A0B36PRI - PROGRAMOVÁNÏ

# Řízení běhu programu, řídící struktury

Jazyk JAVA

České vysoké učení technické Fakulta elektrotechnická

v1.01

# Obsah

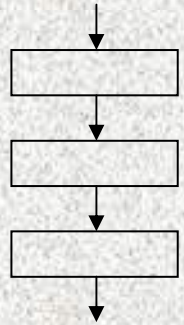
- Řídicí struktury
- Složený příkaz, blok
- Typy řídicích struktur
- Podmíněný příkaz ***`if`***
- Cykly
- Příkaz ***`while`***
- Příkaz ***`do while`***
- Příkaz ***`for`***
- Zpracování posloupností
- Modifikace cyklu – vynechaná část výpočtu
- Příkaz ***`continue`***
- Modifikace cyklu – předčasné ukončení výpočtu
- Příkaz ***`break`***
- Konečnost cyklů
- Programový přepínač
- Příkaz ***`switch`***

# Řídicí struktury

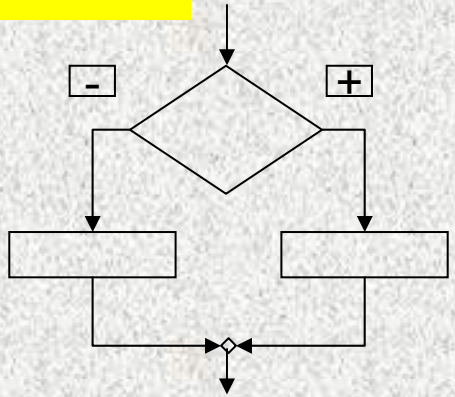
- Řídicí struktura je programová konstrukce, která se skládá z dílčích příkazů a předepisuje pro ně způsob provedení
- Tři druhy řídicích struktur:
  1. *posloupnost*, předepisující postupné provedení dílčích příkazů
  2. *větvení*, předepisující provedení dílčích příkazů v závislosti na splnění určité podmínky
  3. *cyklus*, předepisující opakované provedení dílčích příkazů v závislosti na splnění určité podmínky
- Budeme používat následující složené příkazy:
  1. *složený příkaz* nebo *blok* pro posloupnost
  2. příkaz ***`if`*** pro větvení
  3. příkazy ***`while`***, ***`do while`*** 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*

