

# Seminář Java

## Příkazy a operátory v Javě

Radek Kočí

Fakulta informačních technologií VUT

Únor 2010

- Příkazy
- Operátory

- volání metody
- příkaz je ukončen středníkem (`;`)
- přiřazovací příkaz (`=`)
- řízení toku programu
- návrat z metody (`return`)

## Přiřazení =

- Na levé straně musí být proměnná.
- Na pravé straně musí být *přiřaditelný* výraz.

## Primitivní typy

- přiřazením se hodnota zduplikuje
- konverze typů (`short` → `int`, `int` → `short`)

## Přiřazení odkazu na objekt

- Proměnné objektového typu obsahují odkazy (reference) na objekty, ne objekty samotné!!!
- přiřazením se duplikuje pouze reference

- `if`
- `while`
- `do – while`
- `for`
- `switch`
- `break, continue`

Příkazy mohou být jednoduché

- `pole[i] = 20;`

nebo složené

- `{ pole[i] = 20; i++; }`

Příkaz (neúplného) větvení `if`

```
if (logický výraz) příkaz
```

- platí-li logický výraz (má hodnotu `true`), provede se příkaz

---

Příkaz úplného větvení `if - else`

```
if (logický výraz)
```

```
    příkaz1
```

```
else
```

```
    příkaz2
```

- platí-li logický výraz (má hodnoty `true`), provede se `příkaz1`
- neplatí-li, provede se `příkaz2`
- větev `else` se nemusí uvádět
- větvení `if -- else` můžeme vnořovat do sebe

# Cyklus s podmínkou na začátku

- Tělo cyklu se provádí tak dlouho, dokud platí podmínka

v těle cyklu je jeden jednoduchý příkaz ...

```
while (podmínka)
    příkaz;
```

... nebo příkaz složený

```
while (podmínka) {
    příkaz1;
    příkaz2;
    ...
}
```

- Tělo cyklu se nemusí provést ani jednou – pokud už hned na začátku podmínka neplatí



- Tělo se provádí dokud platí podmínka (vždy aspoň jednou).
- Relativně málo používaný – je méně přehledný než `while`

```
do {  
    příkaz1;  
    příkaz2;  
    ...  
} while (podmínka);
```

- de-facto jde o rozšíření while, lze jím snadno nahradit

```
for (počáteční operace; vstupní podmínka;  
     příkaz po každém průchodu)  
{  
    příkaz1;  
    příkaz2;  
    ...  
}
```

# Příklad použití "for" cyklu

```
for (int i = 0; i < 10; i++) {  
    System.out.println(i);  
}
```

---

Ekvivalent s while:

```
int i=0;  
while (i < 10) {  
    System.out.println(i);  
    i++;  
}
```

# Vícecestné větvení "switch - case - default"

- Větvení do více možností na základě ordinální hodnoty

```
switch(výraz) {  
    case hodnota1: prikaz1a;  
                  prikaz1b;  
                  break;  
    case hodnota2: prikaz2a;  
                  ...  
                  break;  
    default:      prikazDa;  
                  ...  
}
```

- Je-li výraz roven některé z hodnot, provede se sekvence uvedená za příslušným case.
- Sekvenci obvykle ukončujeme příkazem break, který předá řízení ("skočí") na první příkaz za ukončovací závorkou příkazu switch.

# Příkaz "break"

- Realizuje "násilné" ukončení průchodu cyklem nebo větvením switch
- Syntaxe použití break v cyklu:

```
for (int i = 0; i < a.length; i++) {  
    if(a[i] == 0) {  
        break; // skoci se za konec cyklu  
    }  
}  
if (a[i] == 0) {  
    System.out.println("0 je na pozici "+i);  
} else {  
    System.out.println("0 v poli neni");  
}
```

# Příkaz "continue"

- Používá se v těle cyklu.
- Způsobí přeskočení zbylé části průchodu tělem cyklu.
- Běh pokračuje další iterací.

```
for (int i = 0; i < a.length; i++) {  
    if (a[i] == 5)  
        continue;  
    System.out.println(i);  
}
```

- Aritmetické
- Logické
- Relační
- Bitové
- Operátor podmíněného výrazu ? :
- Relační operátory

- `+`, `-`, `*`, `/` a `%` (zbytek po celočíselném dělení)
- platí podobná pravidla jako v C/C++
  - `int / int ⇒ int`
  - `double / int ⇒ double`
  - `short / int ⇒ int`



- logické součiny (AND):
  - `&` (nepodmíněný - vždy se vyhodnotí oba operandy),
  - `&&` (podmíněný - líné vyhodnocování - druhý operand se vyhodnotí, jen nelze-li o výsledku rozhodnout z hodnoty prvního)
- logické součty (OR):
  - `|` (nepodmíněný - vždy se vyhodnotí oba operandy),
  - `||` (podmíněný - líné vyhodnocování - druhý operand se vyhodnotí, jen nelze-li o výsledku rozhodnout z hodnoty prvního)
- negace (NOT):
  - `!`

## Bitové:

- součin `&`
- součet `|`
- exkluzivní součet (XOR) `^` (znak "stříška")
- negace (bitwise-NOT) `~` (znak "tilda")

## Posuny:

- vlevo `<<` o stanovený počet bitů
- vpravo `>>` o stanovený počet bitů s respektováním znaménka
- vpravo `>>>` o stanovený počet bitů bez respektování znaménka

## Podmíněný výraz

- Jediný ternární operátor
- Platí-li první operand (má hodnotu true)  $\Rightarrow$ 
  - výsledkem je hodnota druhého operandu
  - jinak je výsledkem hodnota třetího operandu
- Typ prvního operandu musí být boolean, typy druhého a třetího musí být přiřaditelné do výsledku.

```
if (a > b)
    c = a - b;
else
    c = b - a;
```

---

```
c = (a > b ? a - b : b - a);
```

# Relační (porovnávací) operátory

- Tyto lze použít na porovnávání primitivních hodnot:
  - `<`, `<=`, `>=`, `>`
- Test na rovnost/nerovnost lze použít na porovnávání primitivních hodnot i objektů:
  - `==`, `!=`
  - pozor na srovnávání floating-points čísel na rovnost: je třeba počítat s chybami zaokrouhlení; místo porovnání na přesnou rovnost raději použijeme jistou toleranci:  
`abs(expected-actual) < delta`
  - pozor na porovnávání objektů: `==` vrací `true` jen při rovnosti odkazů, tj. jsou-li objekty identické!

Základní typy komentářů (podobně jako např. v C/C++)

- *řádkové* od značky `//` do konce řádku
- *blokové* (na libovolném počtu řádků) začínají `/*` pak je text komentáře, končí `*/`
- *dokumentační* (na libovolném počtu řádků) od značky `/**` po značku `*/` Každý další řádek může začínat mezerami či \*, hvězdička se v komentáři neprojeví.

```
// řádkový komentář

/*
   blokový
   (víceřádkový) komentář
*/

/**
   dokumentační
   (víceřádkový) komentář
*/
```