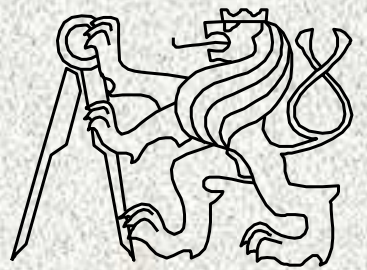
PROGRAMOVACÍ STYLY

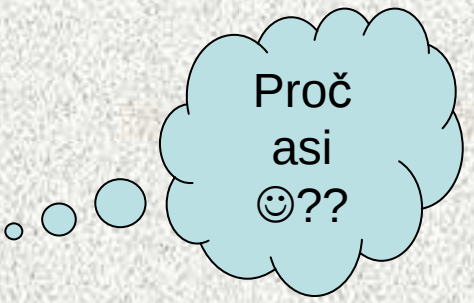


A0B36PR2-Programování 2

Fakulta elektrotechnická
České vysoké učení technické

Kdo je kdo v Programování 2

1. Studenti, absolventi předmětu Programování 1
   - Pro studenty OI paralelně předmět Algoritmizace
2. Přednášející:
   - Ing. Miroslav Balík, PhD.,
   - Doc.Ing. Jelínek Ivan, CSc.
3. Nejdůležitější učitelé: cvičící!!
   - Ing. M. Balík, PhD
   - Ing. Bloch Martin, CSc.
   - Ing. Buk Zdeněk
   - Ing. R. Malinský
   - Ing. M. Mudroch
   - Ing. J. Kuchař
   - Ing. M. Balík
   - Ing. L. Hapalová



- Garant: Doc.Ing. Jelínek Ivan CSc.

Cíl předmětu Programování 2

Předmět navazuje na Programování 1 a klade si za cíl

- naučit studenty vytvořit aplikaci s grafickým uživatelským rozhraním se znalostí témat:
- polymorfismus,
- zpracovaní události,
- princip mechanismu výjimky,
- aplety,
- API databází
- práce s uživatelskými knihovnami, kolekce
- Dále je student seznámen s jazykem C (student je schopen programy v jazyku C analyzovat):
  - komparativní výklad jazyka C,
  - struktura programu
  - funkce a parametry
  - pointery
  - dynamická správa paměti

Organizace předmětu

- A0B36PR2 Programování 2
- Rozsah: 2p+2c
- Zakončení Z,ZK
- Kredity 6
  - po prvním roce je nutné získat alespoň 30
  - výsledná známka ovlivní možnost tvorby rozvrhu ve 3. semestru
- https://eduweb.fel.cvut.cz/courses/A0B36PR2

Osnova přednášek

1. Programovací styly. Opakování a shrnutí základů programování v Javě, objektový přístup, struktura tříd a programu v Javě
2. Třídy 3 Polymorfismus, řešení abstraktní třídou, rozhraní, rozhraní jako typ proměnné, rozhraní a dědičnost, typ interface, ENUM a generika ??
3. Grafické uživatelské rozhraní (GUI) v jazyce Java, typy komunikace, knihovny AWT a SWING, princip GUI, komponenty, kontejnery, správce rozmístění, obsluha událostí
4. Události jako objekt, zpracování události, zdroj události, posluchač události, model šíření události, model šíření událostí, zpracování vlastní události, více zdrojů a posluchačů, rozlišení zdrojů,
5. Výjimky, princip mechanismu zpracování výjimek, kompletní zpracování výjimek, vyhození výjimky, propagace výjimek, vlastní výjimky, hierarchie výjimek, kontrolované a nekontrolované výjimky
6. Aplety, vlastnosti, použití, způsob aktivace, životní cyklus apletu, předávání parametrů do apletu, omezení apletu, zmírnění omezení, podepisování apletů
7. Kontejnery, použití knihoven seznamů, množin, map v jazyce Java, příklady použití
8. Vlákna, vícevláknový program, deadlock a řešení, synchronizace vláken
9. Databáze, velmi přehledově základy SQL, JDBC, připojení z Javy,
10. Základy programovaní v C, charakteristika jazyka, model kompilace, struktura programu, struktura funkce, příklad programu
11. Komparativní výklad jazyka C k jazyku Java, makra, podmíněný překlad, syntaxe jazyka, základní vstup a výstup
12. Systematické programování v C, preprocesor, základní knihovny, struktury, uniony, výčtové typy
13. Pointery v C, dynamická správa paměti, pole a ukazatelé, funkce a pointery

Osnova cvičení

1. Úvodní test, zopakování základů programování a objektového přístupu
2. Polymorfismus, abstraktní třída, zadání semestrální práce
3. Rozhraní, EMUM a generika
4. Grafické uživatelské rozhraní
5. Zpracování událostí
6. Zpracování výjimek
7. Aplety
8. Kolekce
9. Vlákna
10. Databáze
11. Seznámení s prostředím pro vývoj programu v jazyce C, analýza programů v jazyce C, odladění jednoduché úlohy
12. Systematické programování v C, Pointery, dynamická správa paměti, pole a ukazatelé
13. Test, zápočet

