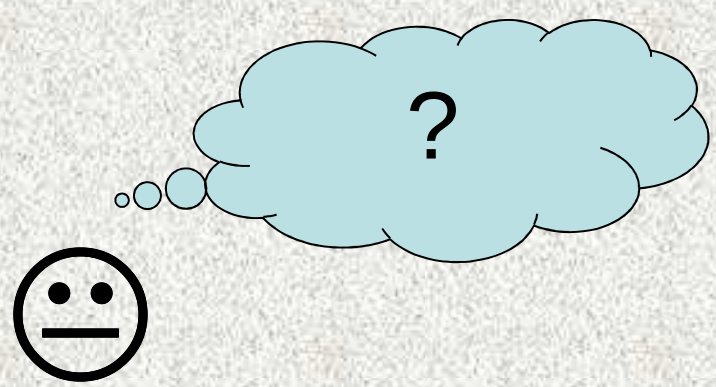
- Po překladu a spuštění se na obrazovku vypíše
  `Nazdar Svete`
- Nejjednodušší zdrojový program – je uložen v jediném souboru. Jméno souboru musí být shodné se jménem třídy (zde PrvniProgram) a přípona (rozšíření) jména souboru je povinná .java (náš program bude tedy uložen v souboru „PrvniProgram.java“)
  - **deklarace veřejné třídy (`public class`),**
  - **hlavní funkce `main` (veřejná statická metoda, public static method)**
- Hlavička funkce funkce `main` ():
  - klíčová slova `public static void` (`void` - procedura)
  - `(String[] args)` specifikace vstupních parametrů
- Konvence: jména tříd se píší s prvním velkým písmenem

JAVA – bloková struktura

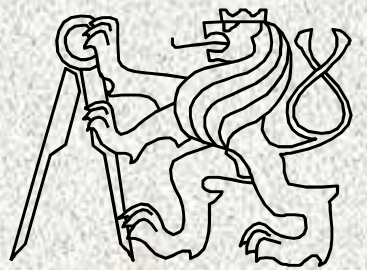
– Program má blokový charakter (blok třídy, blok(y) metod(y))
  • Nejtriviálnější program je tvořen metodou **`main`** ve třídě (třída = program)



Vlastní studium



Programování 1 – úvod
KONEC



A0B36PR1-Programování 1
Fakulta elektrotechnická
České vysoké učení technické