# Typy řídicích struktur

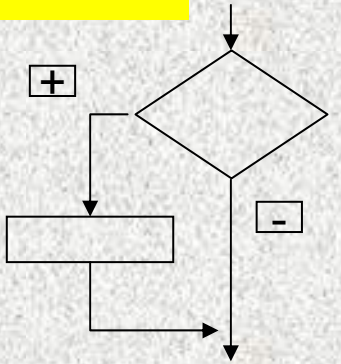
**sekvence**



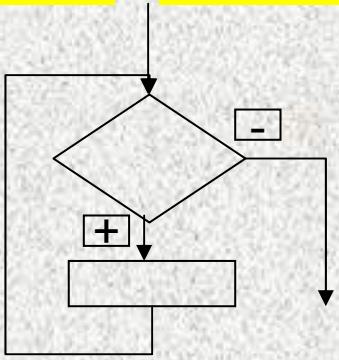
**if**




**if**

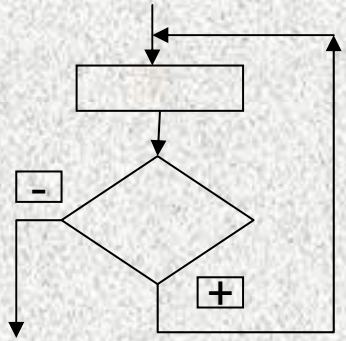



**while**

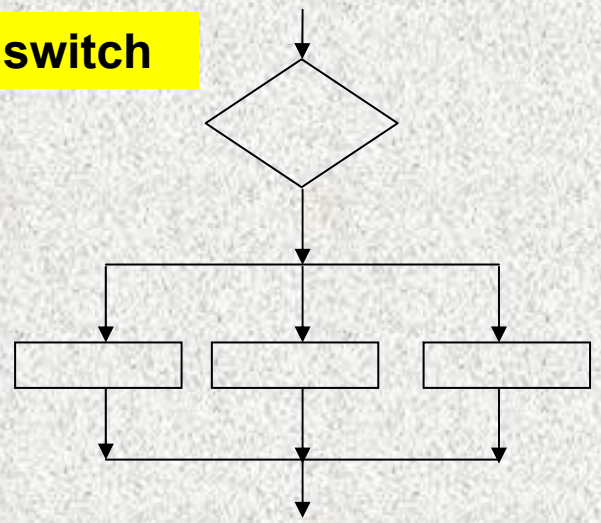



**for**

**do**

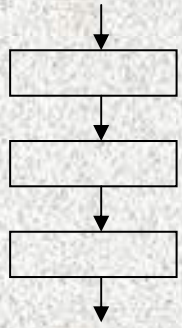



**switch**



# Složený příkaz, blok

```
// Slozeny prikaz
{
x = y *3;
x = x - 3;
y = 234 + x;
}

// Blok
{
x = y*4;
…
   {                 // Zacatek bloku
   int p,q;          // p,q lokalni v bloku
   p = x + y;
   q = p - 3;
   y = p - q;
   System.out.println("q="+q);
   }                 // Konec bloku
// p,q - zde jiz nezname
x = y - 100;
}
```



# Podmíněný příkaz ***if***

- Příkaz **`if`** (podmíněný příkaz) umožňuje větvení na základě podmínky
- Má dva tvary:

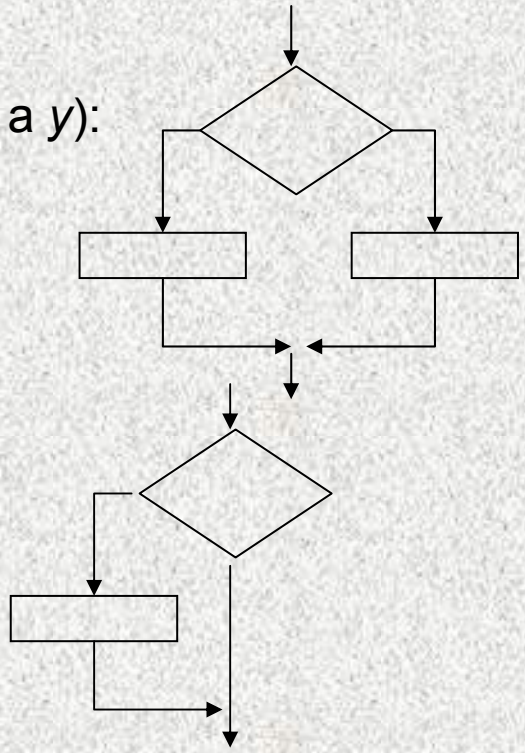
  **`if (`*`podmínka`*`)`** $\mathbf{\mathit{příkaz}_1}$ **`else`** $\mathbf{\mathit{příkaz}_2}$

  **`if (`*`podmínka`*`)`** $\mathbf{\mathit{příkaz}_1}$

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

- Příklad (do *min* uložit a pak vypsat menší z hodnot *x* a *y*):

