A0B36PRI - PROGRAMOVÁNÏ

Objektově orientované programování

Jazyk JAVA

České vysoké učení technické Fakulta elektrotechnická

v1.01

Obsah

- Objektově orientované programování - úvod
- Třída
- Obdélník - příklad definice třídy
- Obdélník - metody
- Speciální metoda - konstruktor třídy
- Třída versus objekt
- Třída a metoda main
- Třída pro testování obdélníku
- Statické a instanční metody
- Statické metody
- Statické atributy a metody
- Přetěžování konstruktorů
- Standardní metody
- Zastínění metod předka
- Více referencí na jeden objekt, smetí
- Třída String

Objektově orientované programování - úvod

- Základní principy
  - Objekty jsou instancemi svých tříd
  - Třídy vytvářejí hierarchický systém
  - Zapouzdření (Encapsulation)
    - objekt navenek zpřístupňuje *rozhraní,* v objektu jsou uložena data a metody, které objekt a jeho chování jednoznačně popisují
  - Dědičnost (Inheritance)
    - objekty mohou *dědit* vlastnosti, data i metody od nadřazených objektů objektů, ke kterým přidávají svoje vlastní rozšíření.
  - Polymorfismus (Polymorphysm)
    - odkazovaný objekt se chová podle toho, jaké třídy je instancí

Třída a objekty

METODY: zrychlit, brzdit, zatočit, zastavit;…

TŘÍDA:
AUTO

DATA:jmeno,barva, maxRychlost, prumSpotreba, klimatizace

objekt: autíčko

objekt: závodní

objekt: auťák

objekt: beruška

Třída - základní pojmy, opakování

- Třída (`class`):
  - data (členské proměnné, konstanty) >>> určují stav objektu
  - metody (podprogramy a funkce) >>> určují schopnosti objektu

- Třída - *jiný pohled*
  - datový, objektový typ,
  - šablona pro generování konkrétních *instancí* třídy, tzv. *objektů*

- Jednotlivé instance třídy (objekty) mají stejné metody, ale nacházejí se v různých stavech

- Stav je dán obsahem členských proměnných

Třída

- Třída je návrhový vzor, který umožňuje definovat 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;
     }
}
```

- nastavuje vlastnosti objektu
- neosahuje návratový typ - nic nevrací, vytváří objekt
- jméno je totožné se jménem třídy (jediná metoda začínající velkým písmenem)
- volání pomocí operátoru **new,** např. malyObdelnik = new Obdelnik(2,5);
- tato metoda vytvoří objekt

Třída Complex

