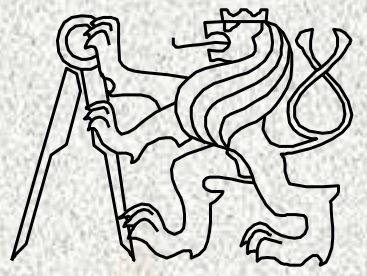
Třídy



A0B36PR1-Programování 1

Fakulta elektrotechnická
České vysoké učení technické

Třídy a objekty

Věci okolo nás lze hierarchizovat do tříd (konceptů).

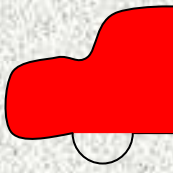
Každá třída je reprezentována svými prvky, objekty dané třídy

Každá třída je charakterizována svými vlastnostmi, svými funkčními možnostmi a svými parametry

Příklad:

Třída „automobil“

- **Funční možnosti automobilu – metody pro ovládání**
- **Parametry – charakteristická data**



Třída a objekty

definice

METODY: zrychlit, brzdit, zatočit, zastavit;…

TŘÍDA:
AUTO

DATA:jmeno,barva, maxRychlost, prumSpotreba, klimatizace

objekt: autíčko

objekt: závodní

objekt: auťák

objekt: beruška

instance

Třída Auto I – data + konstruktor

```
public class Auto {
    String barva;
    String jmeno;
    double prumerSpotreba;
    boolean klimatizace;

public Auto(String f_barva, String f_jmeno, double
  f_prumerSpotreba, boolean f_klimatizace){
        barva = f_barva;
        jmeno =f_jmeno;
        prumerSpotreba = f_prumerSpotreba;
        klimatizace = f_klimatizace;
}

static int pocetAut = 0;// proměnná třídy
```

data

konstruktor

Třída Auto II - metody

```
public void startuje(){
 System.out.println("Startuje auto " + jmeno);
 pocetAut++;
 System.out.println("Jede " + pocetAut + " aut");}
public void zrychlit(){
 System.out.println("Auto: " + jmeno + " zrychluje");}
public void zastavit(){
 System.out.println("Auto: " + jmeno + " zastavuje");
 pocetAut--;
 System.out.println("Jede " + pocetAut + " aut");}
public void brzdit(){
 System.out.println("Auto: " + jmeno + " brzdí");}
public void zatocit(){
 System.out.println("Auto: " + jmeno + " zatáčí");}
public double prumSpotreba(){return prumerSpotreba;}
public double spotreba(int pocetKM){
         return pocetKM * prumerSpotreba;}
static void informace(String napis){
System.out.println("Informace:" + napis);
}
```

metody

Vytvoření objektů a volání metod třídy Auto

```
Auto.informace("Závody");
Auto a = new Auto("modrá", "autíčko", 6, true);
a.startuje();
System.out.println("Max.spotreba “ + a.prumSpotreba());
int pocetKM = 45;
System.out.println("Spotreba "+ a.jmeno + " je " +
                                    a.spotreba(pocetKM));
Auto b = new Auto("žlutá", "závodní", 66, false);
b.startuje();
System.out.println("Max. spotreba " +b.jmeno + " je "
                                    b.spotreba(66));
Auto c = new Auto("červená", "beruška", 66, false);
c.startuje();
a.zrychlit();
b.zrychlit();
Auto d = new Auto("stříbrná", "auťák", 66, false);
d.startuje();
a.zastavit();
c.zatocit();
c.zastavit();
```

Vytvoření objektů a volání metod třídy Auto -příklad

Informace: Závody
Startuje auto autíčko
Jede 1 aut
Max. spotreba 6.0
Spotreba autíčko je 270.0
Startuje auto závodní
Jede 2 aut
Max. spotreba závodní je 4356.0
Startuje auto beruška
Jede 3 aut
Auto: autíčko zrychluje
Auto: závodní zrychluje
Startuje auto auťák
Jede 4 aut
Auto: autíčko zastavuje
Jede 3 aut
Auto: beruška zatáčí
Auto: beruška zastavuje
Jede 2 aut
>>> STOP