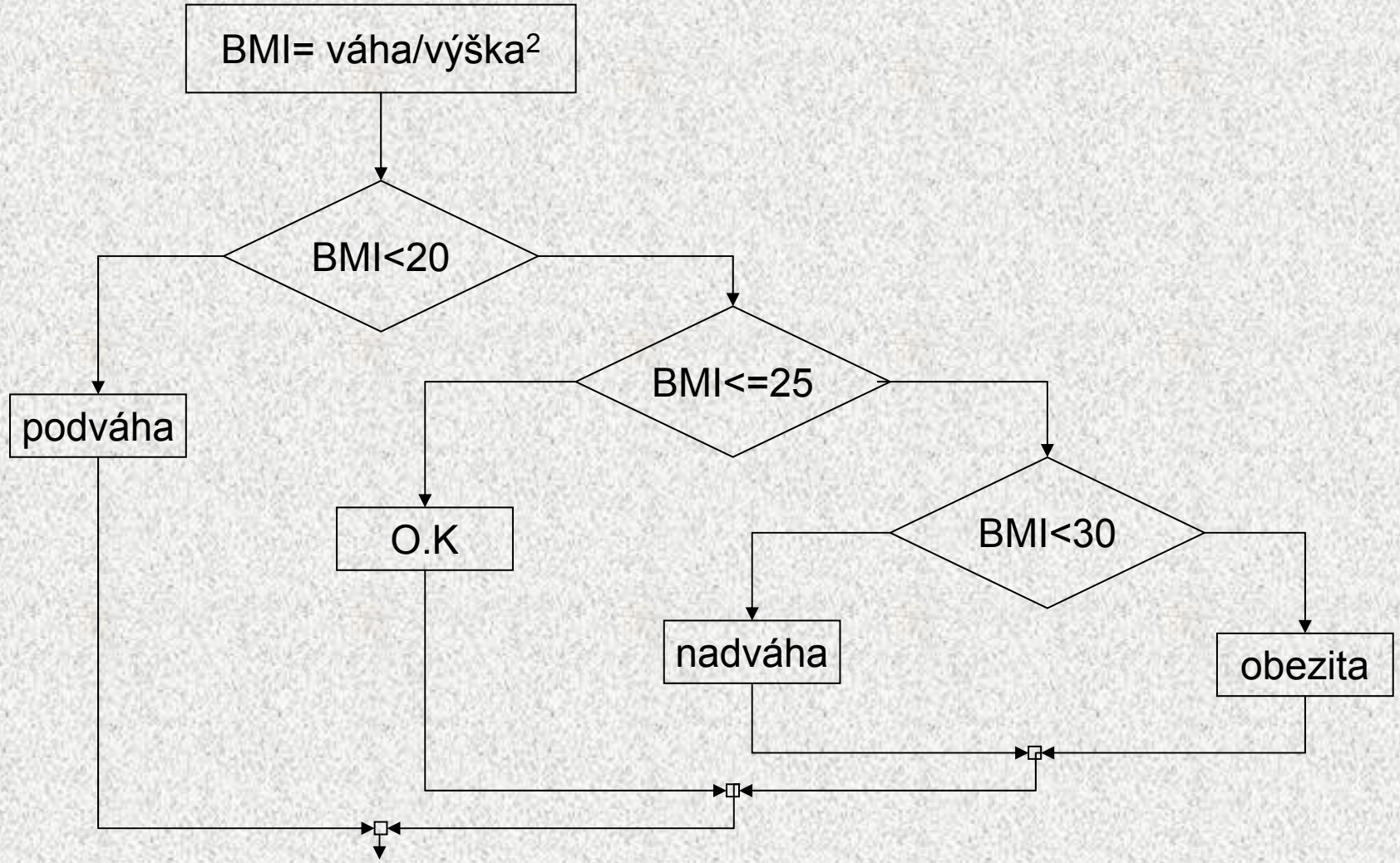
```
// 1. varianta
if (x < y) min = x; else min = y;
System.out.println(min);

// 2. varianta
min = x;
if (y < min) min = y;
System.out.println(min);
```



# Podmíněný příkaz ***if,*** příklad

```
// Urci, zda x lezi v intervalu <-25,50>,
// kdyz ano, nastav b = true; jinak b = false;

int x = 10; boolean b = false;
if(x > -25 && x < 50) b = true;

// Urci, zda y lezi v intervalu (-∞,-10> nebo <10,+∞)
// Pokud ano nastav y = 0,      b = true,
// jinak            y = y+10,   b = false

int x = 33; boolean b;
if(y < -10 || y > 10){
  y = 0;
  b = true;
}
else{
  y = y + 10;
  b = false;
}
```

