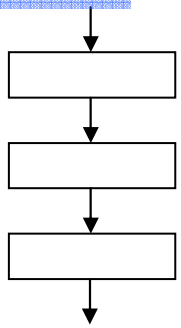
3. přednáška

Obsah:

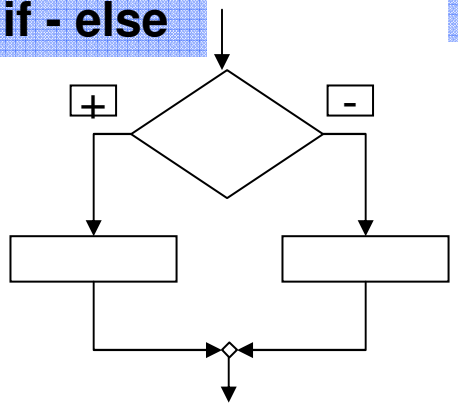
- Řídící struktury
  - sekvence,
  - **`if-else`**, **`switch`**,
  - **`for`**, **`while`**, **`do-while`**.
- Zpracování posloupnosti
  - nalezení největšího prvku, druhého nejvyššího prvku,
  - algoritmus shozeného praporku.

Řídící struktury

sekvence

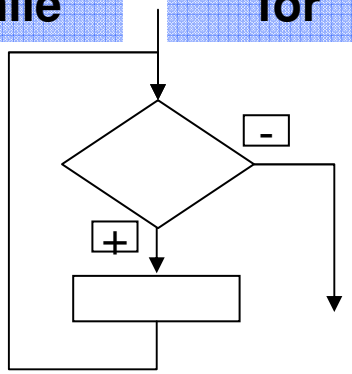


if - else

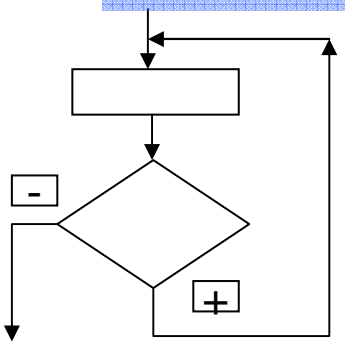


while

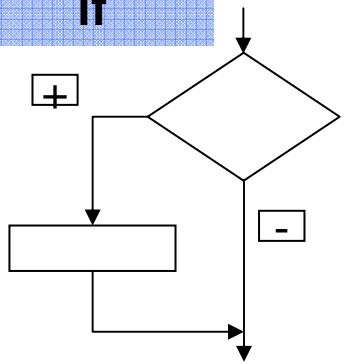
for



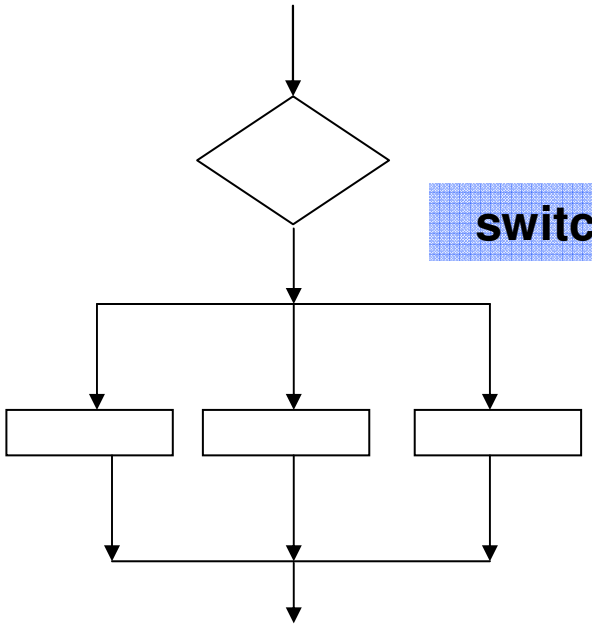
do



if



switch



Řídící struktury

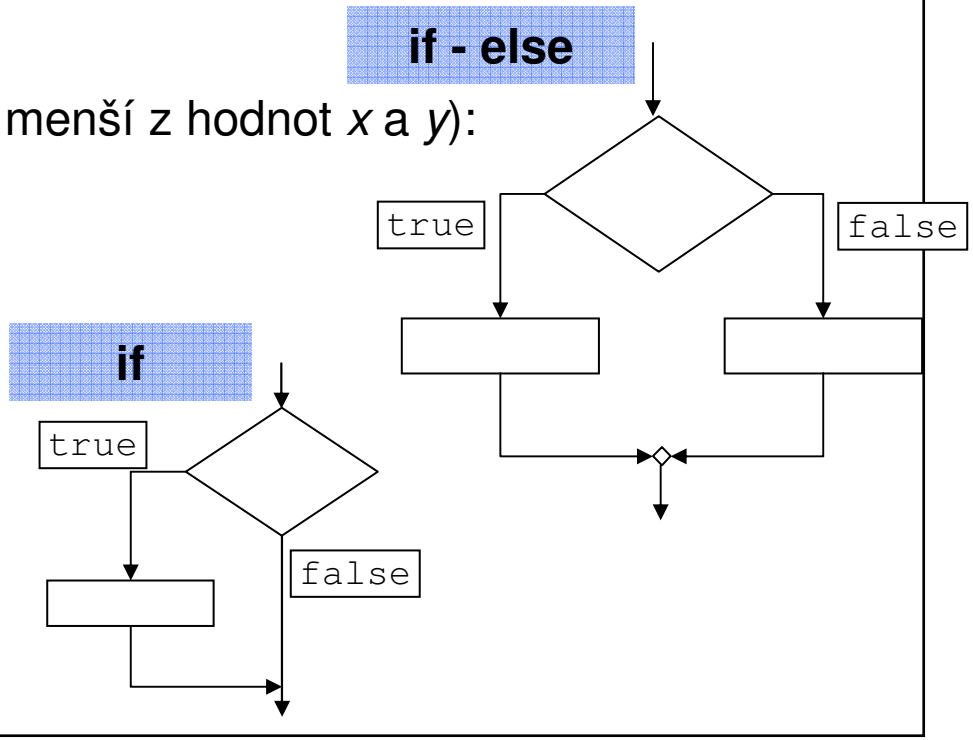