```
public class Complex {
    // datové složky
    public double re; public double im;
    // konstruktory
    Complex
    // 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+"]";}
```

Třída PoužitíComplex

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

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

Třída versus objekt

Třída

návrhový vzor, šablona

reprezentovaná zápisem v Javě, existuje i mimo program

Obdelnik

Clovek

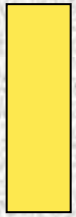
Objekt

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);

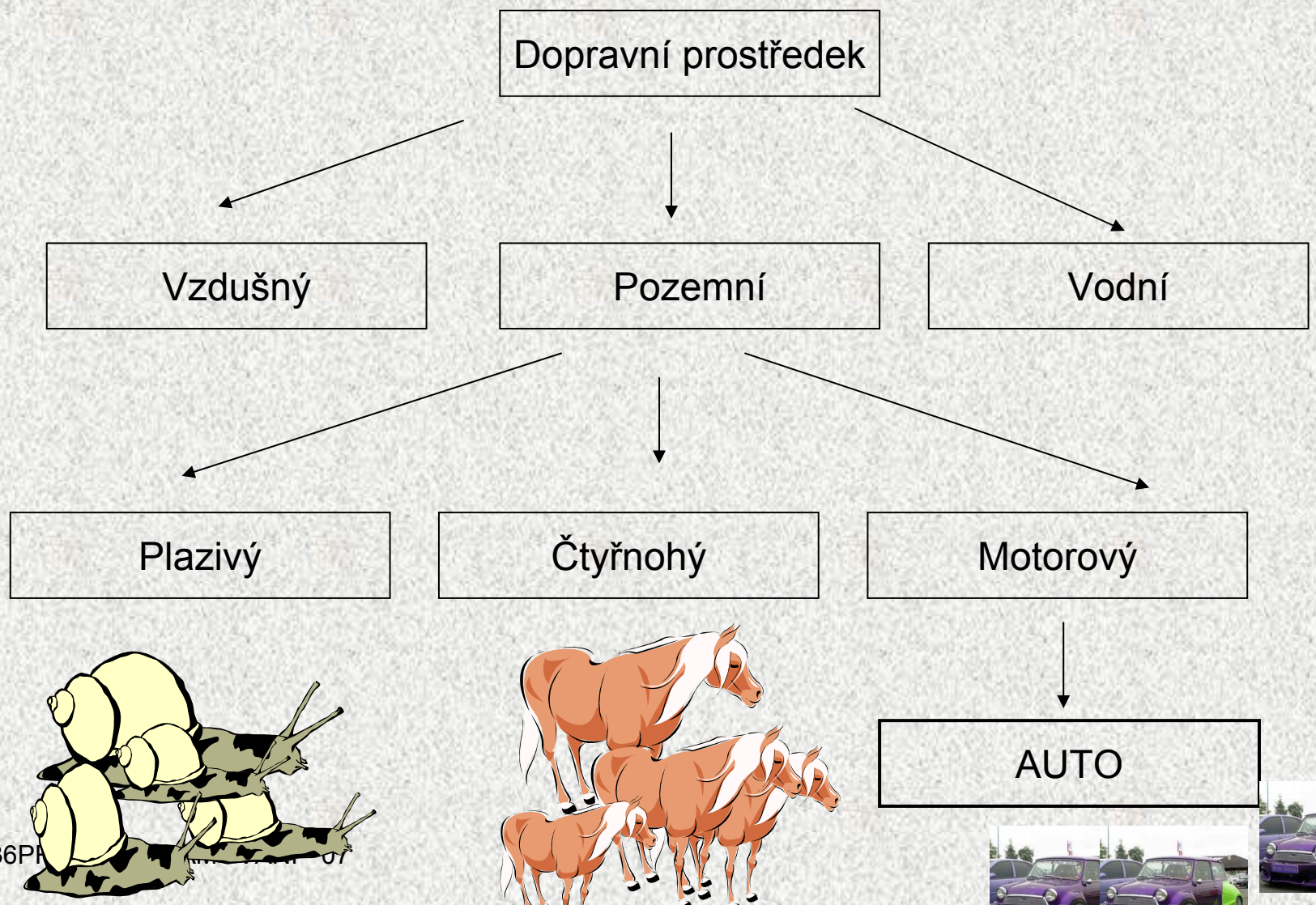
Obdelnik **velky** = new Obdelnik(10,5);



„František Navrátil“

