# Objektno programiranje II

Konstruktor

Tabela objektov

Preobtežene metode

# Razred – ponovitev - Razred Kvadrat

- Napišimo razred Kvadrat.
  - Razred ima eno samo polje, to je stranica kvadrata
  - V razredu naj bodo tri objektne metode
    - Metoda za izračun obsega
    - Metoda za izračun ploščine
    - Metoda za izračun diagonale
  - Ustvari objekt tipa Kvadrat in ga poimenuj K1
  - Vrednost stranice (polja) za objekt K1 naj določi uporabnik
  - S pomočjo objektnih metod izračunaj in izpiši obseg, ploščino in diagonalo objekta (kvadrata) K1

# Razred: ponovitev - Razred Nepremičnina

- Sestavi razred, ki bo v svoja polja lahko shranil ulico, številko nepremičnine ter vrsto nepremičnine (blok, hiša, vila, vikend, ...).
  - Napiši metodo za nastavljanje začetnih vrednosti vseh treh polj/komponent.
  - Napiši metodo za izpis podatkov o nepremičnini. Izpis naj izgleda približno takole:

```
Na naslovu Cankarjeva ulica 32, Kranj je blok.
```

  - V metodi Main ustvari nov objekt, nato pa s pomočjo obeh metod ta objekt inicializiraj in izpiši!

# Razred: ponovitev - Razred Ulomek

- Napiši razred Ulomek, ki naj ima dve javni/public polji (stevec in imenovalec - celi števili).
  - V metodi Main nato iz tega razreda nato izpelji dva objekta(ulomka). Števca naj bosta naključni števili med 0 in 10, imenovalca pa naključni števili med 1 in 10. Ugotovi in izpiši, kateri izmed obeh ulomkov (objektov) je večji!
  - Razred Ulomek dopolni z metodo IzpisiUlomek. Metoda naj bo brez parametrov, njena naloga pa je izpis ulomka (objekta) v obliki "stevec / imenovalec". Uporabo te metode nato prikaži v metodi Main
  - Dodaj še metodo ObratniUlomek, ki zamenja števec in imenovalec tega ulomka

# Konstruktor

- Začetne vrednosti polj objekta lahko določimo s pomočjo konstruktorja
  - Konstruktor je “metoda” za nastavljanje začetnih vrednosti polj objekta
  - Konstruktor nima tipa
  - Konstruktor ima enako ime kot razred
  - Konstruktorjev je lahko več, a se morajo razlikovati  številu, oz. tipu parametrov

# Zgled – razred Pravokotnik in konstruktor

- Napišimo razred Pravokotnik in mu dodajmo še konstruktor

```
class Pravokotnik
{
    //napoved polj/komponent razreda Pravokotnik
    public double stranicaA;
    public double stranicaB;
    //Konstruktor: NIMA tipa, ime je enako kot je ime razreda
    public Pravokotnik(double a, double b)
    {
        stranicaA = a;
        stranicaB = b;
    }
}

static void Main(string[] args)
{
    //ustvarimo objekt P1 s stranicama 3.33 in 7.45
    Pravokotnik P1 = new Pravokotnik(3.33,7.45);
    Console.ReadKey();
}
```

# Vaja – razred Pravokotnik

- Razredu Pravokotnik dodaj še objektno metodo za izračun ploščine pravokotnika
- Ustvari dva objekta tipa Pravokotnik in ugotovi, kateri izmed njih ima večjo ploščino
- Ustvari tabelo 10 pravokotnikov in jim določi naključne vrednosti stranic
- Kateri izmed pravokotnikov v tabeli ima največji obseg

# Konstruktor in rezervirana beseda *this*

- Kadar se imena polj razreda ujemajo z imeni parametrov, uporabimo besedico this: ta označuje objekt, ki ga trenutno obdelujemo

```
class Kvader
{
    public double a;
    public double b; //napoved polj/komponent razreda Kvader
    public double c;
    public Kvader(double a, double b, double c) //Konstruktor
    {
        this.a = a;
        this.b = b;
        this.c = c;
    }
}

static void Main(string[] args)
{
    //ustvarimo objekt K1 z robovi 3, 5 in 9
    Kvader K1 = new Kvader(3,5,9);

    Console.ReadKey();
}
```

# Razred Kvader - nadaljevanje

- V razredu Kvader napiši objektni metodi za izračun površine in prostornine kvadra
- Ustvari dva objekta tipa Kvader in izračunaj ter izpiši njuni Prostornini. Kateri izmed kvadrov ima večjo?
- V razredu Kvader napiši objektno metodo Izpis, ki vrne niz z vsemi podatki o določenem objektu izpeljanem iz razreda Kvader (robove, površino in prostornino, telesno diagonalo)

