# Programování pro přírodovědce

## Objektovo orientované programovanie

Juraj Jašík

juraj.jasik@gmail.com
Univerzita Karlova v Praze
Přírodovědecká fakulta
Katedra organické a jaderné chemie

November 18, 2012

- klasické programovanie
  - identifikovať *úlohy* potrebné na vyriešenie problému
- objektovo orientované programovanie (OOP)
  - nájsť skupinu *objektov*, ktoré modelujú problém
- objekt
  - pomenovaná skupina premenných a podprogramov, ktoré tieto premenné používajú alebo s nimi manipulujú
  - objekt je *inštancia* triedy

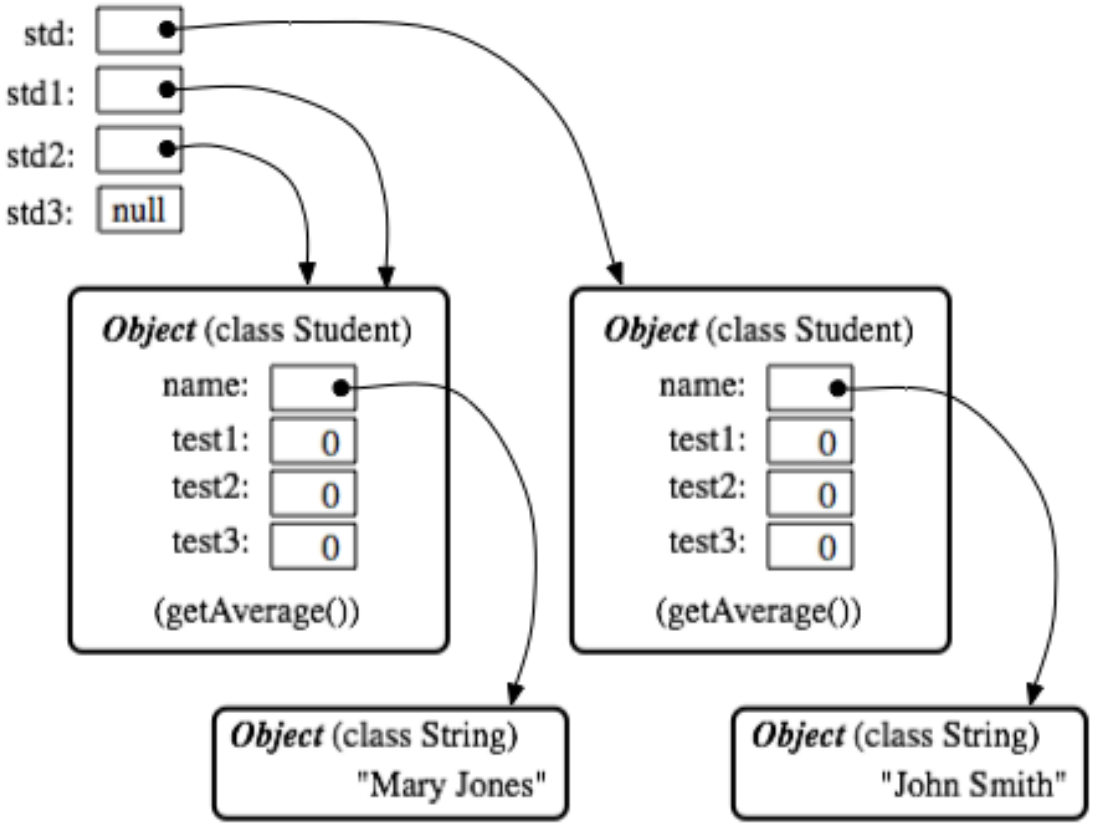
## Definícia triedy

```
public class Student {

    public String name;
    public double test1, test2, test3;

    public double getAverage() {
        return (test1 + test2 + test3) / 3;
    }
}
```