„Julie Capuletová“

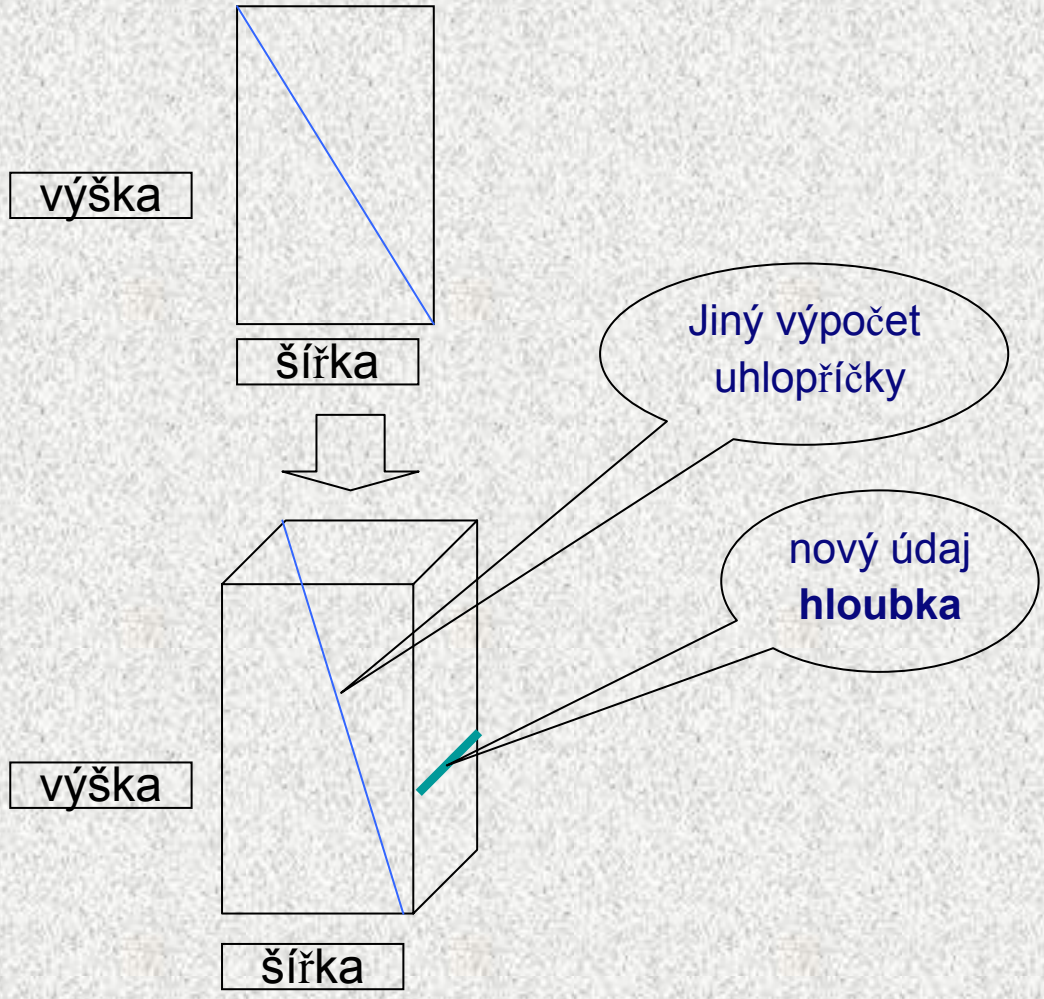
Třídy a dědičnost



Dědičnost – příklad

Obdelnik
sirka;
vyska;
delkaUhlopricky()
hodnotaSirky()

Kvadr extends Obdelnik
hloubka
sirka;
vyska;
delkaUhlopricky()
hodnotaSirky()



Dědičnost – základní vlastnosti

- Mechanismus umožňující
  - rozšiřovat datové položky tříd
  - rozšiřovat či modifikovat metody tříd

- Dědičnost umožní
  - vytvářet hierarchie tříd
  - „předávat“ datové položky a metody k rozšíření a úpravě
  - specializovat, „upřesňovat“ třídy

- Dědění má praktický význam v znuvupoužitelnosti programového kódu

- Dědičnost je základem polymorfismu

Dědičnost – příklad

