Řídicí struktury: větvení, cykly.

Obsah

Příkazy a řídicí struktury v Javě

- Příkazy v Javě
- Přiřazení
- Volání metod a návrh z nich
- Řídicí příkazy (větvení, cykly)

Příkazy v Javě

Příkazy v Javě

V Javě máme následující příkazy:

- Přiřazovací příkaz = a jeho modifikace (kombinované operátory jako je += apod.)
- Řízení toku programu (větvení, cykly)
- Volání metody
- Návrat z metody - příkaz return
- Příkaz je ukončen středníkem ;
- v Pascalu středník příkazy *odděluje*, v Javě (C/C++) *ukončuje*

Přiřazení

Přiřazení v Javě

Operátor přiřazení:

- Operátor přiřazení = (assignment)
- na levé straně musí být proměnná
- na pravé straně výraz *přiřaditelný* (assignable) do této proměnné
- Rozlišujeme přiřazení primitivních hodnot a odkazů na objekty

Přiřazení primitivní hodnoty

Na pravé straně výraz vracející hodnotu primitivního typu

číslo, logická hodnotu, znak (ale ne např. řetězec)

Na levé straně proměnná téhož typu jako přiřazovaná hodnota nebo typu širšího

např. `int`WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=int]

lze přiřadit do `long`WIKIPEDIA

[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=long]

Při zužujícím přiřazení se také provede konverze, ale může dojít ke ztrátě informace

např. `int` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=int] -> `short` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=short]

Přiřazením primitivní hodnoty se hodnota zduplikuje ("opíše") do proměnné na levé straně.

Přiřazení odkazu na objekt

Konstrukci = [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search==] lze použít i pro přiřazení do objektové proměnné:

```
Zivocich z1 = new Zivocich();
```

[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=Zivocich z1 = new Zivocich();]

Co to udělalo?

1. vytvořilo nový objekt typu `Zivocich` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=Zivocich] (`new Zivocich()` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=new Zivocich()])
2. přiřadilo jej do proměnné `z1` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=z1] typu `Zivocich` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=Zivocich]

Nyní můžeme *odkaz* na tentýž vytvořený objekt znovu přiřadit - do `z2` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=z2]:

```
Zivocich z2 = z1;
```

[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=Zivocich z2 = z1;]

Proměnné `z1` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=z1] a `z2` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=z2] ukazují nyní na stejný objekt typu živočich!!!

Proměnné objektového typu obsahují *odkazy* (reference) na objekty, ne objekty samotné!!!

Volání metod a návrat z nich

Volání metody

Metoda objektu je vlastně procedura/funkce, která realizuje svou činnost primárně s proměnnými objektu.

Volání metody určitého objektu realizujeme:

identifikaceObjektu.názevMetody(skutečné parametry)

- **identifikaceObjektu**, jehož metodu voláme
- **.** (tečka)
- **názevMetody**, jíž nad daným objektem voláme
- v závorách uvedeme *skutečné parametry* volání (záv. může být prázdná, nejsou-li parametry)

Návrat z metody

Návrat z metody se děje:

1. Buďto automaticky posledním příkazem v těle metody
2. nebo explicitně příkazem **return** ***návratováhodnota*** WIKIPEDIA The Free Encyclopedia [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=return]

způsobí ukončení provádění těla metody a návrat, přičemž může být specifikována *návratová hodnota*

typ skutečné návratové hodnoty musí korespondovat s deklarovaným typem návratové hodnoty

Řízení toku uvnitř metod - větvení, cykly

Řízení toku programu v těle metody

Příkaz (neúplného) větvení **if** WIKIPEDIA The Free Encyclopedia

[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=if]

`if (logický výraz) příkaz` WIKIPEDIA The Free Encyclopedia

[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=if (logický výraz) příkaz]

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

Příkaz úplného větvení **if - `else`** WIKIPEDIA The Free Encyclopedia

[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=else]WIKIPEDIA

[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=if -]

```
if (logický výraz)
   příkaz1
else
   příkaz2
```

platí-li `logický výraz` WIKIPEDIA

[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search= logický výraz] (má hodnoty `true`WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=true]),

provede se `příkaz1` WIKIPEDIA

[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search= příkaz1]

neplatí-li, provede se `příkaz2` WIKIPEDIA

[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search= příkaz2]

Větev **else**WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=else] se **nemusí uvádět**

Cyklus s podmínkou na začátku

Tělo cyklu se provádí tak dlouho, **dokud** platí podmínka

obdoba **while**WIKIPEDIA

[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=while] v Pascalu

v těle cyklu je jeden jednoduchý příkaz ...

```
while (podmínka)
   příkaz;
```

... nebo příkaz složený

```
while (podmínka) {
   příkaz1;
   příkaz2;
   příkaz3;
   ...
}
```

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

Doporučení k psaní cyklů/větvení

Větvení, cykly: doporučuji vždy psát se **složeným příkazem v těle** (tj. se složenými závorkami)!!!

jinak hrozí, že se v těle větvení/cyklu z neopatrnosti při editaci objeví něco jiného, než chceme, např.:

```
while (i < a.length)
   System.out.println(a[i]); i++;
```

Provede v cyklu jen ten výpis, inkrementaci ne a program se zacyklí.

