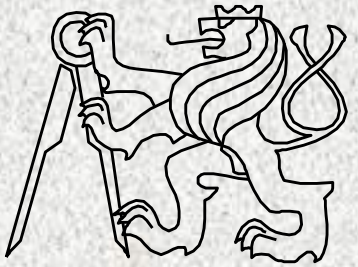
PROGRAMOVACÍ STYLY



BD6B36PJV

Fakulta elektrotechnická
České vysoké učení technické

Dva základní přístupy k imperativnímu programování

- Strukturované – procedurální

  Princip strukturovaného přístupu:

Program = Data + Algoritmus

- Objektově

Programovací jazyky a programovací styly

Programovací jazyky

- deklarativní
- imperativní

Programovací styly

- *Naivní*
- *Procedurální*
- *Objektově orientovaný*
- Návrhové vzory
- SOA - service-oriented architecture

Deklarativní programovací jazyky

- nepopisují „ jak něco udělat “
- popisují „ co má být výsledkem “

- HTML (HyperText Markup Language)
  - specifikuje, jak má stránka vypadat (font - typ, velikost, barva, nadpis), ale nepopisuje, jakým způsobem se má stránka na daném počítači zobrazit

- Prolog
  - Nepopisuje „kroky“ výpočtu, ale:
    - Dodají se fakta o problému – databázi znalostí
    - Sformuluje se požadovaný dotaz – cíl, který lze možná odvodit

Příklad HTML kódu

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
  Transitional//EN">
<html>
  <head>
  <meta http-equiv="content-type" content="text/html;
   charset=UTF-8">
  <title>Super stránka</title>
  </head>
  <body>
    <h1>Nadpis</h1>
    Obsah:
    <ol>
     <li>Úvod</li>
     <li>Závěr</li>
    </ol>
  </body>
</html>
```

Příklad PROLOG

Fakta

```
rodic(jana,petr).
rodic(petr,eva).
dite(X,Y) :- rodic(Y,X).
predek(X,Y) :- rodic(X,Y).
predek(X,Y) :- rodic(X,Z), predek(Z,Y).
```

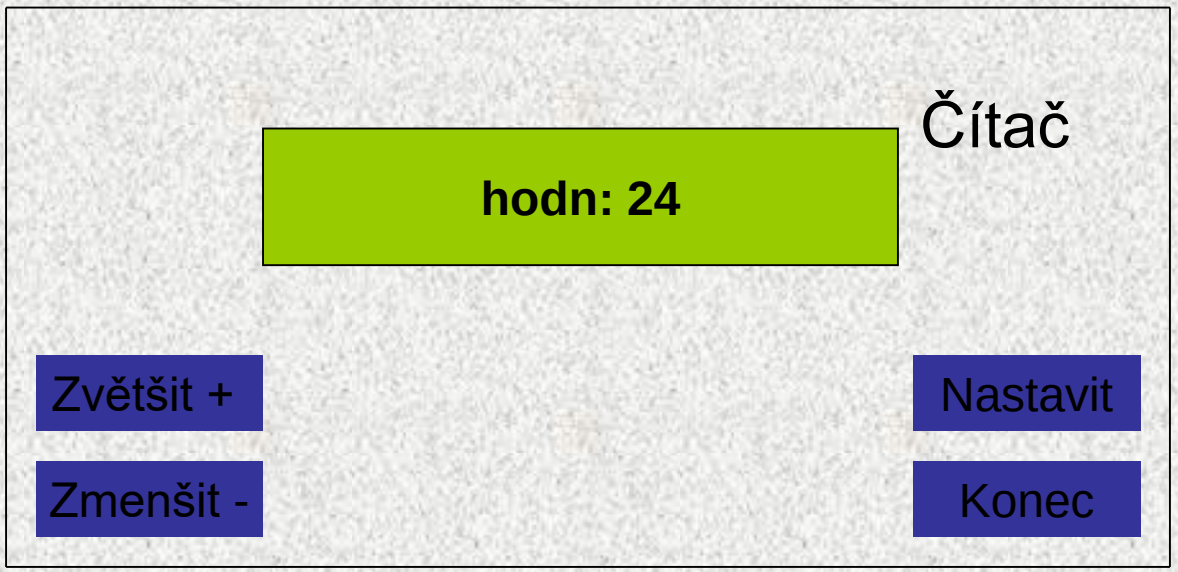
Dotaz a odpověď