```
class Obdelnik {                                    // Kvadr.java
  public int sirka;
  public int vyska;
  public Obdelnik(int sirka, int vyska) {
    this.sirka = sirka;
    this.vyska = vyska;
  }
  public double delkaUhlopricky() {
    double pom;
    pom = (sirka * sirka) + (vyska * vyska);
    return Math.sqrt(pom);
  }
  public int hodnotaSirky() {
    return sirka;
  }
}
```

Dědičnost – příklad

```
public class Kvadr extends Obdelnik {
public int hloubka;

public Kvadr(int sirka, int vyska, int hloubka) {
super(sirka, vyska); // volání konstruktoru předka
this.hloubka = hloubka;
  }

public double delkaUhlopricky() { // překrytí metody předka
double pom = super.delkaUhlopricky();// volání metody předka
pom = (pom * pom) + (hloubka * hloubka);
return Math.sqrt(pom);
  }
```

Dědičnost – příklad

```
public static void main(String[] args) {
Obdelnik obd = new Obdelnik(6, 8);
Kvadr    kva = new  Kvadr(6, 8, 10);
System.out.println("Uhlopricka: "+ obd.delkaUhlopricky());
System.out.println("Uhlopricka: "+ kva.delkaUhlopricky());
System.out.println("Sirka je: " + kva.hodnotaSirky());
System.out.println("Vyska je: " + kva.vyska);
} }
```

Uhlopricka: 10.0
Uhlopricka: 14.14
Sirka je: 6
Vyska je: 8

Dědičnost - komentář

- Kvádr s "nulovou" hloubkou je obdélník, obdélník po rozšíření je „kvádrem“
- Potomek se deklaruje pomocí klauzule `extends`
  - `Kvadr` převezme proměnné `sirka` a `vyska`, metodu `hodnotaSirky`
  - Konstruktor se nedědí, ale může být v podtřídě využit, je volán slovem `super`
    - jako první příkaz v podřízeném konstruktoru, s parametry rodiče!
  - Není-li konstruktor volán přímo, je volán konstruktor bez parametrů!!!
    - musí tedy existovat, ať už implicitní či uživatelský!!!
- Potomek doplňuje novou proměnou `hloubka` a mění metodu `delkaUhlopricky`
- Objekty `Kvadr` mohou využívat proměnné `sirka`, `vyska` a `hloubka`, metody `hodnotaSirky` a vlastní `delkaUhlopricky`
- Metoda `delkaUhlopricky` třídy `Kvadr` překrývá (overriding, zastínění) metodu téhož jména v rodičovské třídě
- Jsou-li jiné parametry (jiný počet, jiný typ), jedná se o tzv. přetížení - overloading) – jedná se tedy o jinou, novou metodu!!

Dědičnost – shrnutí

Vlastnosti dědění v Javě:

- opakovatelné využití programových částí (atributů (členských proměněných) a metod)
- **možnosti**
  - **využití atributů či metod rodiče,**
  - **upřesnění metod (i atributů) modifikací**
  - **přidání dalších atributů a metod**
- vytváření hierarchie objektů, postupně více specializovaných
  - specializace chování objektů !
  - existuje kompatibilita děděných objektů směrem "nahoru"

Poznámky:

- Východisko polymorfismu
- Jazyk Java zná dědění od jediného předka!
- potřeba zdědit vlastnosti od více předků se řeší rozhraním (`interface`) – viz další přednášky

Třída a metoda main

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

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

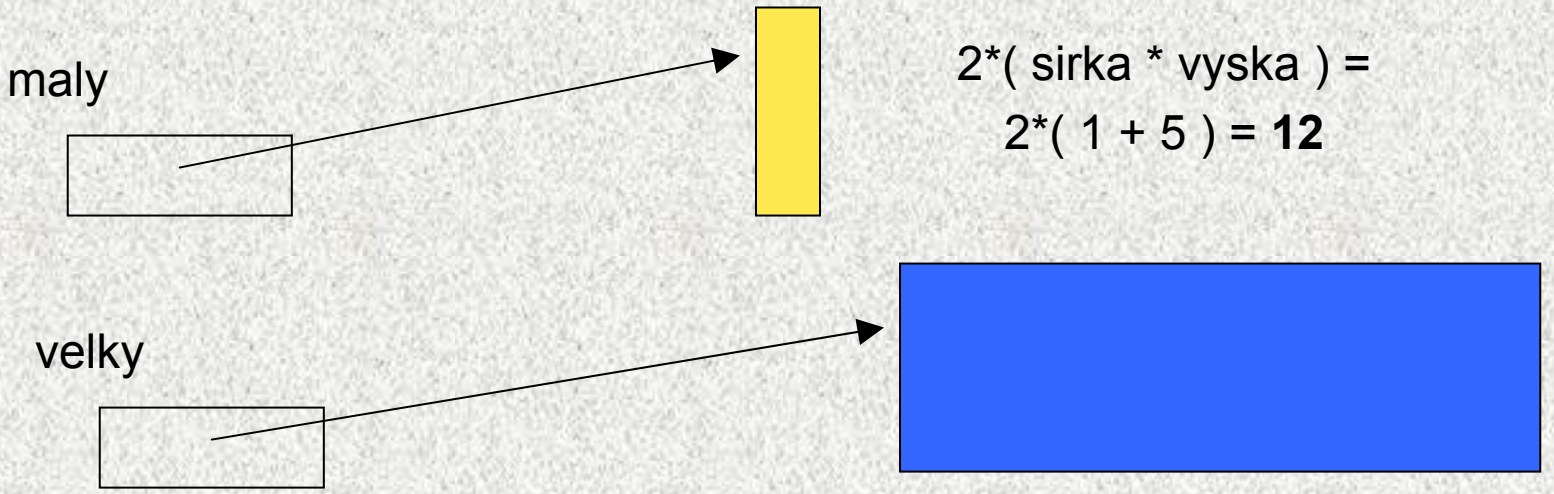
- Metoda ***main*** musí být statická, voláme ji dříve než se vytvoří nějaký objekt

- Každá třída může obsahovat metodu **main,** pak se využívá pro:
  - **testování funkčnosti objektu**
  - **ukázku použití metod objektu**

Třída pro testování obdélníku

