

Objektovo orientované programovanie



Princíp **objektovo orientovaného programovania**:

- Definovať dátové typy:
Triedy, Atribúty, Objekty
- Objekty s vlastými operáciami (funkciami):
Metódami

1. Definícia dátových typov



Motivácia

- Dáta, ktoré logicky patria k sebe, definovať ako dátový typ (ďalšia fólia);

Dátový typ nazývame **trieda (class)**, konkrétne inštancie triedy nazývame **objekty**.

Dátový typ:

Príklad - Adresa



Adresa pozostáva z

- Meno
- Priezvisko
- Ulica
- Číslo
- PSČ
- Mesto



Atribúty
Objektov
Triedy Adresa

V Jave:



```
class Adresa
```

```
{ String meno = "";  
  String priezvisko = "";  
  String ulica = "";  
  String cislo = "";  
  String PSC = "";  
  String mesto = "";  
}
```

Atribúty
objektov
triedy sú
deklarované
ako (globálne)
premenné.

V Java (Pokračovanie):



- Dátový typ, t.j. trieda, definuje množinu objektov rovnakého typu.
- Dátový typ definuje mená a typy spoločných atribútov objektov.
- Dátový typ samotný má meno, ktoré je možné v programe používať pri definícii nových objektov daného dátového typu.

V Java (Pokračovanie):



- V Java je dátový typ definovaný pomocou kľúčového slova `class` nasledovaným menom dátového typu a deklaráciou všetkých atribútov.
- Deklarácie bez inicializácie sú povolené.

(Neskôr uvidíme, že **Trieda** môže mať aj funkcie, nazývané aj metódy.)

Telefónny zoznam



Definujme dátový typ „záznam v telefónnom zozname“, pre jednoduchosť ina „Zaznam“, ktorý pozostáva z mena, priezviska a telefónneho čísla:

```
class Zaznam
{ String meno = "";
  String priezvisko = "";
  String cislo = "";
}
```

Úplný příklad



```
import java.io.*;

class Zaznam
{ String meno = "";
  String priezvisko = "";
  String cislo = "";
}

public class Telefonny_zoznam
{
```


Úplný príklad

Tu je vytvorený nový objekt typu Zaznam!

```
public static Zaznam readZaznam()
{ Zaznam z = new Zaznam();

  z.meno      = readString("Zadaj meno");
  z.priezvisko = readString("Zadaj priezvisko");
  z.cislo     = readString("Zadaj telefonne cislo");

  return z;
}

// Funkcia readString je nami napísana
// funkcia na načítanie reťazca znakov
// z klávesnice (pozri ďalej).
```

Tu sú načítané atribúty záznamu!

Úplný príklad

```
public static void printZaznam(Zaznam z)
{ System.out.println(z.meno) ;
  System.out.println(z.priezvisko) ;
  System.out.println(z.cislo) ;
  System.out.println() ;
}
```

Tu vytvoríme pole pre sto záznamov – náš telefónny zoznam (ako lokálnu premennú).

```
public static void main(String[] pole_parametrov)
{ Zaznam[] zoznam = new Zaznam[100] ;
  int pocetZaznamov = 0 ;
  char c ;
```

Úplný príklad

```
do {
c = readChar („Zvol 1-na zadanie zaznamu, 2-na vyhladanie
zaznamu alebo 3-na vypis vsetkych zaznamov: ");
    switch (c)
    { case '1' : if ( pocetZaznamov < 100 )
                { zoznam[pocetZaznamov]= readZaznam();
                  pocetZaznamov++;
                }
        break;

// Funkcia readChar je nami napísana
// funkcia na načítanie reťazca znakov
// z klávesnice (pozri ďalej).
```

Úplný příklad

```
case '2' : String priezvisko = readString("Priezvisko:");
        for (int i = 0; i < pocetZaznamov; i++)
        { if (priezvisko.equals(zoznam[i].priezvisko))
          { printZaznam(zoznam[i]); }
        }
        break;
```

```
case '3' : for (int i = 0; i < pocetZaznamov; i++)
          { printZaznam(zoznam[i]); }
          break;
```

```
}
```

```
}
```

```
while ( c == '1' || c == '2' || c == '3');
```

```
}
```

Úplný příklad



```
public static String readString(String napis_pre_uzivatela)
{ String s;
  BufferedReader zklavesnice = new BufferedReader(new
InputStreamReader(System.in));
  try
  { System.out.println(napis_pre_uzivatela);
    s = zklavesnice.readLine();
    //System.out.println("Nacital som " + s);
  }
  catch (Exception e)
  { System.out.println("nepodarilo sa");
    s = readString(napis_pre_uzivatela);
  }
  return s;
}
```

Úplný příklad



```
public static char readChar(String napis_pre_uzivatela)
{ char c = ' ';
InputStreamReader zklavesnice = new InputStreamReader(System.in);
try
{ System.out.println(napis_pre_uzivatela);
  c = (char) zklavesnice.read();
  //System.out.println("Nacital som " + c);
}
catch (Exception e)
{
  System.out.println("nepodarilo sa");
  c = readChar(napis_pre_uzivatela);
}
return c;
}
```

Použitie vlastných dátových typov



- Vlastný (t.j. nami definovaný) dátový typ môžeme použiť ako ktorýkoľvek preddefinovaný dátový typ!
- Samozrejme môžeme definovať polia nad vlastnými dátovými typmi.

Použitie vlastných dátových typov



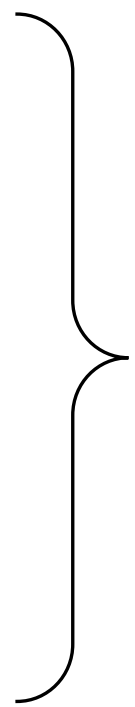
- Vlastný dátový typ môže byť použitý parameter funkcie a tiež ako návratový typ funkcie.
- Dátový typ môže byť taktiež použitý ako atribút iného dátového typu (ďalšia fólia).

Príklad



```
class Meno
{ String krstne_meno;
  String priezvisko;
}
```

```
class Adresa
{ String ulica;
  String cislo;
  String mesto;
}
```



```
class Osoba
{ Meno meno;
  Adresa adresa;
}
```

Použitie vlastných dátových typov (Pokračovanie)

- Podobne ako polia musia byť nové objekty vlastného dátového typu vytvorené pomocou kľúčového slova **new**.

Pozor: guľaté zátvorky!

Príklad: `Zaznam z = new Zaznam();`

- Podobne ako pri poliach je premenná `z` iba pointer na objekt typu `Zaznam` (t.j. `z` obsahuje adresu objektu vytvoreného v pamäti pomocou `new Zaznam()`)

Použitie vlastných dátových typov (Pokračovanie)

- Atribút `meno` nejakého objektu `obj` môže byť získaný pomocou notácie `obj.name`.
- `obj.meno` môžeme teda použiť hocikde, kde sa môže byť použitá premenná rovnakého typu ako je atribút `meno` (podobne ako pri poliach, môžeme ho tejto premennej priradiť hodnotu)
- Pri vnorenom použití dátových typov môžeme pristupovať k atribútom viacnásobným použitím „bodkovej“ notácie, napr. pre objekt `obj` typu `osoba`: môžeme použiť `obj.name.vorname` alebo `obj.adresa.mesto`

Ďalšie príklady:



```
class Bod
{ int x;
  int y;
}
```

```
class Komplexne_cislo
{ double re;
  double im;
}
```

```
class Obdlznik
{ Bod lavy_horny;
  int vyska;
  int dlzka;
}
```

```
class PseudoString
{ char[] retazec_znakov;
  int length;
}
```