- v Jave je trieda *typ*, podobne ako `int`, `boolean`...
- meno triedy môže byť použité pri deklarácii premennej: `Student std;`
- ale pri deklarácii sme ešte nevytvorili objekt!
- premenná `std` neobsahuje objekt samotný ale len *referenciu* na objekt
- objekt je umiestnený v špeciálnom priestore pamät - *heap*
- premenná `std` obsahuje len informáciu potrebnú na nájdenie tohto objektu v pamäti

- objekt vytvoríme pomocou operátora `new`: `Student std = new Student();`
  - vytvorí objekt triedy `Student`
  - vráti *referenciu* (*pointer*) na tento objekt
  - premenná `std` ukazuje na tento objekt, hodnota `std` je referencia na tento objekt
- prístup k objektovým premenným: `std.name`, `std.test1`, ...
- prístup k metódam objektu: `std.getAverage()`
- premenná `std` nemusí ukazovať na žiaden objekt - `std = null;`
  - pri pokuse prístup k premenným alebo metódam - chyba null pointer exception

```
Student std, std1, std2, std3;
std = new Student();
std1 = new Student();
std2 = std1;
std3 = null;
```



# getter a setter

- modifikátor `public` - prístupné z ľubovoľného miesta
- modifikátor `private` - prístupné len z vnútra triedy
- dobrý zvyk: všetky premenné objektu deklarovať ako `private`
- pre čítanie hodnoty poskytnúť `public` metódu - *getter*

**getter**

```
private String title;

public String getTitle() {
   return title;
}
```

- pre možnosť zmeniť hodnotu poskytneme `public` metódu - *setter*

**setter**

```
public String setTitle(String newTitle) {
   title = newTitle;
}
```

- výhoda:
  - úplná kontrola objektu nad svojím stavom
  - getter a setter môžu vykonať komplexnejšie úlohy

```
public String getTitle() {
   titleAccessCount++;
   return title;
}

public String setTitle(String newTitle) {
   if ( newTitle == null )
      title = "(Untitled)";
   else
      title = newTitle;
}
```

# Konštruktor

```
Student std = new Student();
```

- úloha konštruktora:
  1. alokuje pamäť pre objekt
  2. inicializuje premenné objektu
  3. vráti referenciu na objekt
- každá trieda má konštruktor
- pokiaľ nezadefinujeme žiadny konštruktor, systém nám poskytne *implicitný (default) konštruktor*

- môžeme si napísať vlastný konštruktor (*explicitný*)
  - možnosť vykonania komplexných úloh pri inicializácii objektu
  - objektu môžeme predať informáciu potrebnú pre inicializáciu pomocou parametrov
  - v tomto prípade nám systém *neposkytne* implicitný konštruktor a objekt musíme vytvoriť volaním explicitného konštruktora

**definícia konštruktora**

```
public class Student {
   private String name;
   private double test1, test2, test3;

   public Student(String studentsName) {
      name = studentsName;
   }
}
```

- pri volaní explicitného konštruktora sa vykonajú kroky:
  1. alokuje sa pamäť pre objekt (na heape)
  2. inicializujú sa objektové premenné - pokiaľ je v deklarácii uvedený výraz pre počiatočnú hodnotu, tento výraz sa spočíta a premennej sa priradí hodnota
  3. ak sú zadané parametre konštruktoru, tieto sa vypočítajú a hodnoty sa priradia formálnym parametrom
  4. vykoná sa telo konštruktora
  5. konštruktor vráti referenciu na objekt

# Garbage Collection

- k objektu pristupujeme pomocou premenných, ktoré obsahujú referenciu na objekt
- môže sa stať, na objekt už neukazuje žiadna premenná

```
Student std = new Student("John Smith");
std = null;
```

- Java túto situáciu sleduje pomocou nástroja Garbage Collection - nepotrebné objekty uvoľňuje z pamäte automaticky

# Úlohy

1. simuluj kolko krát musíme hodiť dva páry hracích kociek, aby sme dosiahli rovnaké súčty bodov
2. vytvor triedu pre prácu s časovými údajmi (hh:mm:ss) - správne nastavovanie, spočítavanie, odpočítavanie, vypisovanie