# 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){
      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("Podvaha");
      else if(bmi <= 25)System.out.println("Vse OK");
         else if(bmi < 30)System.out.println("Nadvaha!");
            else System.out.println("Obezita!!");
   }
}
```

# Podmíněný příkaz, chyby

- 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

- Chybně:

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

- Správně:

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

# Podmíněný příkaz, příklad

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

```
  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;
 }
```

# Podmíněný příkaz, příklad

- 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

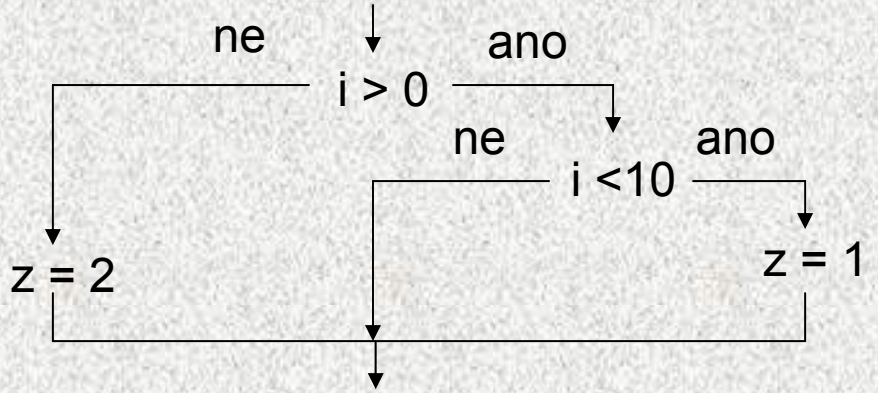
```
if (x < 0)s = -1;
else if (x == 0)s = 0;
     else s = 1;
```

- 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;
```

# Podmíněný příkaz, příklad chyby

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




- Chybně:

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

- Správně:

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

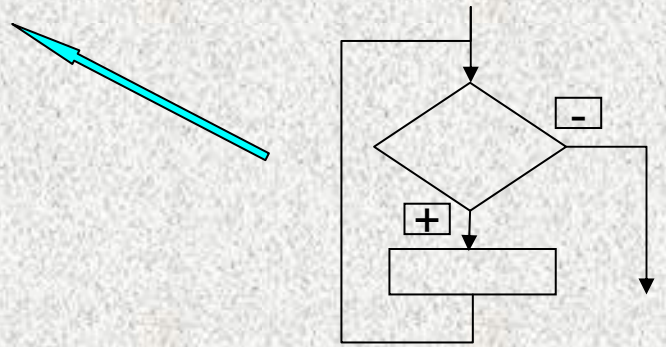
# Podmíněný příkaz, příklad

- Program, který pro zadaný rok zjistí, zda je přestupný
- Přestupný rok je dělitelný 4 a buď není dělitelný 100 nebo je dělitelný 1000

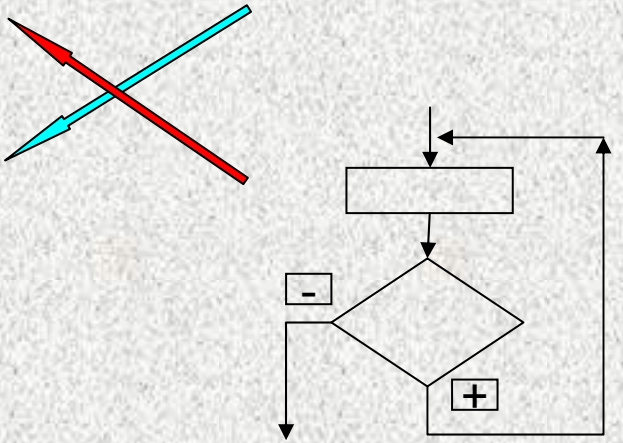
```
package pri03;
import java.util.*;

public class Rok {

  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int rok;
    System.out.println("zadejte rok");
    rok = sc.nextInt();
    System.out.println("rok "+rok);
    if (rok%4==0 && (rok%100!=0 || rok%1000==0))
      System.out.println(" je přestupný");
    else
      System.out.println(" není přestupný");
  }

}
```

