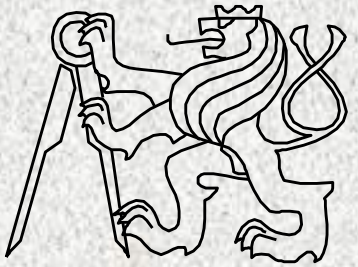
Jak v Javě - řídící konstrukce



BD6B36PJV 001

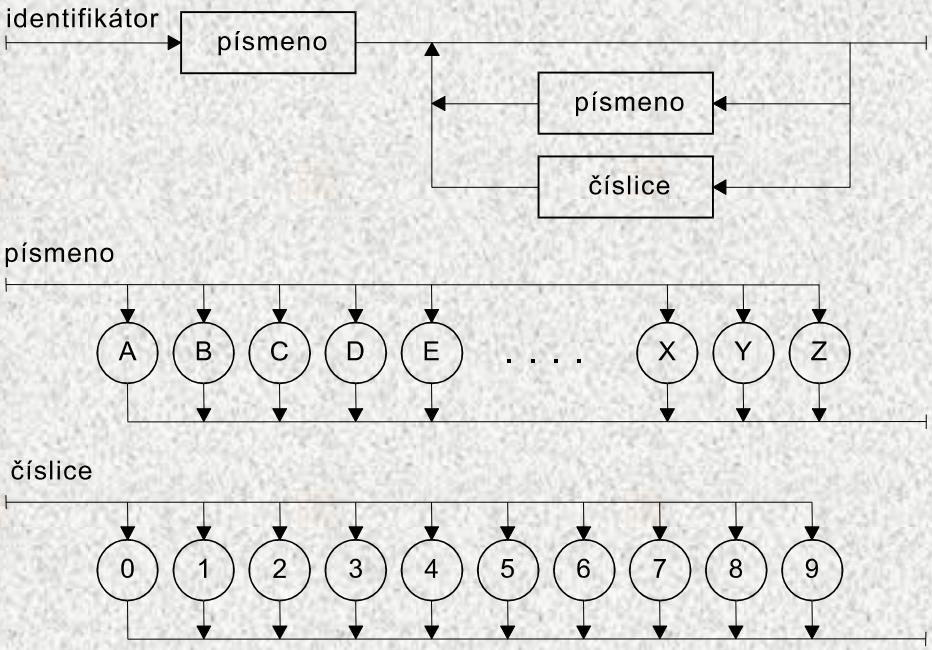
Fakulta elektrotechnická
České vysoké učení technické

Obsah

- Syntaxe a sémantika programovacích jazyků
- Řídící struktury
  - Posloupnost
  - Větvení, typy
  - Cykly, typy
  - **`switch`**
- Continue, break

Vlastnosti programovacích jazyků

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



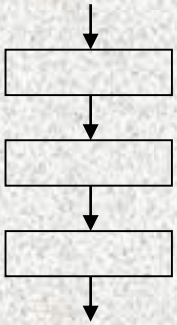
- Sémantika
  - udává význam jednotlivých konstrukcí

Řídicí struktury

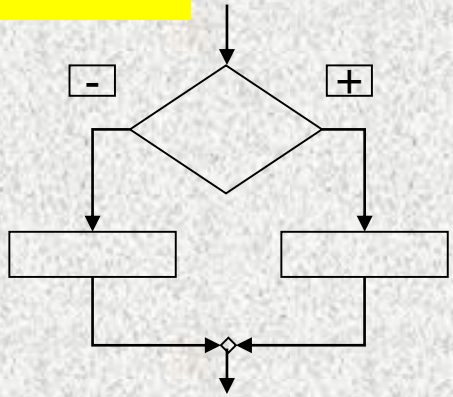
- Řídicí struktura je programová konstrukce, která se skládá z dílčích příkazů a předepisuje pro ně způsob provedení
- Tři druhy řídicích struktur:
  1. *posloupnost*, předepisující postupné provedení dílčích příkazů
  2. *větvení*, předepisující provedení dílčích příkazů v závislosti na splnění určité podmínky
  3. *cyklus*, předepisující opakované provedení dílčích příkazů v závislosti na splnění určité podmínky