- Řídicí struktura je programová konstrukce, která se skládá z dílčích příkazů a předepisuje pro ně způsob provedení
- Tři druhy řídicích struktur:
  - *posloupnost*, předepisující postupné provedení dílčích příkazů
  - *větvení*, předepisující provedení dílčích příkazů v závislosti na splnění určité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

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

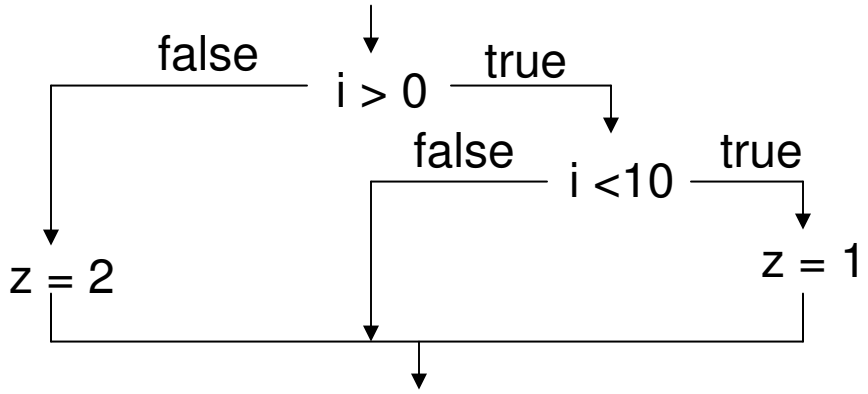
```
// 1. varianta
if (x < y) min = x;
   else min = y;
System.out.println(min);

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



Víceúrovňové větvení

- 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;
```