# Cykly

```
while( ) {
    // Telo cyklu
  }
```

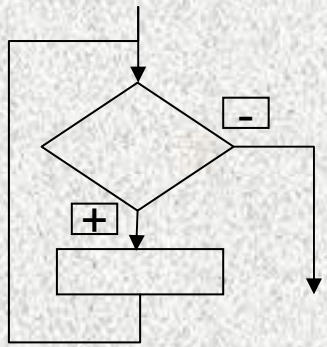



```
do {
    // Telo cyklu
} while( );
```




```
for( ; ; ){
    // Telo cyklu
}
```

# 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?)

- Má-li se opakovaně provádět více příkazů, musíme vytvořit 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?)

# Příkaz ***while***, příklad

```
// while( )          Ucel: opakovani vypoctu prikazu
//
// Pricti k x 6krat y

int x,y;
int i;

x = 0;
Y = 3;
i = 6;

while (i > 0){   // Test na zacatku cyklu
  x = x + y;
  i = i - 1;
}
```

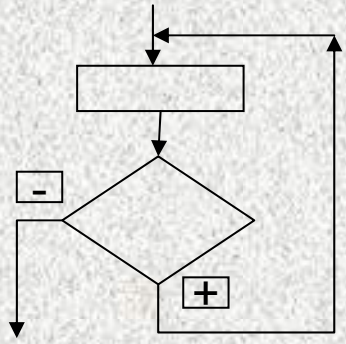
# Příkaz ***while,*** příklad

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

```
public class Faktorial {
  public static void main(String[] args) {
     Scanner sc = new Scanner(System.in);
     System.out.println("Zadejte prirozené cislo");
    int n = sc.nextInt();
    if (n < 1) {
      System.out.println(n + " Neni prirozené cislo");
      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 while***

- Příkaz cyklu *do-while* 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-while* provede tělo cyklu alespoň jednou, nelze jej tedy použít v případě, kdy lze očekávat ani jedno provedení těla cyklu

# Příkaz ***do while,*** příklad

```
// do{  } while( )      Ucel: opakovani vypoctu prikazu
//
// Vypocti n! (n! = 1*2*3*…*n)

int i,k,n;

i = 0;
k = 1
n = 5;

do {
   i = i + 1;
   k = k * i;
} while (i < n); // Test na konci cyklu
```

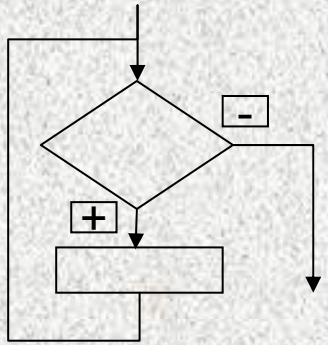
# Příkaz ***for***

- Tvar příkazu *for:*

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

- Provedení příkazu *for* :

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




- 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

# Příkaz ***for,*** příklad

```
// for( ; ; ){  }        Ucel: opakovani vypoctu prikazu
//
// Pricti k x cisla k,k+1,…,k=n
// Vynasob  y cisly k,k+1,…,k=n

int i;
int k,n;
int x=2,y=3;

k  = 4;
n  = 9;

for(i = k; i <= n; i = i + 1){// Test na zacatku cyklu
  x = x + i;
  y = y * i;
}
```

# Příkaz ***for***

```
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;
```

# Zpracování posloupností I

- Příklad: program pro součet posloupnosti čísel
- Hrubé řešení:

```
suma = 0;
while (nejsou přečtena všechna čísla) {
  dalsi = přečti celé číslo;
  suma = suma + dalsi;
}
```

- Jak určit, zda jsou přečtena všechna čísla?
- Možnosti:
  1. počet čísel bude vždy stejný, např. 5
  2. počet čísel bude dán na začátku vstupních dat
  3. vstupní data budou končit „zarážkou“, např. nulou
- Struktura vstupních dat formálně:
  1. $a_1\ a_2\ a_3\ a_4\ a_5$
  2. $n\ a_1\ a_2\ \ldots\ a_n$
  3. $a_1\ a_2\ \ldots\ a_5\ 0$, kde $a_i \neq 0$

# Zpracování posloupností II

- Řešení 1

  vstupní data: $a_1$ $a_2$ $a_3$ $a_4$ $a_5$