Třída jako programová jednotka k řešení problému

- Třída v jazyku Java je programová jednotka tvořená množinou identifikátorů, které mají třídou definovaný význam:
  - data – proměnné, konstanty (členské proměnné, datové složky, atributy)
  - metody - funkce a procedury

zapouzdření

- Takto koncipovaná třída je zdrojem funkcí popisujících řešení problému rozkladem na podproblémy

Jiný pohled na třídu

- Základem uživatelského programu v jazyku Java je třída, ve které
  - musí být deklarována hlavní funkce **`main`**
  - mohou být deklarovány další funkce (procedury) třídy
  - mohou být deklarovány statické proměnné, které jsou použitelné jako nelokální proměnné ve funkcích dané třídy

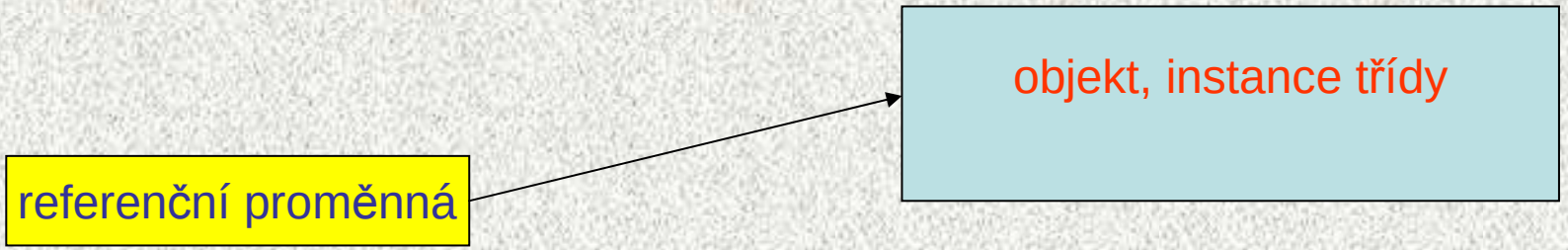
Třída jako zdroj funkcí

- Třída nemusí obsahovat deklaraci hlavní funkce *main*; třída bez hlavní funkce *main* nepředepisuje program, který lze spustit, ale zavádí prostředky, které lze v jiných třídách využít – „knihovnu“
- Příkladem je:
  - knihovní třída *Math* poskytující matematické funkce
  - třída *Scanner* poskytující jednoduché funkce pro vstup a výstup
  - třída Auto

- Poznámka:
  - Třída s hlavní funkcí main tvořící základ programu je specialitou jazyka Java
    - v jiných jazycích, např. v C++, lze program vytvořit bez použití třídy

Třída jako datový typ

- Obecně **je třída popisem strukturovaného datového typu**, tzn. specifikuje
  - množinu hodnot datových objektů skládajících se ze složek (datových)
  - množinu operací s datovými objekty
- Třída je „šablona“ - vzor pro vytváření jednotlivých objektů (instancí)
- **Datové objekty typu třída** (zkráceně jen objekty) se nazývají též instancemi třídy
- V jazyku Java lze objekty (instance tříd) vytvářet ***pouze dynamicky*** pomocí operátoru **new** a přistupovat k nim pomocí referenčních proměnných (podobně jako u pole)



Objekty, základní pojmy

- Objekt - datový prvek, *instance třídy,* dynamicky vytvořen podle „vzoru“- třídy
- Objekt (resp. data) je strukturován tzn. skládá se z jednotlivých položek

Pozn. 1: Třída bez vytvořené instance (objektu) nemůže „pracovat“ (mohou být použity jen její statické metody či proměnné), instance musí být vytvořena pomocí **new**

Pozn. 2: Existují dva druhy neprimitivních datových typů (referencované):

- pole ~ homogenní objekt skládající se z položek stejného typu
- objekty ~ heterogenní objekt skládající se z položek různého typu