Hodnocení a zkouška

Zdroje bodů pro hodnocení	Body
aktivita a DÚ na cvičeních	25 b (min. 10 b)
semestrální práce	40 b (min. 20 b)
kontrola rozpracovanosti sem. práce	10 b (min. 5 b)
písemný zkouškový test	20 b (min. 10 b)
Ústní zkouška	25 b (-10b pokud k ústní)

Možnost nechat si zapsat známku nebo jít k ústní zkoušce – odečte se 10 bodů

Klasifikace *na základě bodového hodnocení)*				
klasifikace	***počet bodů***	**číselně**	**slovně**	
A	**90 - 100**	**1**	**výborně**	
B	**80 - 89**	**1,5**	**velmi dobře**	
C	**70 - 79**	**2**	**dobře**	
D	**60 – 69**	**2,5**	**uspokojivě**	
E	**50 - 59**	**3**	**dostatečně**	
F	**< 50**	**4**	**nedostatečně**	

Body ze cvičení, maximálně 75, při 71-75 resp. 66-70 bodech ze cvičení: (A)výborně resp. (B)velmi dobře bez zkoušky

Doporučená literatura

Základní zdroje:

- Poznámky z přednášek a cvičení
- Slidy z přednášek http://eduweb.fel.cvut.cz/courses/A0B36PR2
- Základní příručky jazyka Java:
  - Herout, P.: Učebnice jazyka Java, Kopp, 2007
  - Keogh, J.:Java bez předchozích znalostí, Computer Press, 2005
  - Zakhour, S: Java 6, výukový kurz, CPress, Brno, 2007
  - Pecinovský, R.: Java 5.0, Novinky a upgrade aplikací, Computer Press, 2005
- Základní příručky jazyka C:
  - Herout, P.: Učebnice jazyka C. III. vyd. Kopp, Č.Budějovice, 2002.
  - Kernighan, B.W.-Ritchie, D.M.: Programovací jazyk C. Computer Press,
  - Herout, P.: Učebnice jazyka C – 2 díl, Kopp, Č.Budějovice, 2002.

Další zdroje (publikace v češtině**)**:

- Eckel, B.: Myslíme v jazyku Java, Grada, 2000, I + II
- Chapman, S., J.: Začínáme programovat v jazyce JAVA, Computer Press, 2001
- Hawlitzek, JAVA2, příručka programátora, Grada, 2000
- Shildt, H.: Java 2, Příručka programátora, Softpress, 2001
- Herout, P.: JAVA, grafické uživatelské prostředí a čeština, Kopp, 2001

Začínáme

Šest zákonů programování

1. V každém programu je alespoň jedna chyba
2. Každý program lze zkrátit alespoň o jeden řádek
3. Nejjednodušší chyby se nejhůře hledají
4. Každou opravou se do programu zanese nová chyba
5. Když už se zdá, že program je v pořádku, určitě jste něco přehlédli
6. Programátor dělá to co umí, počítač si dělá, co chce