Dobře:

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

Hledání a výpis druhého největšího prvku - naivně

- Ze dvou čísel x, y:

```
if (x > y)System.out.println("max2 je" + y);
```

- Ze třech čísel x, y,z:

```
if (x>y && y>z)System.out.println("max2 je" + y);
else if (z>y && y>x)System.out.println("max2 je" + y);
else if (x>z && z>y)System.out.println("max2 je" + z);
else if (y>z && z>x)System.out.println("max2 je" + z);
else if (y>x && x>z)System.out.println("max2 je" + x);
else if (z>x && x>z)System.out.println("max2 je" + x);
```

počet různých pořadí je 3*2*1 = 6

- Ze čtyř čísel x, y, z, u:
  počet různých pořadí je 4*3*2*1 = 24
- Z pěti čísel …
  počet různých pořadí je 5*4*3*2*1 = 120
- A co když nebudou všechna čísla různá?

Hledání a výpis druhého největšího prvku - lépe

1. Zavedeme dvě proměnné max a max2
2. Nastavíme počáteční hodnoty z prvních dvou hodnot posloupnosti (**`x`** a **`y`**)
3. Pro každé další číslo ze vstupní posloupnosti rozhodneme, zda je nutné jejich hodnoty změnit (čteme je pomocí cyklu, viz. dále)

ad 1:
```
int max, max2;
```

ad 2:
```
if(x>y){max=x; max2=y;}
  else{max=y; max2=x;}
```

totéž jinak:
```
max  = x>y ? x : y;
max2 = x>y ? y : x;
```

ad 3: Příchozí číslo (**z**) je

- větší než **max**
- menší (nebo roven) než **max**, ale větší než **max2**
- menší než **max2**

Aktualizuj max i max2
{max2 = max;max = z}

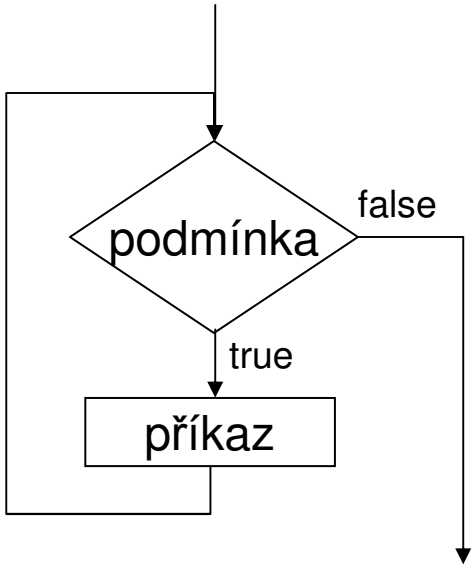
Aktualizuj max2
{max2 = z}

Nedělej nic

Příkazy cyklu: while

- Základní příkaz cyklu, který má tvar

  while (*podmínka*) *příkaz*



Příklad:

```
int x = 1;
while(x<10){
    System.out.print(x);
    x++;
}
```

//co bude vypsáno?

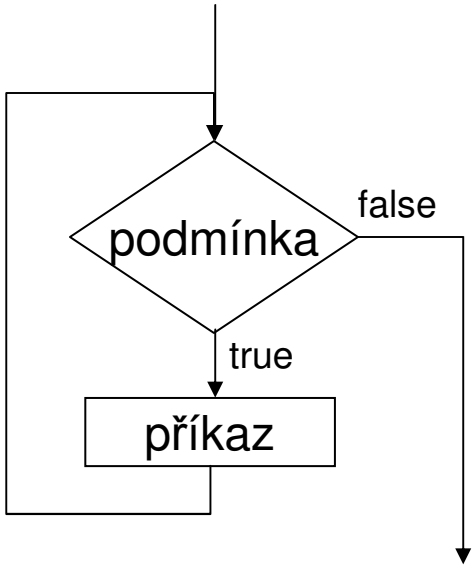
```
int x = 1;
while(x<10){
    System.out.print(x + ", ");
    x+=2;
}
```

//co bude vypsáno?