```
package pri03;
import java.util.*;
public class Suma1 {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int dalsi, suma, i;
    System.out.println ("Zadejte 5 cisel");
    suma = 0;
    for (i = 1; i <= 5; i++) {
      dalsi = sc.nextInt();
      suma = suma + dalsi;
    }
    System.out.println ("Soucet je " + suma);
  }
}
```

# Zpracování posloupností III

- Řešení 2

  vstupní data: $n\ a_1\ a_2\ \ldots\ a_n$

```
public class Suma2 {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int dalsi, suma, i, n;
     System.out.println("Zadejte pocet cisel");
     n = sc.nextInt();
     System.out.println("Zadejte " + n +" cisel");
     suma = 0;
     for (i = 1; i <= n; i++) {
       dalsi = sc.nextInt();
       suma = suma + dalsi;
     }
     System.out.println("Soucet je " + suma);
  }
}
```

# Zpracování posloupností IV

- Řešení 3

  vstupní data: $a_1\ a_2\ \ldots\ a_n\ 0$, kde $a_i \neq 0$

```
public class Suma3 {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int dalsi, suma;
    System.out.println("Zadejte radu cisel
                        zakoncenou nulou");
    suma = 0;
    dalsi = sc.nextInt();
    while (dalsi != 0) {
      suma = suma + dalsi;
      dalsi = sc.nextInt();
    }
  System.out.println("Soucet je " + suma);
  }
}
```

# 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 čísel dělitelných 10:

# Modifikace cyklu – vynechaná část výpočtu

```
while( ) {
    …
    if( )
      continue;
    …    // Vynechana cast vypoctu
}
```

```
do {
    …
    if( )
      continue;
    …    // Vynechana cast vypoctu
} while( );
```

```
for( ; ; ){
    …
    if( )
      continue;
    …    // Vynechana cast vypoctu
}
```

# *continue* - příklad

```
// continue      Ucel: vynechani casti tela cyklu
//
// Pricti k x cisla z intervalu <k,n>
// Vynasob  y cisly z intervalu <k,m>, kde m < n

int i;
int k,n,m;
int x=2,y=3;

k = 4;
n = 9;
m = n-2;

for(i = k; i <= n; i = i + 1){// Test na zacatku cyklu
  x = x + i;
  if(i >= m)
     continue;
  y = y * i;
}
```

# Příkaz ***break***

- Příkaz *break* vnořený do cyklu 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

# Modifikace cyklu – předčasné ukončení výpočtu

```
while( ) {
    …
    if( )
      break;     // Predcasne ukonceni cyklu
    …
}
```

```
do {
    if( )
      break;     // Predcasne ukonceni cyklu
    …
} while( );
```

```
for( ; ; ){
    if( )
      break;     // Predcasne ukonceni cyklu
 …
}
```

# Příkaz ***break*** – předčasné ukončení výpočtu, příklad

```
// break  Ucel: predcasne ukonceni cyklu
//
// Pricti k x cisla z intervalu <k,n>
// Vynasob  y cisly z intervalu <k,n>
// Kdyz je  y > m ihned ukonci dalsi vypocty
int i;
int k, n, m=18;
int x=2,y=3;

k = 4;
n = 9;

for(i = k; i <= n; i = i + 1){// Test na zacatku cyklu
  x = x + i;
  y = y * i;
  if(y > m)
     break;                    // Ihned ukonci cyklus
}
```

# 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);
       }
   }
}
```