# Vaja – razred Denarnica

- Sestavimo razred *denarnica*, ki bo omogočal naslednje operacije: dvig, vlogo (to bosta objektni metodi) , lastnika denarnice in ugotavljanje stanja (polji). Začetna vrednost naj se postavi s konstruktorjem.
  - Ustvari objekt D1: ime lastnika naj bo Maja, začetno stanje pa 250!
  - Izpiši podatke o objektu D1
  - S pomočjo objektne metode Polog v denarnico dodaj poljuben znesek: vnese ga uporabnik
  - S pomočjo objektne metode Dvig iz denarnice vzemi poljuben znesek: vnese ga uporabnik
  - Izpiši novo stanje

# Vaje - Prodajalec

- Prodajalcu napišimo program, ki mu bo pomagal pri vodenju evidence o mesečni in letni prodaji.
  - Polje razreda bo tabela zneskov, ki jo inicializiraš s pomočjo konstruktorja
  - Napiši objektno metodo SkupnaLetnaProdaja, ki izračuna in vrnse skupno letno prodajo
  - Napiši objektno metodo, ki izpiše skupno letno prodajo
  - Ustvari objekt P1: konstruktor naj prodajo po mesecih nastavi na začetno vrednost 0
  - Podatke o prodaji po mesecih naj vnese uporabnik programa
  - S pomočjo objektnih metod izpiši skupno letno prodajo

# Privzeti konstruktor

- Če konstruktorja ne napišemo ga "naredi" prevajalnik sam (a metoda ne naredi nič). Konstruktor, ki ga avtomatično generira prevajalnik, je vedno public, nima nobenega tipa (niti void), nima argumentov, vrednosti numeričnih polj postavi na 0, polja tipa *bool* postavi na *false*, vsem referenčnim poljem (spremenljivkam) pa priredi vrednost *null*.
- Če konstruktor napišemo sami, se prevajalnik ne “vmešava” več – **prevajalnik torej v tem primeru ne naredi svojega konstruktorja in ga zato tudi ne moremo uporabiti**

# Privzeti konstruktor

- Primer: razred Zajec

```
public class Zajec
{
  public string serijska;
  public bool spol;
  public double masa;
}
```

//konstruktorja nismo
//napisali, zato ga ustvari
//prevajalnik

Privzeti konstruktor, ki ga doda prevajalnik

```
public Zajec()
{
}
```

Nove objekte tipa Zajec bi torej ustvarjali takole:

```
Zajec rjavko=new Zajec();
```

# Podpisi metod

- Pri pisanju metod ("navadnih" in objektnih) se moramo držati pravila, da se morajo razlikovati ne v imenu, ampak **podpisu**. Podpis metode je sestavljen iz imena metode in tipov vseh parametrov. Tip rezultata neke metode **NI** del podpisa.

Primeri različnih podpisov metod:

- public static int **NekaMetoda**(**double x**)
- public static int **NekaMetoda**()
- public static int **NekaMetoda**(**int x**)
- public static int **NekaMetoda**(**double x,int y**)
- public static double **NekaMetoda**(**int x, int y**)
- public static int **nekaDrugaMetoda**(**double y**)

# Preobtežene metode

- So metode, ki imajo enako ime, razlikujejo pa se v številu ali pa tipu svojih parametrov (vhodnih podatkov/argumentov). **Preobtežene metode se torej razlikujejo v podpisu**!

- Namesto izraza **preobtežene** metode se pogosto uporablja tudi izraz **preobložene** metode.

- Preobtežen je lahko tudi konstruktor: v razredu zato lahko napišemo več konstruktorjev

# Preobtežene metode - zgled

- Napišimo razred *Zajec* , ki ima poleg osnovnega še tri konstruktorje – skupaj torej kar štiri!

```
public class Zajec
{
   public string serijska;
   public bool spol;
   public double masa;
   public Zajec() //Osnovni konstruktor brez parametrov
   {
        this.spol = true; // vsem zajcem na začetku določimo moški spol
        this.masa = 1.0; // in tehtajo 1kg
        this.serijska = "NEDOLOČENO"; // serijska številka je še nedoločena
   }
   public Zajec(string serijskaSt) //Prvi preobteženi konstruktor
   {       this.serijska = serijskaSt;  }
   public Zajec(bool spol) //Drugi preobteženi konstruktor
   {       this.spol = spol;    }
   public Zajec(string serijskaSt, bool spol, double masa) //Tretji preobteženi konstruktor
   {
      this.serijska = serijska;
      this.masa = masa;
      this.spol = spol;
   }
}
```