Typy řídicích struktur

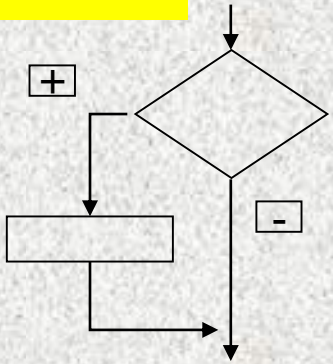
sekvence



if

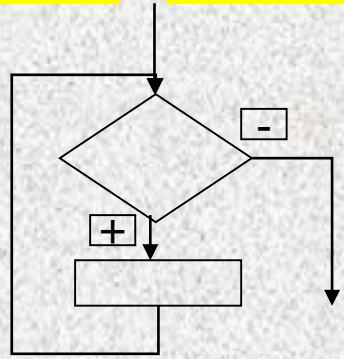


if

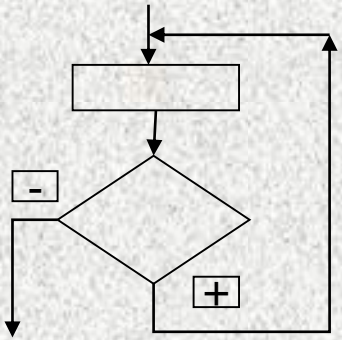


while

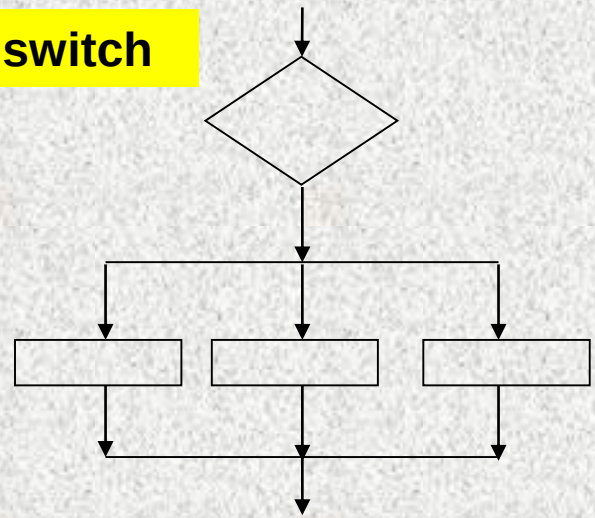
for



do



switch



Řídicí struktury

- Budeme používat následující složené příkazy:
  1. složený příkaz nebo blok pro posloupnost
  2. příkaz **`if`** pro větvení
  3. příkazy **`while`**, **`do`** *nebo* **`for`** pro cyklus

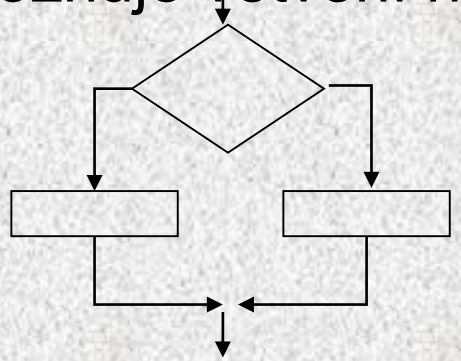
- Řídicí struktury mají obvykle formu strukturovaných příkazů

- Další strukturované příkazy jazyka Java:
  - Složený příkaz: **{** <posloupnost příkazů> **}**
  - Blok: **{** <posloupnost deklarací a příkazů>**}**

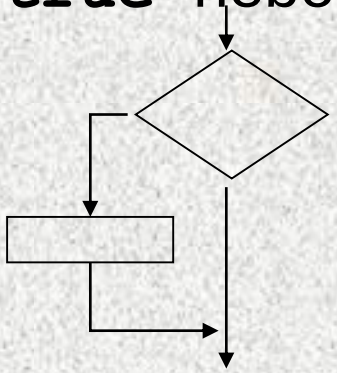
Pozn.:*Deklarace jsou v bloku lokální, tzn. neplatí vně bloku*