```
?- rodic(X,petr).
X = jana ;
No
?- dite(X,jana).
X = petr ;
No
?- predek(jana,eva).
Yes
```

Programovací styly - příklad

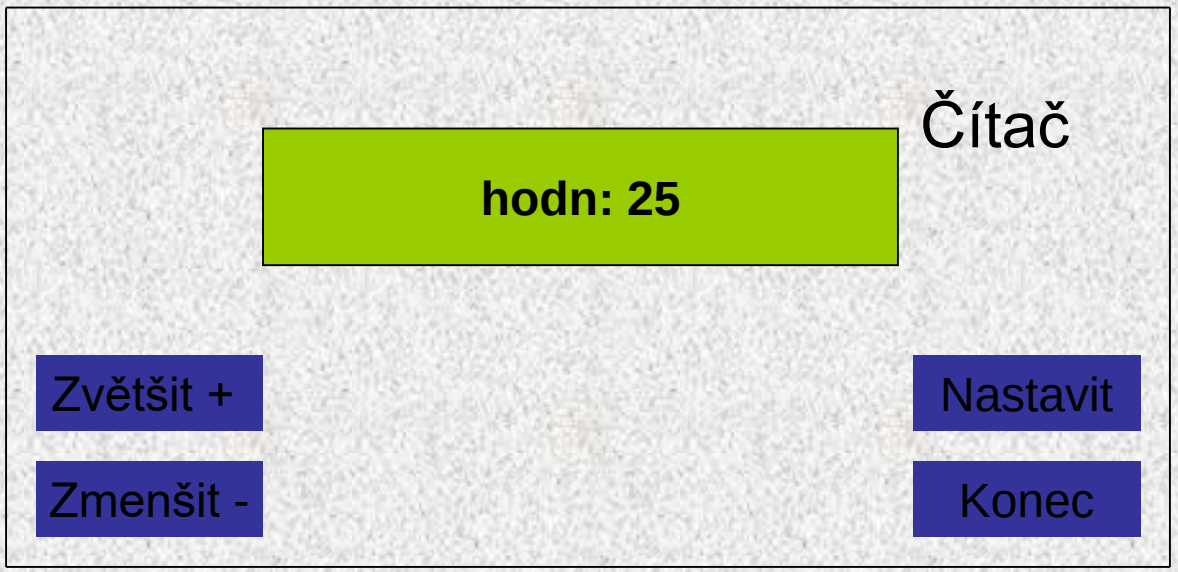
Na příkladu programu simulujícího čítač si ukážeme programovací styly

Stav čítače:



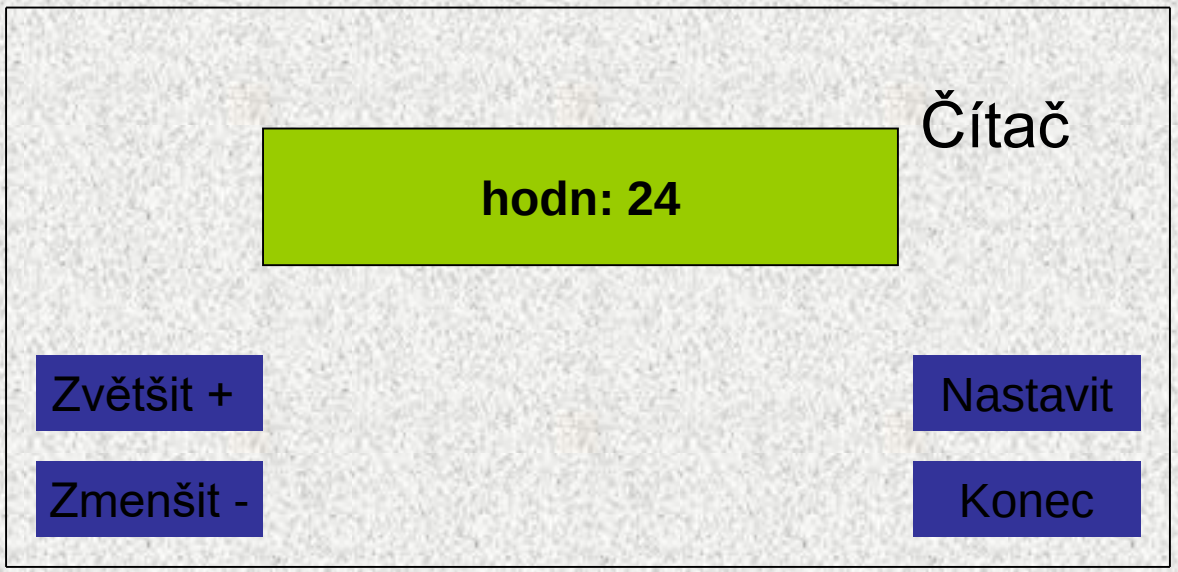
Programovací styly - příklad

Stav čítače po stisku „Zvětšit“



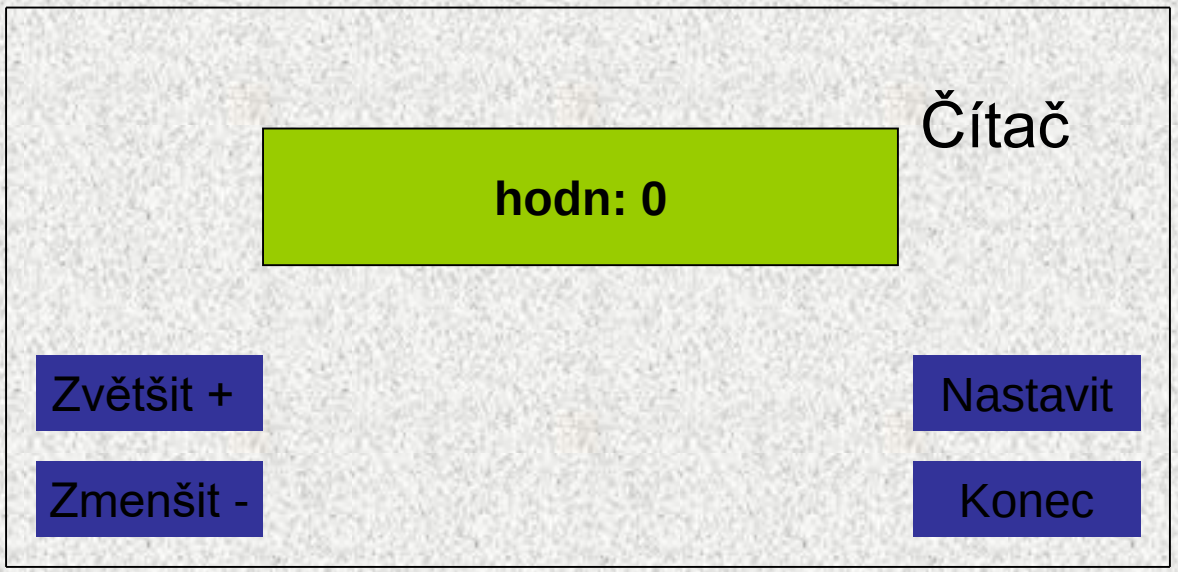
Programovací styly - příklad

Stav čítače po stisku „Zmenšit“



Programovací styly - příklad

Stav čítače po stisku „Nastavit“



Programovací styly - příklad

Stav čítače po stisku „Konec“

Programovací styly – příklad, realizace

- Příklad komunikace programu (textově):

Hodnota = 24

0) Konec
1) Zvětšit
2) Zmenšit
3) Nastavit
Vase volba:

1

Hodnota = 25

0) Konec
1) Zvětšit
2) Zmenšit
3) Nastavit

Vase volba:

2

Hodnota = 24

Naivní styl

- Jednoduché úlohy lze řešit přímočaře, vše implementováno v jedné třídě:

```
public class CitacNaivni{
final static int pocHodn = 0;
  static int hodn, volba;
  public static void main(String[] args) {
   Scanner sc = new Scanner(System.in);
   hodn = pocHodn;
   do {
   System.out.println("Hodnota = "+hodn);
   System.out.println("0) Konec\n1) Zvětšit\n2 Zmenšit\n3)
                                                  Nastavit");
   System.out.print("Vaše volba: ");
      volba = sc.nextInt();
      switch (volba) {
        case 0: break;
        case 1: hodn++; break;
        case 2: hodn--; break;
        case 3: hodn = pocHodn; break;
        default: System.out.println("Nedovolená volba");
      }
    } while (volba>0);
    System.out.println("Konec");
  }}
```

Procedurální styl

- Připomeňme hlavní zásady:
  - Zadaný problém se snažíme rozložit na podproblémy
  - Pro řešení podproblémů zavádíme abstraktní příkazy, které nejprve specifikujeme a pak realizujeme pomocí procedur a funkcí

- Aplikace na čítač - dva dílčí podproblémy:

1. provedení požadované operace s čítačem

```
static void operace(int op)
```

2. komunikace s uživatelem, jejímž výsledkem je kód požadované operace

```
static int menu()
```

Procedurální styl – operace s čítačem

```
public class CitacProceduralni {
  final static int pocHodn = 0;
  static int hodn;

  static void operace(int op) {
    switch (op) {
      case 1: hodn++; break;
      case 2: hodn--; break;
      case 3: hodn = pocHodn; break;
    }
  }
```

Procedurální styl - komunikace

```
static int menu() {
      Scanner sc = new Scanner(System.in);
      int volba;
    do {
      System.out.println("0. Konec");
      System.out.println("1. Zvětšit");
      System.out.println("2. Zmenšit");
      System.out.println("3. Nastavit");
      System.out.print("Vaše volba: ");
      volba = sc.nextInt();
      if (volba<0 || volba >3) {
        System.out.println("Nedovolená volba");
        volba = -1;
      }
    } while (volba<0);
    return volba;
  }
```

Procedurální styl – main

```
public static void main(String[] args) {
    int volba;
    hodn = pocHodn;
    do {
      System.out.println("Hodnota = "+hodn);
      volba = menu();
      if (volba>0) operace(volba);
    } while (volba>0);
    System.out.println("Konec");
  }
```

Poznámky:

- procedurální řešení čítače (nelokální proměnná pro hodnotu + nelokální konstanta udávající počáteční hodnotu) je nedokonalé (proč?)
- čítač je typ, pro který jsou definovány určité operace (realizovatelné procedurami a funkcemi) a jehož implementace (proměnná pro hodnotu) by neměla být volně přístupná
- pro realizaci čítače je vhodnější objektově orientovaný styl

Procedurální přístup v Javě - shrnutí

- Dosavadní využití třídy
  - To jsme poznali dosud - procedurální přístup
  - Základní princip
    - Strukturalizace problému, algoritmizace
    - Návrh datových struktur

- Základem uživatelského programu v jazyku Java je třída, ve které
  - musí být deklarována hlavní funkce **`main`**
  - jsou deklarovány další **`(static)`** funkce (či procedury) třídy, případně i v jiných třídách – projekt, **`package`**
  - mohou být deklarovány statické proměnné, které jsou použitelné jako nelokální proměnné ve funkcích dané třídy
  - program probíhá spuštěním příkazů metody **`main`**

Objektový přístup – přerušení výkladu

- Vybudován s využitím knihovny SWING