Pozn.3:Položky objektu označujeme též jako

- členské proměnné (member variables)
- datové složky
- atributy objektu

Příklad: třída Complex

- Příkladem třídy jako datového typu je třída *Complex*
  - hodnotami typu jsou komplexní čísla tvořená dvojicemi čísel typu *double* (reálná a imaginární část)
  - množinu operací tvoří obvyklé operace nad komplexními čísly (absolutní hodnota, sčítání, odčítání, násobení a dělení)

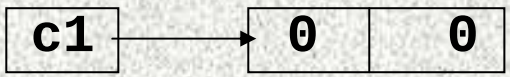
Třída Complex

```
  public class Complex {
// datové složky
  public double re; public double im;
// konstruktory
  public Complex() {}
  public Complex(double r, double i) {re=r; im=i;}
// metody (operace)
  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+"]";}
```

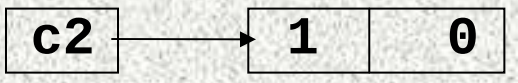
Konstruktory – vytvoření a inicializace objektu

- Třída *Complex* umožňuje vytvořit a inicializovat objekt třemi způsoby:

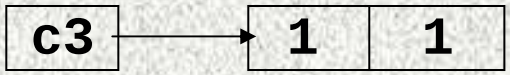
```
Complex c1 = new Complex();
```




```
Complex c2 = new Complex(1);
```




```
Complex c3 = new Complex(1,1);
```



- Tyto tři způsoby inicializace objektu jsou dány třemi konstruktory třídy *Complex*
  - konstruktor bez parametrů *Complex*() inicializuje vytvořený objekt hodnotami 0,0
  - konstruktor s jedním parametrem *Complex*(*double re*) inicializuje vytvořený objekt hodnotami *re*,0
  - konstruktor se dvěma parametry *Complex*(*double re, double im*) inicializuje vytvořený objekt hodnotami *re* a *im*

Instanční metody

- Operace s objekty se realizují pomocí instančních metod (dále jen metod)
- Metody mohou mít parametry a mohou vracet výsledek nějakého typu
- Volání (aplikace) metody *m* na objekt referencovaný proměnnou *p* má tvar:
  - p.m(<*seznam skutečných parametrů*>)

Zkráceně budeme říkat „metoda *m* se volá na objekt *p*“

Třída Complex

```
public static void main(String[] args) {
    Complex c1=new Complex(3,4); // konstrukce objektu
    Complex c2=new Complex(1,1); // konstrukce objektu
    System.out.println(new Complex());
    System.out.println(c1);
    System.out.println(c1.abs());  // |c1|
    System.out.println(c1.plus(c2)); // c1 + c2
  }
}
```

```
[0.0, 0.0]
[3.0, 4.0]
5.0
[4.0, 5.0]
```

Instanční metody II

- Příklady metod definovaných třídou *Complex* pro objekty typu *Complex*:
- **`double abs()`**
  - výsledkem volání **`c.abs()`** je absolutní hodnota komplexního čísla *c*
- **`Complex plus(Complex y)`**
  - výsledkem volání **`c1.plus(c2)`** je reference na nový objekt typu *Complex*, jehož hodnotou je součet komplexních čísel *c*1 a *c*2

Poznámka:

viz PR2

- **`String toString()`**
  - je metodou každého objektu, nepřekreje-li se, pak jméno třídy + hash kód např. alg7.Auto@1f6a7b9
  - výsledkem volání **`c.toString()`** je řetězec tvořící znakovou reprezentaci komplexního čísla *c* (metodou je zavedena implicitní typová konverze z typu *Complex* na typ *String*), často se překrývá!