Příkaz `if`

- Příkaz **`if`** (podmíněný příkaz) umožňuje větvení na základě podmínky
- Má dva tvary:
  - if (*podmínka*) *příkaz*$_1$ else *příkaz*$_2$
  - if (*podmínka*) *příkaz*$_1$



kde *podmínka* je logický výraz
(výraz, jehož hodnota je typu **`boolean`**, tj. **`true`** nebo **`false`**)



Příkaz if

- Jestliže v případě splnění či nesplnění podmínky má být provedeno více příkazů, je třeba z nich vytvořit složený příkaz nebo blok
- Příklad: jestliže *x* < *y*, vyměňte hodnoty těchto proměnných
- Špatné řešení:

```
if (x < y)
   pom = x; // provede se pro x<y
   x = y; // provede se vždy !!!
   y = pom; // a co toto?
```

Správně

```
if (x < y)
{
    pom = x;
    x = y;
    y = pom;
}
```

Příkaz if

- Příklad: do *min* uložte menší z čísel *x* a *y* a do *max* uložte větší z čísel
- Špatné řešení:

```
if (x < y)
  min = x;
  max = y;
else
  min = y;
  max = x;
```

Správně

```
if (x < y) {
    min = x;
    max = y;
} else {
    min = y;
    max = x;
}
```

Příkaz if

- Do příkazu ***if*** lze vnořit libovolný příkaz, tedy i podmíněný příkaz
- Příklad: do *s* uložte –1, 0 nebo 1 podle toho, zda *x* je menší než nula, rovno nule nebo větší než nula

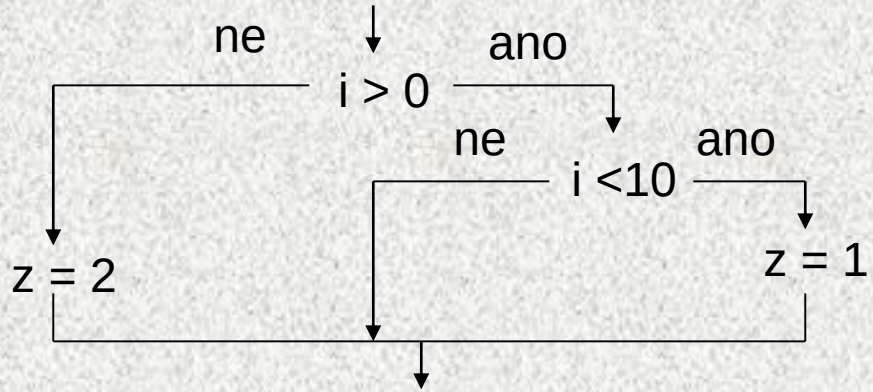
```
if (x < 0)s = -1;
      else if (x == 0)s = 0;
            else s = 1;
```

- Příklad: do *max* uložte největší z čísel *x*, *y* a *z*

```
if (x > y)
    if (x > z) max = x; else max = z;
else
    if (y > z) max = y; else max = z;
```

Příkaz if

- Pozor na vnoření neúplného *if* do úplného *if*
- Příklad: zapište příkazem *if* následující větvení:



- Špatně:

```
if (i > 0)
   if (i < 10) z = 1;
          else z = 2;
```

Správně

```
if (i > 0) {
   if (i < 10) z = 1;
} else z = 2;
```

BMI – body mass index

- **Index určující hranici mezi obezitou a nadváhou:**
- **BMI= váha/výška^2**

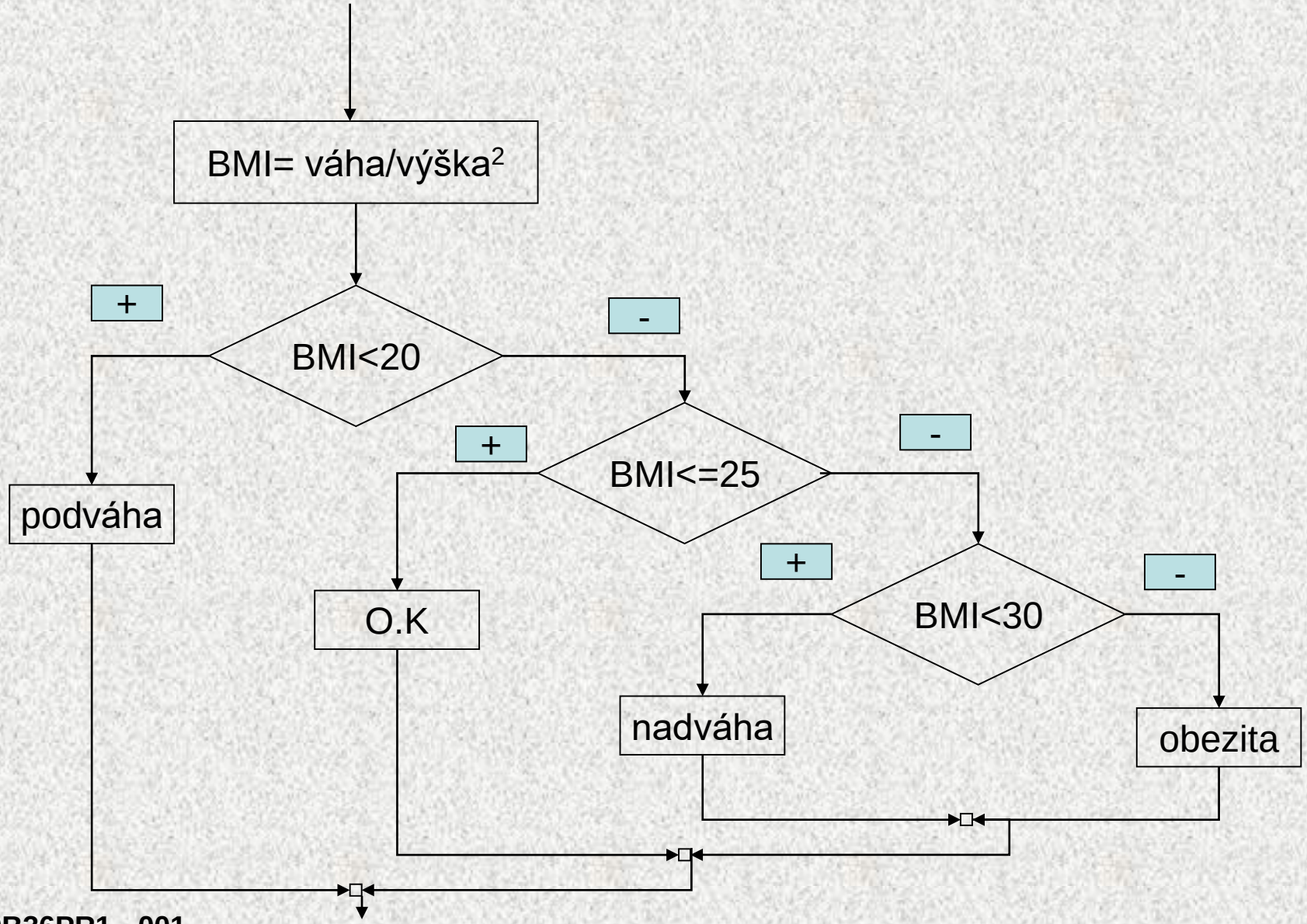
BMI>=30

25<=BMI<30

20<=BMI<25

BMI<20

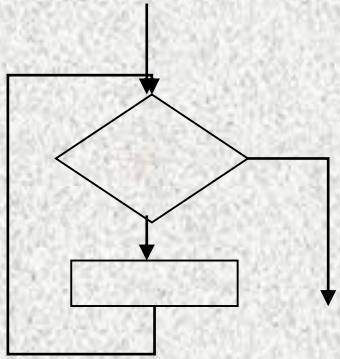
BMI


BMI= váha/výška2
BMI<20
+
-
podváha
BMI<=25
+
-
O.K
BMI<30
+
-
nadváha
obezita

BMI

```
public class BodyMassIndex{
 public static void main(String[] args){
   Scanner sc = new Scanner(System.in);
   System.out.println("Vas Body mass index");
   System.out.print("Vaha (kg): ");
   double vaha = sc.nextDouble();
   System.out.print("Vyska (cm): ");
   double vyska = sc.nextDouble()/100;
   double bmi = vaha/(vyska*vyska);
   System.out.printf("BMI:    %6.3f %n  ",  bmi);
   if (bmi < 20)System.out.println("stihla");
     else if(bmi <= 25) System.out.println("Vse OK");
       else if(bmi < 30) System.out.println("Nadvaha!");
         else System.out.println("Obezita!!");
}
}
```

Příkaz while



- Základní příkaz cyklu, který má tvar

  while (*podmínka*) *příkaz*

- Příklad:

```
q = x;
while (q>=y) q = q-y;
```

(jsou-li hodnotami proměnných *x* a *y* přirozená čísla, co je hodnotou proměnné *q* po skončení uvedeného cyklu?)