Programovací jazyky a programovací styly

Programovací jazyky

- deklarativní
- imperativní

Programovací styly

- Naivní
- Procedurální
- Objektově orientovaný
- Návrhové vzory

Deklarativní programovací jazyky

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

- Jiný příklad – Prolog
  - Nepopisuje „kroky“ výpočtu, ale:
    - fakta o problému – databázi znalostí
    - požadovaný výsledek – 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(otto,eva).
dite(X,Y) :- rodic(Y,X).
```

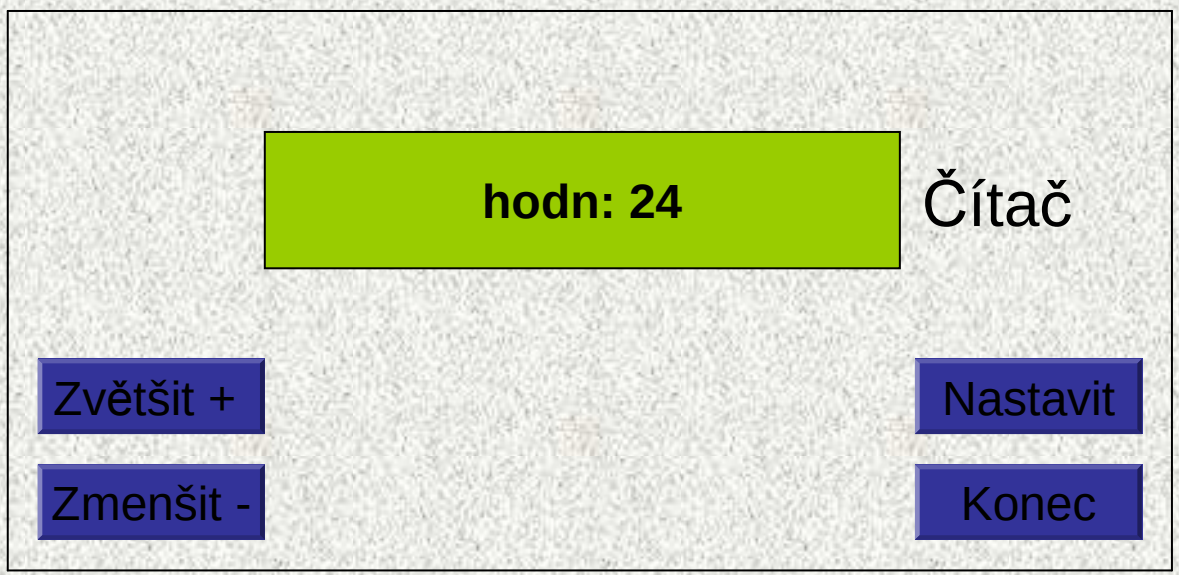
Dotaz a odpověď

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