```
public class ObdelnikTest {
    public static void main(String[] args) {
        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 a instanční metody

- Pod pojmem metoda se skrývají dva druhy metod:
  - **instanční** (objektové) metody (metody objektů)
  - **statické** metody (metody třídy)

- Oba druhy mohou mít parametry a mohou vracet výsledek
- **Instanční metoda** označuje operaci nad objektem čili instancí dané třídy.
  Je dostupná jen přes referenci na objekt. Voláme ji tedy takto:
  ***referenčníProměnná.jménoMetody*(*seznam argumentů*)**

  např.: **maly.vypoctiObvod();**
  „vidí“ statické i nestatické atributy třídy

- **Statická metoda** je dostupná pomocí jména třídy – není třeba vytvářet objekt. Voláme ji takto:

  JménoTřídy.jménoMetody(seznam argumentů)

  nebo

  referenčníProměnná.jménoMetody(seznam argumentů)

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");
    }
}
```

Statické atributy a metody

- Některé třídy obsahují pouze statické atributy a statické metody.
- Knihovna matematických funkcí - třída **java.lang.Math** obsahuje statické proměnné (zde konstanty typu *double*) **PI** a **E**

- Statické metody reprezentující matematické funkce:
  - **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

Přetěžování konstruktorů

- Konstruktor je metoda, která vytvoří objekt a nastaví jeho počáteční hodnoty.
- Vytvořme 3 typy konstruktorů pro Obdelnik:
  - Bez parametrů - vytvoří obdélník o stranách 0x0,
  - S jedním parametrem - vytvoří čtverec,
  - Se dvěma parametry - sirka x vyska

```
public class Obdelnik {
    …
 Obdelnik(){
        sirka =  vyska =0;
    }
 Obdelnik(int a){
        sirka =  vyska = a;
    }
 Obdelnik(int s, int v){
        sirka = s;
        vyska = v;
    }
}
```

Přetěžování konstruktorů

- Vytvořme konstruktor, který u vytvářeného obdélníku specifikuje jeho barvu.

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

}
```

Stejný kód, správné by bylo použití jednoho místa pro tento kód - jednodušší opravy.