# Konečnost cyklů

- Aby algoritmus byl konečný, musí každý cyklus v něm uvedený skončit po konečném počtu kroků
- Nekonečný cyklus je častou chybou
- Základní pravidlo pro konečnost cyklu:
  - provedením těla cyklu se musí změnit hodnota proměnné vyskytující se v podmínce cyklu
- Triviální příklad špatného cyklu:

```
while (i != 0) j = i-1;
```

  Tento cyklus se buď neprovede ani jednou, nebo neskončí

- Uvedené pravidlo konečnost cyklu ještě nezaručuje
- Konečnost cyklu závisí na hodnotách proměnných před vstupem do cyklu

```
double x=0; int i = -1;
while (i < 0){
  x = x + Math.sin(i* 0.6);
  i--;          // Nekonecny cyklus ? (modulo n)
}
```

# Konečnost cyklů

- Příklad:

```
while (i != n) {
  P;      // příkaz, který nezmění hodnotu proměnné i
  i++;
}
```

- Otázka: co musí splňovat hodnoty proměnných *i* a *n* před vstupem do cyklu, aby cyklus skončil?
- Odpověď – vstupní podmínka konečnosti cyklu:

  $i <= n$
- Vstupní podmínku konečnosti cyklu lze určit ke každému cyklu (někdy je to velmi obtížné)
- Splnění vstupní podmínky konečnosti cyklu musí zajistit příkazy předcházející příkazu cyklu
- Zásada: zabezpečený program testuje přípustnost vstupních dat
- Poznámka: pomůcka v Javě – příkaz **`assert`**

# Příkaz **switch**

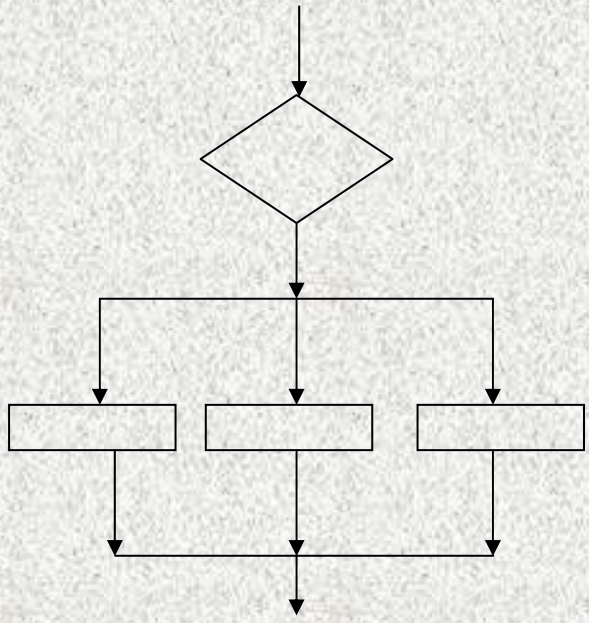
- Příkaz *switch* (programový přepínač) umožňuje větvení do více větví na základě různých hodnot výrazu (***hodnotou výrazu musí být celé číslo***)
- Základní tvar příkazu:

```
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 $příkazy_{def}$

# Příkaz ***switch,*** příklad

```
// switch( )     Ucel: vetveni programu do vice smeru
//
// Testuj x, kdyz x = … tak nastav do v = …
//                    2                    300
//                    6                    400
//                jina hodnota             500

int v,x=6;

switch(x) {                 // Reseni 1
   case 2:
     v = 300;
     break;
   case 6:
     v = 400;
     break;
   default:
      v = 500;
     break;
}
```

# Příkaz ***switch***

```
// switch( )     Ucel: vetveni programu do vice smeru
//
// Testuj x, kdyz x = ….  tak nastav do v = ….
//                   2                        300
//                   6                        400
//              jina hodnota                  500

int v,x=6;

v = 500;
switch(x) {                 // Reseni 2
   case 2:
      v = 300;
      break;
  case 6:
      v = 400;
      break;
  }
```

# 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");
  }
}
```



pro zájemce

A0B36PRI - PROGRAMOVÁNÏ

# Řízení běhu programu, řídící struktury

Jazyk JAVA

# Konec

České vysoké učení technické Fakulta elektrotechnická