Pišme proto vždy takto:

```
while (i < a.length) {
    System.out.println(a[i]); i++;
}
```

Příklad použití "while" cyklu

Dokud nejsou přečteny všechny vstupní argumenty:

```
int i = 0;
while (i < args.length) {
   "přečti argument args[i]"
   i++;
}
```

Dalším příkladem je použití **while**WIKIPEDIA

[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=while] pro realizaci celočíselného dělení se zbytkem:

Příklad: Celočíselné dělení se zbytkem [http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/tomp/ucebnice/rizeni/DeleniOdcitanim.java]

Cyklus s podmínkou na konci

Tělo se provádí **dokud** platí podmínka (vždy aspoň jednou)

obdoba **repeat**WIKIPEDIA

[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=repeat] v Pascalu (podmínka je ovšem *interpretována opačně*)

Relativně málo používaný - je méně přehledný než **while**WIKIPEDIA

[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=while]

Syntaxe:

```
do {
```

```
   příkaz1;
   příkaz2;
   příkaz3;
   ...
} while (podmínka);
```

Příklad použití "do-while" cyklu

Dokud není z klávesnice načtena požadovaná hodnota:

```
String vstup = "";
float cislo;
boolean nacteno; // vytvoř reader ze standardního vstupu
BufferedReader in = new BufferReader(new InputStream(System.in));
// dokud není zadáno číslo, čti
do {
   vstup = in.readLine();
   try {
      cislo = Float.parseFloat(vstup);
      nacteno = true;
   } catch (NumberFormatException nfe) {
      nacteno = false;
   }
} while(!nacteno);
System.out.println("Nacteno cislo "+cislo);
```

Příklad: Načítej, dokud není zadáno číslo [http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/tomp/ucebnice/rizeni/DokudNeniZadano.java]

Cyklus "for"

obecnější než **for** WIKIPEDIA The Free Encyclopedia

[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=for] v Pascalu, podobně jako v C/C++

De-facto jde o rozšíření **while** WIKIPEDIA The Free Encyclopedia

[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=while], lze jím snadno nahradit

Syntaxe:

```
for(počáteční op.; vstupní podm.; příkaz po každém průch.)
   příkaz;
```

anebo (obvyklejší, bezpečnější)

```
for (počáteční op.; vstupní podm.; příkaz po každém průch.) {
```

```
    příkaz1;
    příkaz2;
    příkaz3;
    ...
}
```

Příklad použití "for" cyklu

Provedení určité sekvence určitý počet krát

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

Vypíše na obrazovku deset řádků s čísly postupně 0 až 9

1. Příklad: Pět pozdravů [http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/tomp/ucebnice/rizeni/PetPozdravu.java]
2. Příklad: Výpis prvků pole objektů "for" cyklem [http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/tomp/ucebnice/rizeni/PolozkyForCyklem.java]

Doporučení k psaní `for` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=for] cyklů (1)

Používejte asymetrické intervaly (ostrá a neostrá nerovnost):

- podmínka daná počátečním přiřazením **i = 0** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=i = 0] a inkrementací **i++** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=i++] je *neostrou nerovností*, zatímco
- opakovací podmínka `i < 10` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=i < 10] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=] je *ostrou nerovností* -> i už nesmí hodnoty 10 dosáhnout!

Vytvarujte se složitých příkazů v hlavičce (kulatých závorkách) **for** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=for] cyklu -

- je lepší to napsat podle situace před cyklus nebo až do jeho těla

Doporučení k psaní `for`WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=for] cyklů (2)

Někteří autoři nedoporučují psát deklaraci řídicí proměnné přímo do závorek cyklu

`for (int i = 0; ...`WIKIPEDIA

[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=for (int i = 0; ...]

ale rozepsat takto:

```
int i;
for (i = 0; ...
```

potom je proměnná `i`WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=i] přístupná ("viditelná") i mimo cyklus - za cyklem, což se však ne vždy hodí.