Statické versus instanční metody

- Třída může definovat dva druhy metod:
  - statické metody – metody třídy, procedury a¨funkce
  - instanční metody – metody objektů
- Metody obou druhů mohou mít parametry a mohou vracet výsledek nějakého typu
- Statická metoda označuje operaci (dílčí algoritmus, řešení dílčího podproblému), jejíž vyvolání (provedení) obsahuje jméno třídy, jméno metody a seznam skutečných parametrů
  - jméno_třídy.jméno_metody(seznam skutečných parametrů)

  Statickým metodám třídy odpovídají v jiných jazycích procedury (nevracejí žádnou hodnotu) a funkce (vracejí hodnotu nějakého typu)
- Instanční metoda označuje operaci nad objektem (instancí (!)) dané třídy, jejíž vyvolání obsahuje referenční proměnnou objektu, jméno metody a seznam skutečných parametrů
  - *referenční_proměnná.jméno_metody*(*seznam skut. parametrů*)
- Statickým metodám třídy budeme i nadále říkat procedury a funkce
- Instančním metodám budeme zkráceně říkat metody

Struktura objektu

- Hodnota objektu je strukturovaná, tzn. skládá se dílčích hodnot, které mohou být obecně různého typu (heterogenní datová struktura – na rozdíl od pole)
- Objekt je tedy abstrakcí paměťového místa skládajícího se z částí, ve kterých jsou uloženy dílčí hodnoty - nazývají se položkami objektu (složkami, atributy, instančními proměnnými, fields, attributes)
- Položky objektu jsou označeny jmény, která mohou (ale nemusí) být třídou zveřejněna
- Příklad: objekty (instance) třídy *Complex* jsou tvořeny dvěma položkami typu *double*, jejichž jména jsou *re* a *im* a třída tato jména zveřejňuje
- Veřejnou položku se jménem ***f*** objektu, který je referencován proměnnou ***p***, lze označit zápisem

  p.f

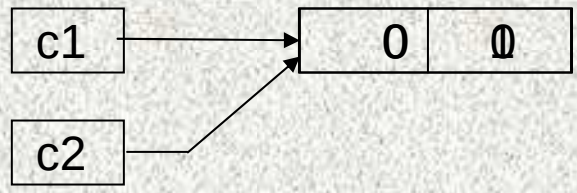
Příklad: objektu vytvořenému deklarací

```
Complex c = new Complex();
c.re = 1;
c.im = 1;
```

Více referencí na jeden objekt

- Objekt může být referencován i více referencemi
- Jestliže hodnotu referenční proměnné typu *T* přiřadíme jiné referenční proměnné téhož typu, pak obě proměnné referencují tentýž objekt
- Příklad:

```
Complex c1 = new Complex();
Complex c2 = c1;
c1.im = 1;
System.out.println(c2.im);
// vypíše 1
```

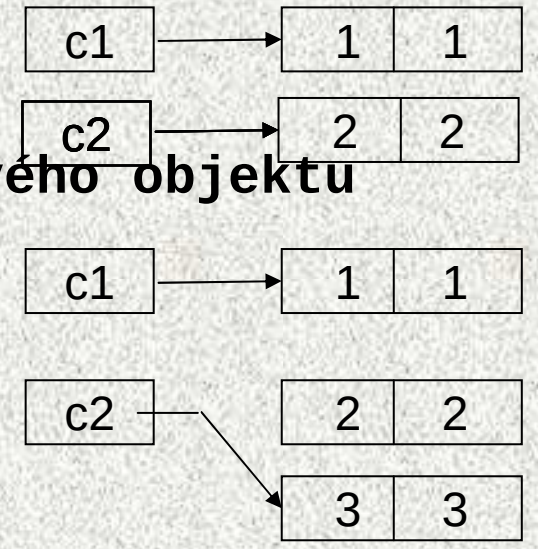


Sbírání smetí

Objekt se stane „smetím“, není-li přístupný pomocí žádné reference

```
Complex c1 = new Complex(1,1);
Complex c2 = new Complex(2,2);

c2 = c2.plus(c1); // vznik nového objektu
```



Paměť přidělená nepřístupným objektům se uvolňuje automaticky (sbírání smetí, garbage collection)