Vyzkoušejte pro x = 10, y = 3.

Příkaz while

- Má-li se opakovaně provádět více příkazů, musí tvořit složený příkaz
- Příklad:

```
q = x;
p = 0;
while (q>=y) {
  q = q-y;
  p = p+1;
}
```

(jsou-li hodnotami proměnných *x* a *y* přirozená čísla, co je hodnotou proměnných *p* a *q* po skončení uvedeného cyklu?)

Vyzkoušejte pro x = 10, y = 3.

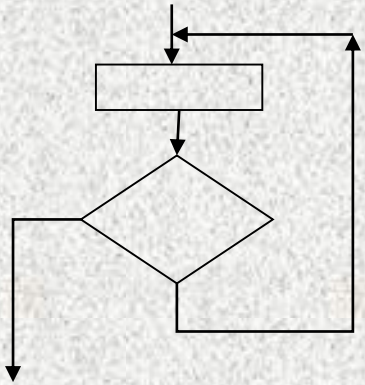
Příklad

Výpočet faktoriálu přirozeného čísla *n* (*n*! = 1 × 2 × ... × *n*)

```
public class Faktorial {
  public static void main(String[] args) {
    System.out.println("zadejte přirozené číslo");
    int n = sc.nextInt();
    if (n<1) {
      System.out.println(n + " není přirozené číslo");
      System.exit(0);
    }
    int i = 1;  int f = 1;
    while (i<n) {
      i = i+1;
      f = f * i;
    }
    System.out.println (n + "! = " + f);
  }}
```

Příkaz do

- Příkaz cyklu *do* se od příkazu *while* liší v tom, že podmínka se testuje až za tělem cyklu
- Tvar příkazu:
  - do *příkaz* while (*podmínka*);
- Vnořeným příkazem je nejčastěji složený příkaz
- Příklad (faktoriál):



```
f = 1; i = 0;
do {
   i = i+1;
   f = f*i;
} while (i<n);
```

- Poznámka k sémantice: příkaz *do* provede tělo cyklu alespoň jednou, nelze jej tedy použít v případě, kdy lze očekávat ani jedno provedení těla cyklu

Příkaz for

- Cyklus je často řízen proměnnou, pro kterou je stanoveno:
  - jaká je počáteční hodnota
  - jaká je koncová hodnota
  - jak změnit hodnotu proměnné po každém provedení těla cyklu
- Příklad:

```
f = 1; i = 1;  // počáteční hodnota řídicí proměnné
while (i<=n) { // podmínka určující koncovou hodnotu
  f = f*i;     // tělo cyklu
  i = i+1;     // změna řídicí proměnné
}
```

- Cykly tohoto druhu lze zkráceně předepsat příkazem *for*:

```
f = 1;
for (i=1; i<=n; i=i+1) f=f*i;
```

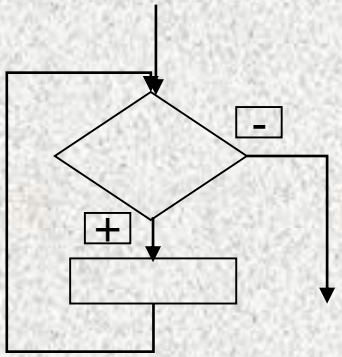
Příkaz for

- Tvar příkazu *for:*

```
for ( inicializace ; podmínka ; změna ) příkaz
```

- Provedení příkazu *for* :

```
inicializace;
while (podmínka) {
  příkaz
  změna;
}
```



- Změnu řídicí proměnné přičtením resp. odečtením 1 lze zkráceně předepsat pomocí operátoru inkrementace resp. dekrementace:
  - **`x++`** ***x*** **se zvětší o 1**
  - **`x--`** ***x*** **se zmenší o 1**