Vícecestné větvení "switch - case - default"

Obdoba pascalského **select - case - else**WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=select - case - else]

Větvení do více možností na základě ordinální hodnoty

Syntaxe:

```
switch(výraz) {
   case hodnota1: prikaz1a;
                  prikaz1b;
                  prikaz1c;
                  ...
                  break;
   case hodnota2: prikaz2a;
                  prikaz2b;
                  ...
                  break;
   default:       prikazDa;
                  prikazDb;
                  ...
}
```

Je-li *výraz* roven některé z hodnot, provede se sekvence uvedená za příslušným **case**WIKIPEDIA

[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=case].

Sekvenci obvykle ukončujeme příkazem **break**WIKIPEDIA

[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=break], který předá řízení ("skočí") na první příkaz za ukončovací závorkou příkazu **switch**WIKIPEDIA

[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=switch].

Příklad: Vícecestné větvení [http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/tomp/ucebnice/rizeni/VicecestneVetveni.java]

Vnořené větvení

Větvení **if - else**WIKIPEDIA

[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=if - else] můžeme samozřejmě vnořovat do sebe:

```
if(podmínka_vnější) {
   if(podmínka_vnitřní_1) {
      ...
   } else {
      ...
   }
} else {
   if(podmínka_vnitřní_2) {
      ...
   } else {
      ...
   }
}
```

Vnořené větvení (2)

Je možné "šetřit" a neuvádět složené závorky, v takovém případě se **else**WIKIPEDIA

[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=else] vztahuje vždy k nejbližšímu neuzavřenému **if**WIKIPEDIA

[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=if], např. znovu předchozí příklad:

```
if(podmínka_vnější)
   if(podmínka_vnitřní1)
      ...
   else // vztahuje se k nejbližšímu if
        // s if (podmínka_vnitřní_1)
```

```
      ...
else // vztahuje se k prvnímu if,
     // protože je v tuto chvíli
     // nejbližší neuzavřené
   if (podmínka_vnitřní_2)
      ...
   else // vztahuje se k if (podmínka_vnitřní_2)
      ...
```

Tak jako u cyklů - tento způsob zápisu nelze v žádném případě doporučit!!!

Příklad: Vnořené větvení [http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/tomp/ucebnice/rizeni/VnoreneVetveni.java]

Řetězené "if - else if - else"

Někdy rozvíjíme pouze druhou (negativní) větev:

```
if (podmínka1) {
   ...
} else if (podmínka2) {
   ...
} else if (podmínka3) {
   ...
} else {
   ...
}
```

Neplatí-li podmínka1, testuje se podmínka2, neplatí-li, pak podmínka3...

neplatí-li žádná, provede se příkaz za posledním - samostatným - **else**WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=else].

Opět je dobré všude psát složené závorky!!!

Příklad: Řetězené if [http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/tomp/ucebnice/rizeni/VicecestneVetveniIf.java]

Příkazy "break"

Realizuje "násilné" ukončení průchodu cyklem nebo větvením **switch**WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=switch]

Syntaxe použití `break`WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=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("Nasli jsme 0 na pozici "+i);
} else {
    System.out.println("0 v poli neni");
}
```

použití u `switch` WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=switch] jsme již viděli„Vícecestné větvení "switch - case - default"“

Příkaz "continue"

Používá se v těle cyklu.

Způsobí přeskočení zbylé části průchodu tělem cyklu

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

Výše uvedený příklad vypíše čísla 1 WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=1], 2 WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=2], 3 WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=3], 4 WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=4], 6 WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=6], 7 WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=7], 8 WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=8], 9 WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=9], nevypíše hodnotu 5 WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=5].

Příklad: Řízení průchodu cyklem pomocí "break" a "continue" [http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/tomp/ucebnice/rizeni/BreakContinue.java]

"break" a "continue" s návěštím

Umožní ještě jemnější řízení průchodu vnořenými cykly:

- pomocí návěští můžeme naznačit, který cyklus má být příkazem **break**WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=break] přerušen nebo
- tělo kterého cyklu má být přeskočeno příkazem **continue**WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=continue].

Příklad: Návěští [http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/tomp/ucebnice/rizeni/Navesti.java]

Doporučení k příkazům break a continue

Poznámka

Raději NEPOUŽÍVAT, ale jsou menším zlem než by bylo **goto**WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=goto] (kdyby v Javě existovalo...), protože nepředávají řízení dále než za konec struktury (cyklu, větvení).

Poznámka

Toto však již neplatí pro **break**WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=break] a **continue**WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=continue] na návěští!

Poznámka

Poměrně často se používá **break**WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=break] při sekvenčním vyhledávání prvku.