Programovací styly - příklad

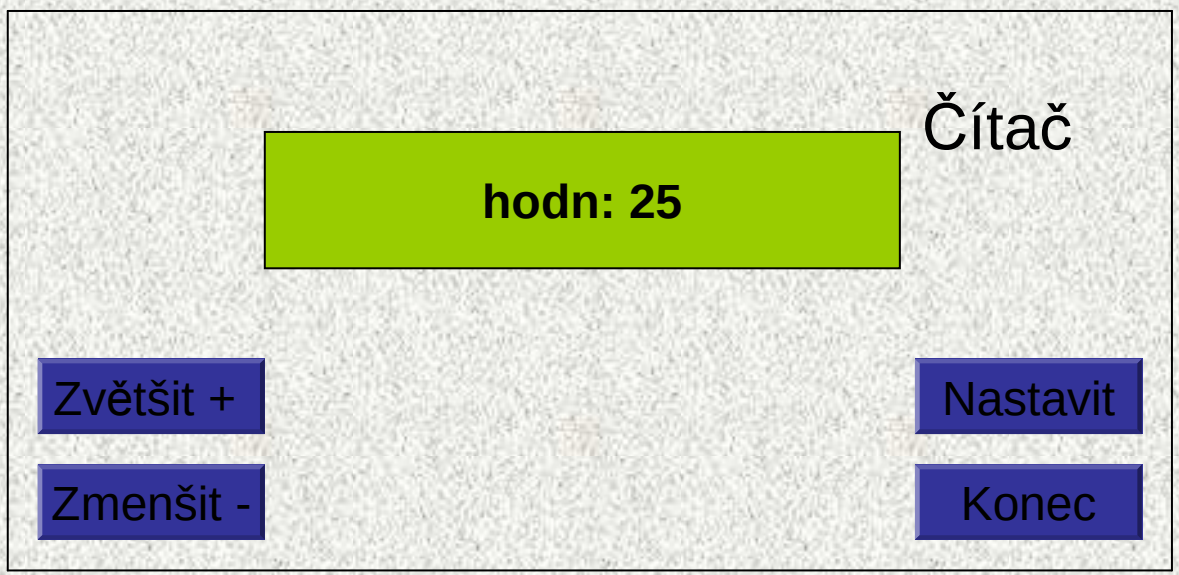
Na příkladu programu simulujícího čítač si ukážeme programovací styly

Stav čítače:



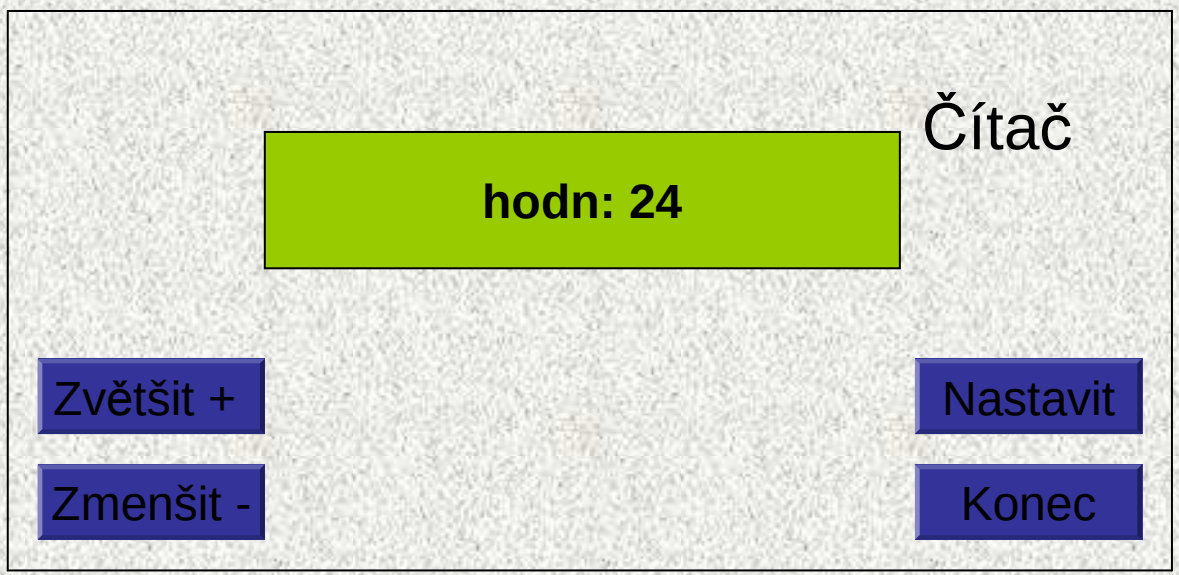
Programovací styly - příklad

Stav čítače po stisku „Zvětšit“



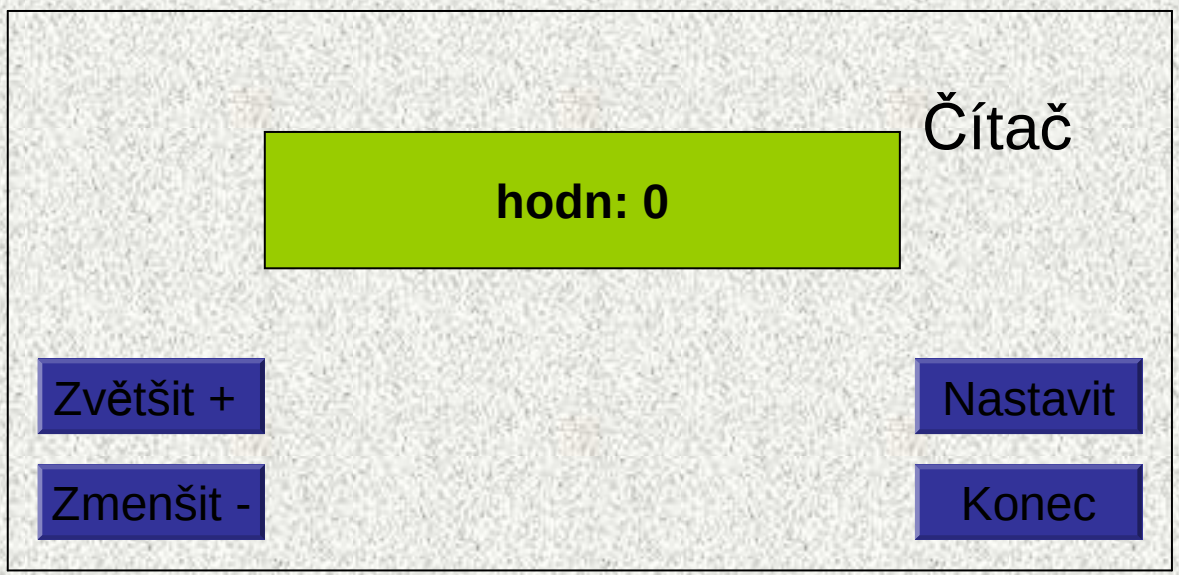
Programovací styly - příklad

Stav čítače po stisku „Zmenšit“



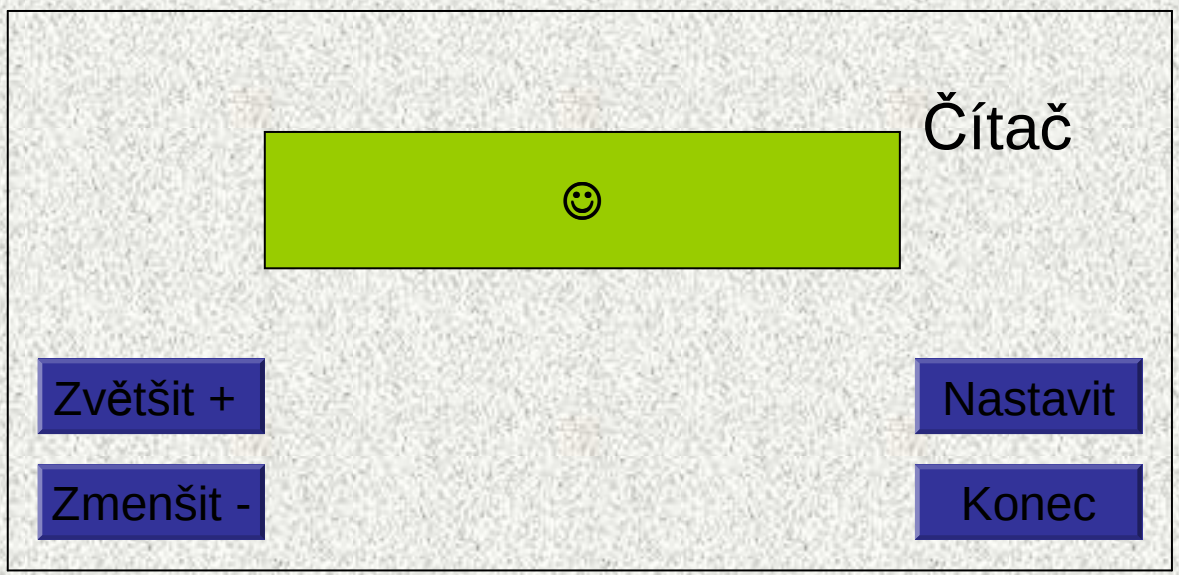
Programovací styly - příklad

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 Citac1{
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()
```