# Razred Zajec – ustvarjanje objektov

- Ustvarimo sedaj štiri zajce, vsakič pa uporabimo drug konstruktor:

```
/*ustvarimo novega zajca in mu s konstruktorjem določimo
   privzete vrednosti polj*/
Zajec prvi = new Zajec();
/*ustvarimo novega zajca in mu s konstruktorjem določimo
   serijsko številko*/
//spol je privzet (false), masa prav tako (0)
Zajec drugi = new Zajec("12001-01-1");
/*ustvarimo novega zajca in mu s konstruktorjem določimo
   spol; serijska številka je privzeta (null), masa prav
   tako (0)*/
Zajec tretji = new Zajec(true);
/*ustvarimo novega zajca in mu s konstruktorjem določimo
   serijsko, spol in maso */
Zajec cetrti = new Zajec("12001-01-2",true,1.55);
```

# Zgled – razred Complex

- Ustvarimo razred *Complex*, ki naj predstavlja kompleksno število.

```
public class Complex
{
    public float realna;
    public float imaginarna;
    public Complex() // privzeti konstruktor
    {
        realna = 0;
        imaginarna = 0;
    }
    public Complex(float real, float imag) //Preobteženi konstruktor
    {
        realna = real;
        imaginarna = imag;
    }
    public string ToString() //Objektna metoda
    {
        return (realna + " + ( " + imaginarna + " * i )");
    }
}
```

# Zgled – razred Compleks

- Ustvarimo dva objekta tipa Complex in demonstrirajmo uporabo metode ToString()

```
//ustvarjanje objekta s privzetim konstruktorjem
Complex C1 = new Complex();
Console.WriteLine(C1.ToString());

//ustvarjanje objekta s preobteženim konstruktorjem
Complex C2 = new Complex(5, 9);
Console.WriteLine(C2.ToString());
```

# Tabela objektov

- S tem, ko smo sestavili nov razred, smo dejansko sestavili nov tip podatkov. Ta je "enakovreden" vsem v C# vgrajenim tipom. Torej lahko naredimo tudi tabelo podatkov, kjer so ti podatki objekti.

```
public class Zajec   //razred Zajec
{
    public string serijska;
    public bool spol;
    public double masa;
}
public static void Main(string[] ar)
{
    Zajec[] zajci= new Zajec[250];//imamo do 250 zajcev
    for(int i=0;i<zajci.Length;i++)
    {
        zajci[i] = new Zajec();  //  "rodil" se je nov zajec
        zajci[i].serijska = "1238-12-" + i;
        zajci[i].spol = false;
        zajci[i].masa = 0.12;
    }
}
```

# Zgled: razred Firma

```
public class Zaposleni
{
  public string imeInPriimek;
  public double dohodek;
  public Zaposleni(string ime, double letniDohodek) //konstruktur
  {
      imeInPriimek = ime;
      dohodek = letniDohodek;
  }
}
static void Main(string[] args)
{
  Zaposleni[] Firma = new Zaposleni[5]; //Tabela 5 objektov tipa Zaposleni
  for (int i = 0; i < Firma.Length; i++) //Vnos podatkov
  {
      Console.Write("Ime in priimek: ");string ime = Console.ReadLine();
      Console.Write("Dohodek: ");double dohodek=double.Parse(Console.ReadLine());
      Firma[i] = new Zaposleni(ime, dohodek); //ustvarimo novega zaposlenega

  }
  Console.WriteLine("Tabela zaposlenih in njihovih letnih dohodkov: ");
  for (int i = 0; i < Firma.Length; i++) //izpis podatkov o vseh zaposelnih
  {
      Console.WriteLine(Firma[i].imeInPriimek + ": " + Firma[i].dohodek);
  }
}
```

# Vaja:

- Za vodenje evidence o padavinah v zadnjem letu potrebujemo naslednje podatke: ime kraja in podatke o količini padavin v zadnjih 12 mesecih (12 števil v polju/tabeli). Kreirajmo ustrezen razred in metodi za vnos podatkov o padavinah za vseh 12 mesecev, ter metodo, ki vrne povprečno mesečno količino padavin ustreznega kraja!

- Napišimo razred *Tocka*, ki naj ima dve zasebni polji (koordinati točke), privzeti konstruktor, ki obe koordinati postavi na 0, ter konstruktor z dvema parametroma, s katerima inicializiramo koordinati nove točke. Napišimo še metodo, ki izračuna in vrne razdaljo točke od središča koordinatnega sistema.