Příkazy cyklu: while

- Základní příkaz cyklu, který má tvar

  while (*podmínka*) *příkaz*



Příklad:

```
int x = 1;
while(x<10){
    System.out.print(x);
    x++;
}
```

//co bude vypsáno? **123456789|**

```
int x = 1;
while(x<10){
    System.out.print(x + ", ");
    x+=2;
}
```

//co bude vypsáno? **1, 3, 5, 7, 9, |**

Příkazy cyklu test na rovnost či nerovnost

Příklad:

```
int x = 1;
while(x!=10){ //nahrazeno znaménko ‘<‘
   System.out.print(x);
   x++;
}
```

//co bude vypsáno? **123456789|**

```
int x = 1;
while(x!=10){//nahrazeno znaménko <
   System.out.print(x + “, ”);
   x+=2;
}
```

//co bude vypsáno? **1, 3, 5, 7, 9, 11, 13, 15, 17, 19, …**

Používejte test na nerovnost!

Příkaz while

- 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?

Příklad: Faktoriál

- Výpočet faktoriálu přirozeného čísla *n*.

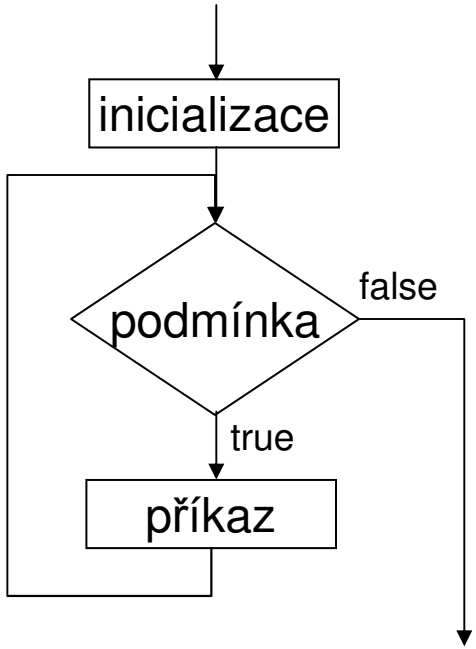
  $n! = 1 \times 2 \times \ldots \times n$

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

Příkaz for

- Tvar příkazu *for:*

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



- Převedení příkazu *for* na *while*:

```
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
- Příklad:

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

Zpracování posloupností

- 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 posloupnosti I

- Řešení 1

  vstupní data: a_1 a_2 a_3 a_4 a_5

```
public class Suma1 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int dalsi, suma, i;
        System.out.println ("zadejte 5 čísel");
        suma = 0;
        for (i=1; i<=5; i++) {
            dalsi = sc.nextInt();
            suma = suma+dalsi;
        }
        System.out.println ("součet je " + suma);
    }
}
```

Zpracování posloupnosti II

- Řešení 2

  vstupní data: n a_1 a_2 ... a_n

```
public class Suma2 {
    public static void main(String[] args) {
        int dalsi, suma, i, n;
        System.out.println("zadejte počet čísel");
        n = sc.nextInt();
        System.out.println("zadejte " + n +" čísel");
        suma = 0;
        for(i=1; i<=n; i++) {
            dalsi = sc.nextInt();
            suma = suma+dalsi;
        }
        System.out.println("součet je " + suma);
    }
}
```

Zpracování posloupnosti III

- Řešení 3

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

```
public class Suma3 {
    public static void main(String[] args) {
        int dalsi, suma;
        System.out.println("zadejte řadu čísel
                            zakončenou nulou");
        suma = 0;
        dalsi = sc.nextInt();
        while(dalsi!=0) {
            suma = suma+dalsi;
            dalsi = sc.nextInt();
        }
        System.out.println("součet je " + suma);
    }
}
```

Přerušení běhu cyklu: 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:

Přerušení běhu cyklu: 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);
      }
   }
}
```