Pozn.: často vše implementováno v jedné třídě

Procedurální styl – operace s čítačem

```
public class Citac2 {
  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 (globální proměnná pro hodnotu + globální 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

Objektově orientovaný styl

- Hlavní zásady:
  - Při rozkladu problému specifikujeme nové datové typy (datové abstrakce), tzn. specifikujeme operace, které se budou s daty těchto typů provádět
  - Pro realizaci datových abstrakcí použijeme objekty a třídy
- Připomeňme si charakteristiku objektu:
  - datové položky a metody objektu jsou dány typem objektu - třídou
  - objekt se může nacházet v různých stavech daných hodnotami datových položek (atributů) objektu
  - chování objektu je dáno metodami (operacemi), které jsou pro něj definovány

Třídy a objekty - připomenutí

- Třída popisující nový datový typ obsahuje:
  - deklarace položek, ze kterých se skládají objekty daného typu
  - deklaraci konstruktoru, kterým se inicializují vytvořené objekty
  - deklarace metod, které realizují operace s objekty
- Položky se obvykle deklarují tak, aby nebyly z vnějšku objektu přístupné:
  `private` *typ jméno*;
- Konstruktor je inicializační operace, která má jméno třídy a nevrací žádnou hodnotu; schéma deklarace konstruktoru třídy *T* (konstruktor by měl být z vnějšku přístupný!! ☺):

```
public T(specifikace parametrů) {
  tělo konstruktoru
}
```

- Ve třídě může být deklarováno několik konstruktorů, které se liší počtem a/nebo typy parametrů (přetěžování)
- Obvyklé schéma deklarace metody (přístupnost metody je dána logikou problému):

```
public typ jméno(specifikace parametrů) {
  tělo metody
}
```

Přístup k položkám objektu – připomenutí 1

- V metodě (konstruktoru) jsou položky objektu, na který se metoda (konstruktor) aplikuje, přímo přístupné prostřednictvím svých jmen

```
public class Citac {
   private int hodn;
   private int pocHodn;
...
public void zvetsit() {
    hodn++;  // zvětší se hodnota položky objektu, na který
             // se metody aplikuje
}
```

- Chceme-li zdůraznit, že se jedná o položku objektu, můžeme (někdy musíme) před její jméno napsat **this.**

```
public void zvetsit() { this.hodn++; }
```

Přístup k položkám objektu – připomenutí 2

- Položka objektu může být deklarována jako **`public;`** pak je zvenčí objektu přístupná pomocí operátoru "." a svého jména

```
public class Complex {
    public double re, im;
}
...
Complex x = new Complex();
  x.re = 1.25; x.im = 5.3;
```

- **Nevhodné řešení – nesplňuje vlastnost zapouzdření!!!**
  - Řešení – setry a getry

```
public class Complex {
    double re, im;
}
Complex x = new Complex();
public double getRe(){
   return this.re;}
public void putRe(double re ){
   this.re = re;}
```

Třída Complex

```
public class Complex {

  public double re;
  public double im;

  public Complex() {re=0; im=0;}
  public Complex(double r) {re=r; im=0);}
  public Complex(double r, double i) {re=r; im=i;}

  public double abs() {
    return Math.sqrt(re*re+im*im);
  }

  public Complex plus(Complex c) {
    return new Complex(re+c.re, im+c.im);
  }

  public Complex minus(Complex c) {
    return new Complex(re-c.re, im-c.im);
  }

  public String toString() {
    return "["+re+", "+im+"]";
  }
```

Čítač jako datový typ

- Čítač zavedeme jako datový typ s operacemi

  zvetsit, *zmensit*, *nastavit* a *hodnota*

  a s datovými položkami

  hodn a *pocHodn*

- Grafické vyjádření:

název typu

datové položky

operace (metody)

- Poznámka: hodnota položky *pocHodn* bude stanovena při vytvoření objektu

Třída Citac