Vzájemné volání konstruktorů

- Vytvoříme jedem „univerzální“ konstruktor a ostatní jej budou volat

```
public class Obdelnik {
    …
Obdelnik(int s, int v){

        this(s,v,Color.BLACK);
    }

Obdelnik(int s, int v, Color c){
        sirka = s;
        vyska = v;
      barva = c;
    }
}
```

nastavení implicitní hodnoty

volání konstruktoru téže třídy

Standardní metody

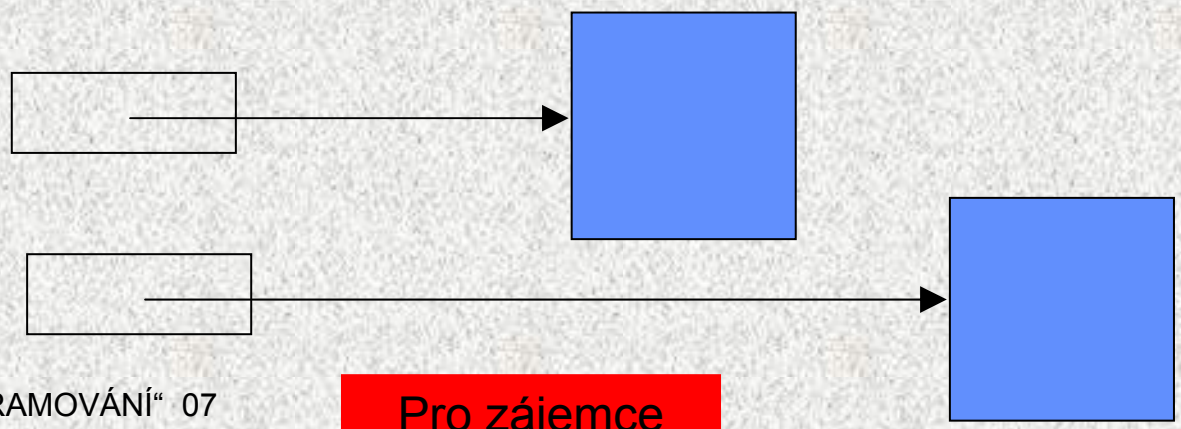
- Každá třída (kromě jediné) v Javě je potomkem třídy **Object,** která implementuje několik základních metod.
- Základní jsou metody **toString** a **equals.**

```
public class ObdelnikTest {
    public static void main(String[] args) {
        Obdelnik prvni = new Obdelnik(5,7);
        Obdelnik druhy = new Obdelnik(5,7);
        System.out.println("Prvni: " + prvni);
        System.out.println("Stejne? "+(prvni==druhy));
        System.out.println("Stejne? "+prvni.equals(druhy));
    }
}
```

Prvni: pri07.Obdelnik@11b86e7

false

false



Pro zájemce

Standardní metody třídy Object

```
public String toString(){
 return getClass().getName() + "@" +
                Integer.toHexString(hashCode());
}
```

- výsledkem volání *x.toString*( ) je řetěz znakové reprezentace objektu *x,*
- metodou je zavedena implicitní typová konverze z typu objektu *x* na řetězec, použitá např. při výpisu objektu

```
public boolean equals(Object obj) {
  return (this == obj);
    }
```

- standardní chování neporovnává položky

Pro zájemce

Zastínění metod předka

- Každá třída (kromě jediné) v Javě je potomkem třídy **Object,** která implementuje několik základních metod
- Základní jsou metody **toString** a **equals.**

```
public String toString(){
 return "Obdelnik: "+sirka+" x "+vyska;
}
```

```
public boolean equals(Object o){
 if(!(o instanceof Obdelnik))return false;
                                  // porovnam jen s obdelniky
 Obdelnik obd = (Obdelnik) o; //pretypovani
        return (sirka == obd.sirka)&&(obd.vyska==vyska);
}
```