Třída – 2. příklad

Třída je návrhový vzor - definuje vlastnosti a chování

vlastnosti …. atributy

chování …. metody (funkce a procedury)

Příklad obdélník - popisuje objekty „reálného světa“

atributy (vlastnosti):

- šířka,
- výška,
- barva,
- pozice na obrazovce, …

metody (chování, reakce na požadavky okolí)

- nastavení barvy,
- výpočet obvodu, obsahu,
- posunutí, …

Obdélník - příklad definice třídy

```
public class Obdelnik {
    Color barva;
    int sirka, vyska;
    Point pozice;
    int vypoctiObvod(){
        return 2*(sirka+vyska);
    }
    int vypoctiObsah(){
        return sirka*vyska;
    }
    void nastavBarvu(Color c){
        barva = c;
    }
}
```

Jméno třídy začíná velkým písmenem

Atributy objektu - definují typ a jména vlastností

Metody objektu - definují chování, schopnosti, reakce

Speciální metoda - konstruktor třídy

```
public class Obdelnik {
    …
  Obdelnik(int s, int v){
     sirka = s;
     vyska = v;
    }
}
```

Konstruktor:

- tato metoda vytvoří objekt
- nastavuje vlastnosti objektu
- jméno je totožné se jménem třídy (jediná metoda začínající velkým písmenem)
- volání pomocí operátoru **new,** např.

```
malyObdelnik = new Obdelnik(2,5);
```

- neosahuje návratový typ - nic nevrací, vytváří objekt

Třída versus objekt

Třída

návrhový vzor, šablona

reprezentovaná zápisem v Javě, existuje i mimo program

Obdelnik

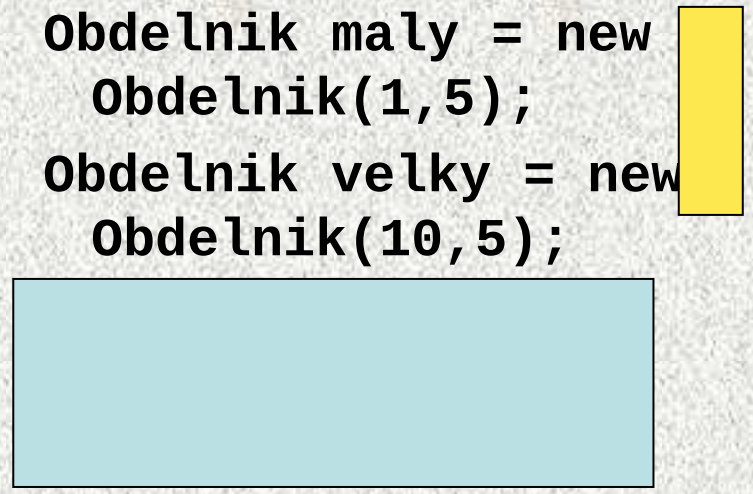
Clovek

Objekt

jeden konkrétní výrobek vyrobený podle třídy

vytvořen za běhu programu, žije během života programu (lze jej uložit na disk, není reprezentován kódem programu

```
Obdelnik maly = new
  Obdelnik(1,5);
Obdelnik velky = new
  Obdelnik(10,5);
```



„František Navrátil“

„Julie Capuletová“

Třída a metoda main - opakování

- Základem aplikace, tj. uživatelského programu, je třída, ve které
  - je deklarována spouštěcí metoda přesně takto:

```
public static void main( String[ ] args ) {
  ... }
```

Metoda ***main*** *musí být statická, voláme ji dříve než se vytvoří nějaký objekt*