Příkaz for

Příklad:

```
f = 1;
for (i=1; i<=n; i++) f=f*i;
```

Příklad 2a

```
for (int i=1; i<10; i++) {
  System.out.println(i);
}
```

Příklad 2a

```
for (int i=1; i<10; ++i) {
   // místo i++ je zde ++i, jak se změní výstup?
  System.out.println(i);
}
```

Příkaz `continue`

- Příkazy *while* a *for* testují ukončení cyklu před provedením těla cyklu
- Příkaz *do* testuje ukončení cyklu po provedení těla cyklu
- Někdy je třeba ukončit cyklus v nějakém místě uvnitř těla cyklu (které je v tom případě tvořeno složeným příkazem)
- Příkaz *continue* předepisuje předčasné ukončení průchodu těla cyklu
- Příklad:

```
for (int i=1; i<=100; i++) {
  if (i%10==0) continue;
  System.out.println(i);
}
```

- příkaz vypíše čísla od 1 do 100 s výjimkou dělitelných 10

Příkaz break

- Příkaz *break* vnořený do podmíněného příkazu ukončí předčasně příkaz, schematicky:

```
while (...) {
   ...
   if (ukončit) break;
   ...
}
```

- Příkaz **break** předepisuje předčasné ukončení těla cyklu
- Příklad:

```
for (int i=1; i<=100; i++) {
   if (i%10==0) break;
   System.out.println(i);
}
```

- příkaz vypíše čísla od 1 do 9

Příkaz break a continue – příklad

```
public class PrikazContinueBreak {
    public static void main(String[] args ) {

  for (int i=1; i<=30; i++) {
  if (i%10==0) continue;
  System.out.println("hodnota i = " + i);
}

  for (int i=1; i<=30; i++) {
  if (i%10==0) break;
   System.out.println("hodnota i = " + i);
}
```

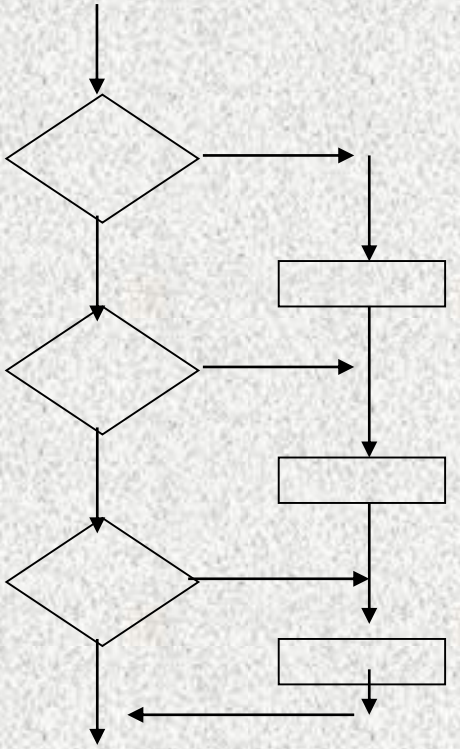
Příkaz switch

- Příkaz *switch* (přepínač) umožňuje větvení do více větví na základě různých hodnot výrazu (nejčastěji typu *int* nebo *char*)
- Základní tvar příkazu:

```
switch (výraz) {
   case konstanta1 : příkazy1 ;
   case konstanta2 : příkazy2 ;
   ...
   case konstantan : příkazyn ;
   default : příkazydef ;
}
```