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--;
}
```

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:

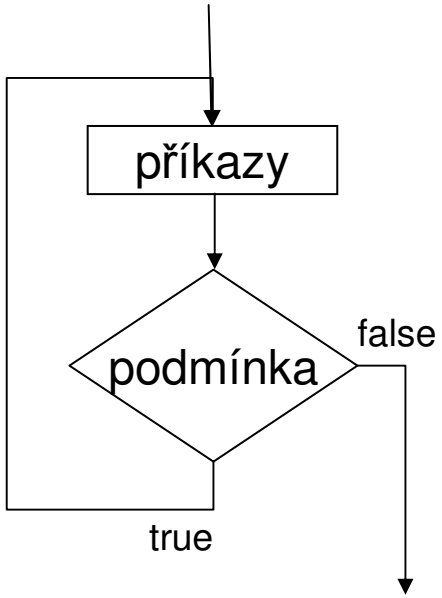
```
i <= n
```

- 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.

Metoda shozeného praporku

- Testování jedné podmínky pro všechny prvky dané posloupnosti.
  1. Nastav praporek indikující splnění podmínky na hodnotu **true.**
  2. Postupně procházej posloupnost a pro každý nový člen ověř, že je podmínka stále splněna
  3. Pokud není splněna, shoď praporek
- Příklad: Zjistěte zda daná posloupnost obsahuje **pouze sudá čísla**

  Využijeme cyklus **do.**



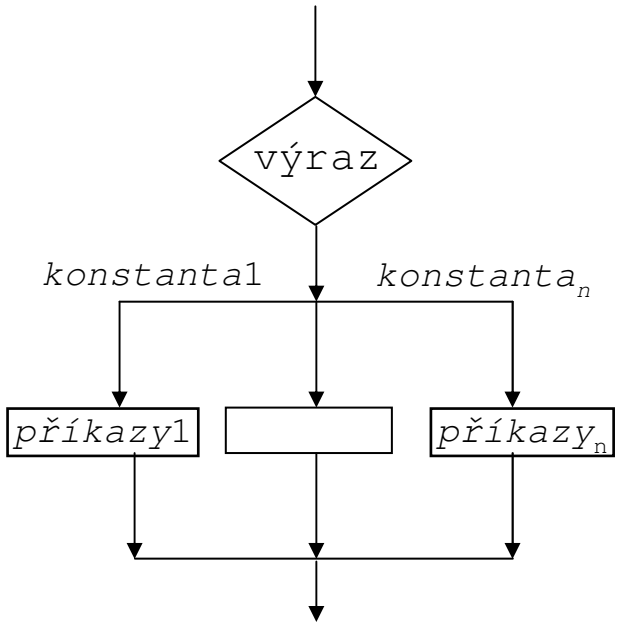
Metoda shozeného praporku - příklad

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    boolean vsechnaSuda = true; // praporek, je zvednutý
    int cislo; // aktualní clen posloupnosti
    System.out.println("Budou všechna čísla sudá?");
    System.out.println("Zadávej čísla (0 je stop).");
    do {
        cislo = sc.nextInt();
        if (cislo % 2 == 1)
            vsechnaSuda = false; // pád praporku, liché
        else
            vsechnaSuda = true; // Chybné nahození praporku
    } while (cislo != 0);
    if (vsechnaSuda)System.out.println(“Všechna sudá”);
    else System.out.println(“Alespoň jedno bylo liché");
}
```

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 všechny *příkazy*, počínaje těmi, které jsou označeny *konstantou* označující stejnou hodnotu až po break.
  - není-li žádná větev označena hodnotou výrazu, provedou se $příkazy_{def}$

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;
      System.out.print(den+"."+mesic+"."+rok+" je "+n+". den v
                       roce");
   }
}
```