```
public class Citac {
  private int hodn;
  private int pocHodn;

  public Citac(int ph) {
     pocHodn = ph;
     hodn = ph;
  }

  public void zvetsit() {hodn++;}

  public void zmensit() {hodn--;}

  public void nastavit() {hodn = pocHodn;}

  public int hodnota() {return hodn;}
}
```

Použití čítače jako objektu

```
public static void main(String[] args) {
    int volba;
    Citac citac = new Citac(0);
//        Citac citac1 = new Citac(3);
    do {
      System.out.println("Hodnota = "+citac.hodnota());
      volba = menu();
      switch (volba) {
        case 1: citac.zvetsit(); break;
        case 2: citac.zmensit(); break;
        case 3: citac.nastavit(); break;
     //       case 4: System.out.println("nic"); break;
      }
    } while (volba>0);
    System.out.println("Konec");
  }
}
```

Třída CitacModulo, dědění

```
public class CitacModulo extends Citac {
public CitacModulo(int ph) {
super(ph);
}
@Override
public void zvetsit() {
System.out.println("  " + hodn % 5);
hodn++;
hodn = ((hodn % 5) == 0) ? pocHodn : hodn;
    }
}
```

Menu jako datový typ

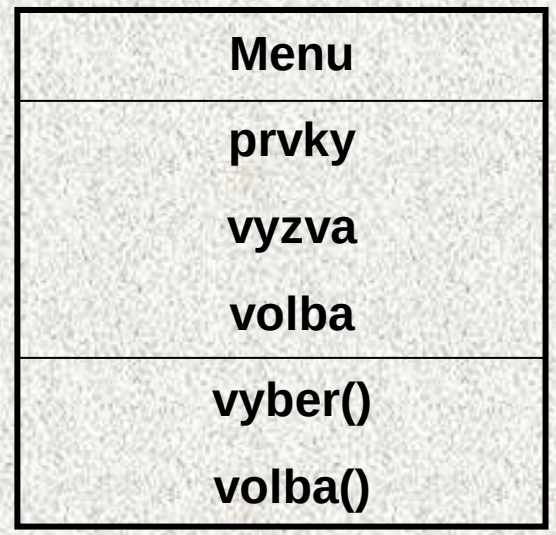
- Menu zavedeme jako datový typ s operacemi

  vyber a *volba*

  a s datovými položkami

  prvky, vyzva a *volba*

- Grafické vyjádření:



Třída Menu

```
public class Menu {
  String[] prvky;
  String vyzva;
  private int volba;
  public Menu(String[] prvky, String vyzva) {
    this.prvky = prvky;
    this.vyzva = vyzva;}
  public int vyber() {
      Scanner sc = new Scanner(System.in);
      do {
      for (int i=0; i<prvky.length; i++)
        System.out.println(prvky[i]);
      System.out.print(vyzva);
      volba = sc.nextInt();
      if (volba<0 || volba>=prvky.length) {
        System.out.println("Nedovolená volba");
        volba = -1;
      }
    } while (volba<0);
    return volba;}
  public int volba() {
    return volba;
  }}
```

Čítač a Menu jako objekt

```
public class Citac4 {
public static void main(String[] args) {
    String[] nabidka = {
      "0) Konec", "1) Zvětšit", "2) Zmenšit", "3) Nastavit"
    };
    Citac citac = new Citac(0);
    Menu menu = new Menu(nabidka, "Vaše volba: ");
    do {
      System.out.println("Hodnota = "+citac.hodnota());
      switch (menu.vyber()) {
        case 1: citac.zvetsit(); break;
        case 2: citac.zmensit(); break;
        case 3: citac.nastavit(); break;
      }
    } while (menu.volba()!=0);
    System.out.println("Konec");
  }
}
```

Třída MenuX jako potomek třídy Menu - dědění

1. zdědí od třídy Menu:
   - všechny datové položky

     ```
     private String[] prvky;
     private String vyzva;
     private int volba;
     ```

   - všechny metody

     ```
     public int vyber();
     public int volba();
     ```

2. přidá
   - datovou položku
     - vytváření menu libovolné délky (**`private int pocetPrvku, volny; )`**
   - metody
     - přidávání položky menu (**`pridejPolozkuDoMenu(String s))`**
     - zadání libovolné výzvy (**`pridejVyzvu(String s))`**

Třída MenuX jako potomek třídy Menu

- vytváření menu libovolné délky (**pocetPrvku + volny)**
- přidávání jednotlivé položky menu (pridejPolozkuDoMenu(String s))
- zadání libovolné výzvy (pridejVyzvu(String s))

MenuX
prvky **vyzva** **volba** **pocetPrvku** **volny**
vyber() **volba()** **pridejPolozkuDoMenu(String s)** **pridejVyzvu(String s)**

Třída MenuX jako potomek Menu