kde *konstanty* jsou téhož typu, jako *výraz*

příkazy jsou posloupnosti příkazů

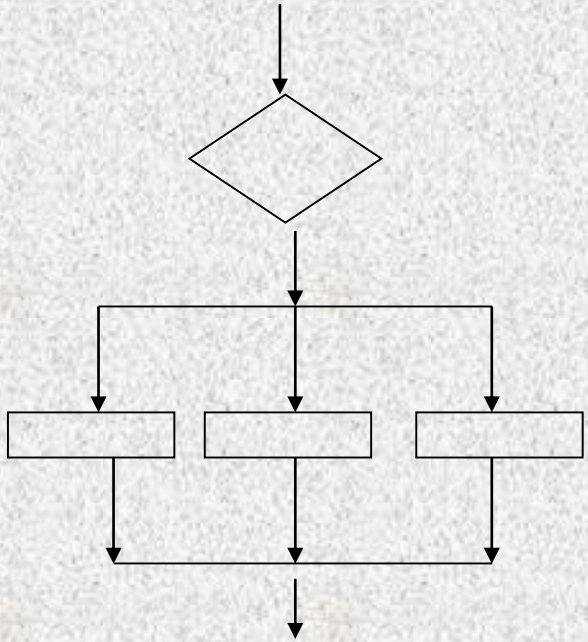


- Sémantika (zjednodušeně):
  - vypočte se hodnota *výrazu* a pak se provedou ty *příkazy*, které jsou označeny *konstantou* označující stejnou hodnotu a příkazy další
  - není-li žádná větev označena hodnotou výrazu, provedou se *příkazy*$_{\text{def}}$

Příkaz switch

- Příkaz *switch* (přepínač) umožňuje větvení do více větví na základě různých hodnot výrazu (nejčastěji typu *int* nebo *char*)
- Základní tvar příkazu:

```
switch (výraz) {
   case konstanta1 : příkazy1 ; break;
   case konstanta2 : příkazy2 ; break;
   ...
   case konstantan : příkazyn ; break;
   default : příkazydef ;
}
```



kde *konstanty* jsou téhož typu, jako *výraz*

příkazy jsou posloupnosti příkazů

- Sémantika (zjednodušeně):
  - vypočte se hodnota *výrazu* a pak se provedou ty *příkazy*, které jsou označeny *konstantou* označující stejnou hodnotu
  - není-li žádná větev označena hodnotou výrazu, provedou se *příkazy*$_{\text{def}}$

Příkaz switch

```
public class PrikazSwitch {):
    switch (n) {
      case 1: System.out.print ("*"); break;
      case 2: System.out.print ("**"); break;
      case 3: System.out.print ("***"); break;
      case 4: System.out.print ("****"); break;
      default: System.out.print ("---");
    }
```

```
***
```

- Příkaz *break* dynamicky ukončuje větev
- Pokud jej neuvedeme, pokračuje se v provádění další větve !!!
- Příklad: co se vypíše, má-li *n* hodnotu 3 a příkaz *switch* zapíšeme takto:

```
    switch (n) {
      case 1: System.out.print ("*");
      case 2: System.out.print ("**");
      case 3: System.out.print ("***");
      case 4: System.out.print ("****");
      default: System.out.print ("---");
    }
```

```
***
****
---
```

Příklad – den v roce

Program pro výpočet pořadového čísla dne v roce

```
public class Den {
public static void main(String[] args) {
        System.out.println("zadejte den, měsíc a rok");
        int den = sc.nextInt();
        int mesic = sc.nextInt();
        int rok = sc.nextInt();
        int n = 0;
        switch (mesic) {
          case 1: n = den; break;
          case 2: n = 31+den; break;
          case 3: n = 59+den; break;
          case 4: n = 90+den; break;
          case 5: n = 120+den; break;
          case 6: n = 151+den; break;
          ...
          case 12: n = 334+den; break;
        }
    if (mesic>2 && rok%4==0 && (rok%100!=0 || rok%1000==0))
          n = n+1;
    Sys...print (den+"."+mesic+"."+rok+" je "+n+". den v roce");
      }
    }
```

Příkaz **`for`** – detaily I

Nevhodná řešení :

```
for (int i=1; i<=n; i++) f=f*i;//Systém.out.print(i); nezná i!

int i;
for (i=1; i<=n; i++) f=f*i; Systém.out.print(i); // i?

int i=0;
for (; i<=n; i++) f=f*i;          //nevhodně mimo

int i=0;
for (; i<=n;) f=f*i++;            // inkrementace mimo

int i=0;
for (i=1; i<=n; f=f*i) i++;       // přehozené
```

Ještě k příkazu `for` II

```
int i=0;
for (; i<=n; f=f*i++);              // smíšené

int i=1;
for (; ;) {
if (i>n) break; f=f*i++;            // nesmyslné
 }


int i=1;
for (; i<=n; f=f*i, i++);           // nesmyslné
```