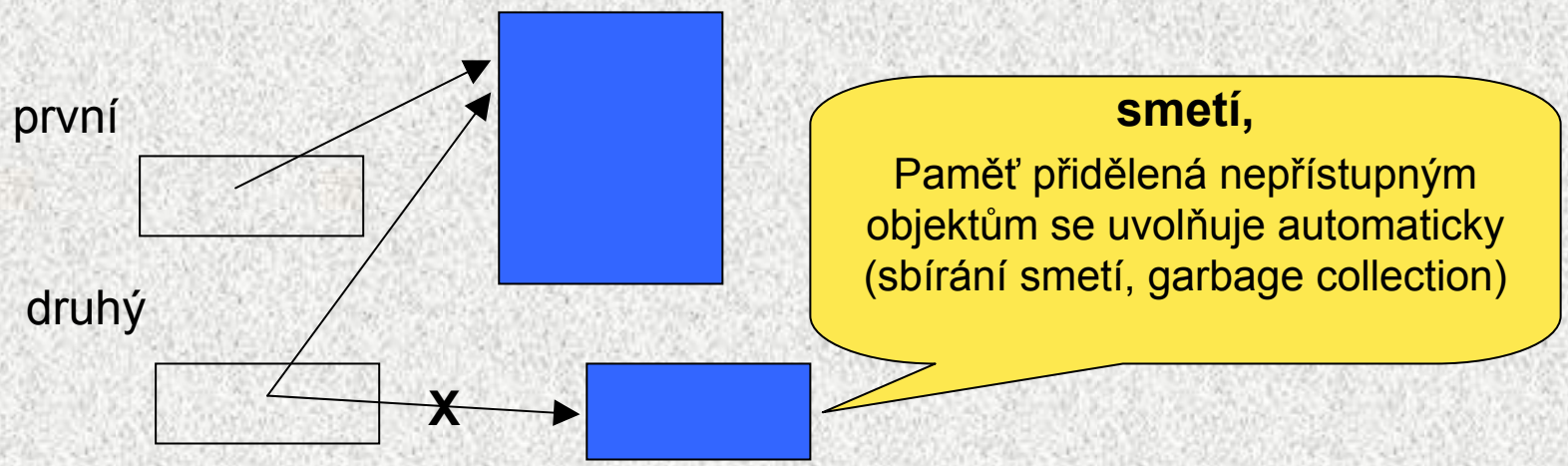

Pro zájemce

Více referencí na jeden objekt, smetí

```
public class ObdelnikTest {
    public static void main(String[] args) {
        Obdelnik prvni = new Obdelnik(5,7);
        Obdelnik druhy = new Obdelnik(5,2);
        druhy = prvni;
        System.out.println("Stejne? "+(prvni==druhy));
        System.out.println("Stejne? "+prvni.equals(druhy));
    }
}
```

true

true



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

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

Pro zájemce, pokračování z před. č.5

Třída Čítač

```
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;
    }
```

položky objektu (instanční proměnné)

konstruktor

instanční metody

Pro zájemce, pokračování z před. č.5

Použití čítače jako objektu

```
public static void main(String[] args) {
    int volba;
    Citac citac = new Citac(0);
     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;
      }
    } while (volba>0);
    System.out.println("Konec");
  }
}
```

Pro zájemce, pokračování z před. č.5

Hierarchie tříd

- V on-line dokumentaci jazyka Java týkající se třídy *ArrayList* najdeme následující obrázek:

```
java.lang.Object
    |
    +--java.util.AbstractCollection
        |
        +--java.util.AbstractList
            |
            +--java.util.ArrayList
```

- Tento obrázek vyjadřuje, že:
  - třída *ArrayList* (definovaná v balíku *java.util*) je podtřídou třídy *AbstractList*
  - třída *AbstractList* je podtřídou třídy *AbstractCollection*
  - třída *AbstractCollection* je podtřídou nejvyšší třídy java.lang.*Object*

Pro zájemce

A0B36PRI - PROGRAMOVÁNÏ

Objektově orientované programování

Jazyk JAVA

Konec

České vysoké učení technické Fakulta elektrotechnická