```
class MenuX extends Menu{
//private String[] prvky;
//private String vyzva;
//private int volba;
private int pocetPrvku;
private int volny=0;
 public MenuX(String[] prvky, String vyzva,int pocet) {
    super(prvky, vyzva);
    this.pocetPrvku = pocet;
    prvky = new String[pocetPrvku];
  }
public void pridejPolozkuDoMenu(String s){
   if (volny < pocetPrvku)prvky[volny++]=s;
  }
public void pridejVyzvu(String s){
    vyzva=s;
        }
}
```

Třída Čítač 5

```
public class Citac5 {
  public static void main(String[] args) {
    Citac citac = new Citac(0);
    String[] nabidka = new String[6];
    MenuX menu = new MenuX(nabidka, "Vaše volba: ", 6);
    menu.pridejPolozkuDoMenu("0) Konec");
    menu.pridejPolozkuDoMenu("1) Zvětšit");
    menu.pridejPolozkuDoMenu("2) Zmenšit");
    menu.pridejPolozkuDoMenu("3) Nastavit");
    menu.pridejPolozkuDoMenu("4) nove");
     menu.pridejPolozkuDoMenu("5) XXX");
    menu.pridejVyzvu("Vaše volba: ");
    do {
      System.out.println("Hodnota = "+citac.hodnota());
      switch (menu.vyber()) {
        case 1: citac.zvetsit(); break;
        case 2: citac.zmensit(); break;
        case 3: citac.nastavit(); break;
        case 4: System.out.println("nic"); break;
      }
    } while (menu.volba()!=0);
    System.out.println("Konec");
  }}
```

Třída CitacGUI

- Vybudován s využitím knihovny SWING, viz příslušná přednáška
- Řešení viz příklady

Zásady objektového programování

- Jasné definování rozhraní
- Důsledné skrytí implementace, zapouzdření