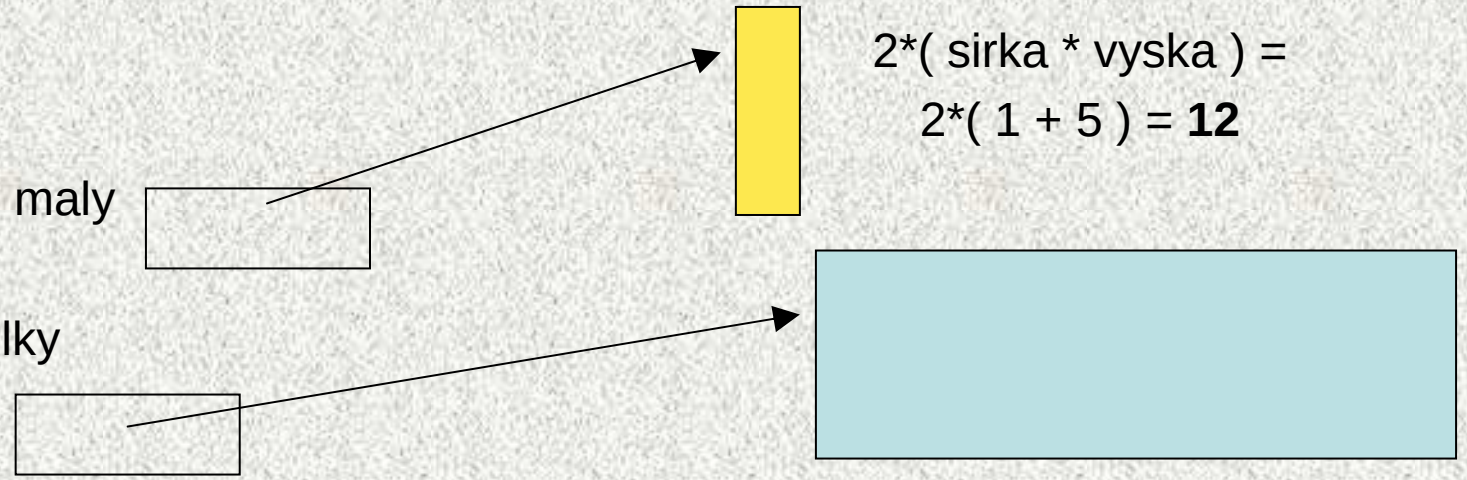
Každá třída může obsahovat metodu **main,** pak se využívá pro:

- testování funkčnosti objektu
- ukázku použití metod objektu

Třída pro testování obdélníka

```
public class ObdelnikTest {
public static void main(String
 Obdelnik maly;
 maly =  new Obdelnik(1,5);
 Obdelnik velky = new Obdelnik(10,5);
 System.out.println("Obvod maleho je "+
                                maly.vypoctiObvod());
    }
}
```

Není třeba předávat šířku a výšku, každá instance (objekt) zná své rozměry!!



Statické metody

```
public class Obdelnik {
   int sirka ….
   static int vratPocetRohu(){
   // sirka = 6;
  //CHYBA,statická metoda nevidí instanční proměnné
         return 4;
     }
}
```

```
public class ObdelnikTest {
    public static void main(String[] args) {
      Obdelnik maly =  new Obdelnik(1,5);
      System.out.println("Pocet rohu obdelnika je
      "+Obdelnik.vratPocetRohu());
      System.out.println("Maly obdelnik ma
      "+maly.vratPocetRohu()+ " rohu.");
       }
}
```

4

4

Je to možné, ale nelogické!!!

Statické atributy a metody

Některé třídy obsahují pouze statické atributy a statické metody.

Knihovna matematických funkcí - třída **`java.lang.Math`** obsahuje

- Příklad je i třída Complex bez metody `main`

statické proměnné (zde konstanty typu **double**) **PI** a **E**

Statické atributy a metody - Math

statické metody reprezentující matematické funkce:

double x = **Math.sin**(0,5);

- **sin, cos, tan, …** goniometrické funkce
- **abs …** absolutní hodnota
- **min, max**
- **log …** logaritmus
- **sqrt …** odmocnina
- **pow(double a, double b) … a^b**
- **random …** vrací náhodné **double** číslo z intervalu <0;1)
- **round …** zaokrouhlení
- a mnohé další

$$a^{\frac{1}{3}} = \sqrt[3]{a}$$