- Využívání kompozice před děděním
- Atomizace činností metod
- Nezávislost metod na sobě
- Redukce duplicit

- Další zásady viz cvičení
- Tendence k návrhovým vzorům
  - http://objekty.vse.cz/Objekty/Vzory-uvod#CoPred
  - Rudolf Pecinovský: Návrhové vzory – 33 vzorových postupů pro objektové programování, Computer Press

Návrhové vzory

- návrhový vzor - ucelený opakující se prvek použitý při implementaci softwarových produktů
- návrhový vzor v architektuře - gotika

Průčelí budovy ČVUT, novorenesanční stavba, sochy Práce a Vědy

Katedrála sv. Víta

Návrhové vzory

návrhový vzor tedy usnadňuje a zefektivňuje návrh

- klient - server architektura (dvouvrstvá architektura)
- tří-, čtyřvrstvá architektura
- model 2 - Model-view-controller (MVC)

Klient - Server architektura

- klient

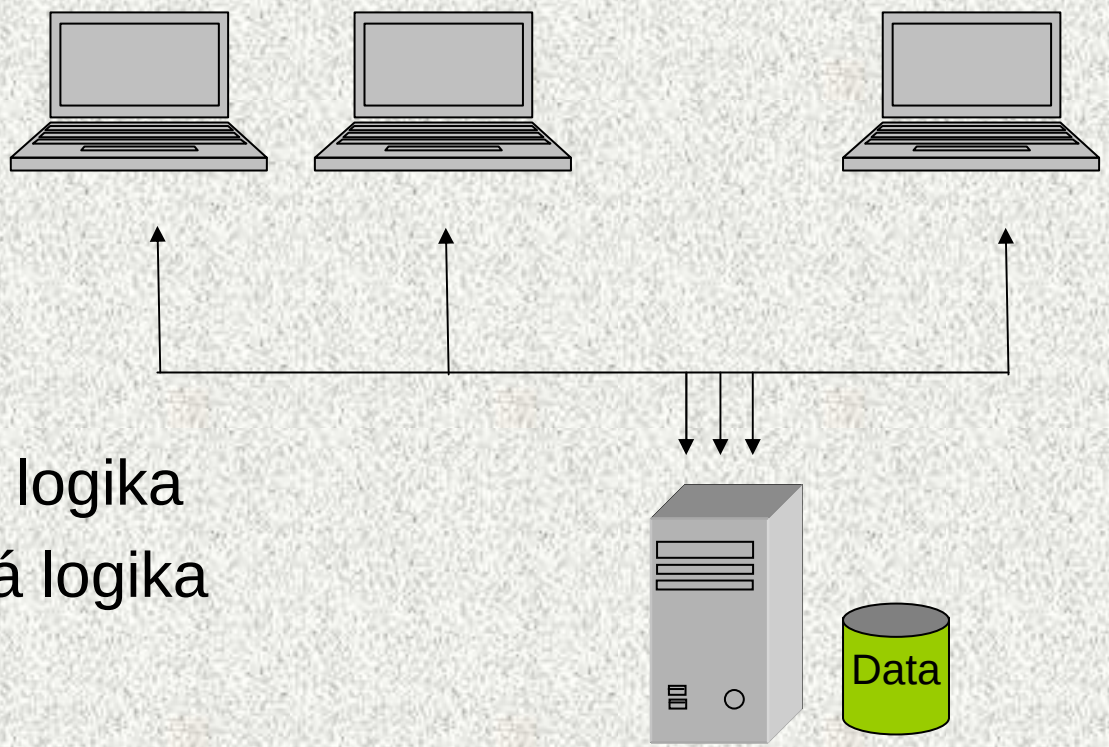
  program, který vysílá požadavky serveru

- server

program, který obsluhuje množství požadavků klientů

příklad

webový server



dvě vrstvy:

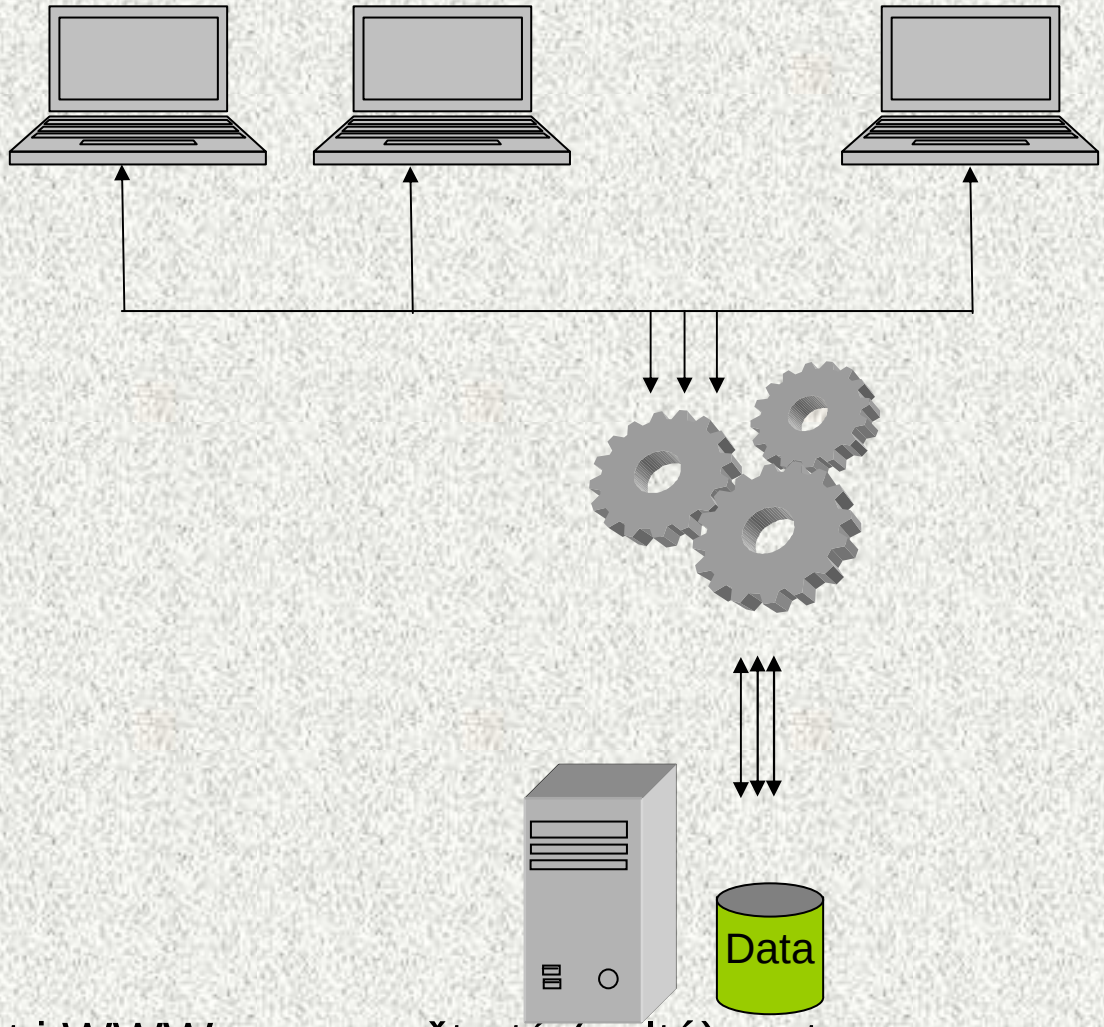
klient - prezentační logika

server - databázová logika

Třívrstvá architektura

Vrstvy:

1. prezentační logika



2. aplikační logika

3. databázová logika

Jedním z klientů může být i WWW server - čtvrtá (nultá) vrstva

Model 2 - Model-view-controller (MVC)

Komponenty

1. Model - datová (databázová) logika, veškeré informace popisující stav aplikace
2. View - pohled, zobrazení
3. Controller - řízení, řadič, reaguje na události (typicky pocházející od uživatele) a zajišťuje změny v modelu nebo v pohledu