Objektno programiranje

Razred

Objekt

Srečo Uranič

Objektno programiranje – kaj je to?

- Dosedanje programiranje je bilo klasično:
  - Pisali smo metode, ki smo jim podatke posredovali preko parametrov
  - Med parametri in metodami ni bilo tesne povezave

- **Objekt** – podatki in postopki nad njimi združeni v celoto. Prednosti dela z objekti
  - uporabniku ni nikoli potrebno vedeti, kaj se dogaja znotraj objekta (črna škatla); objekt mora le znati ustvariti, ta pa mu potem sam sporoča, katere lastnosti in metode pozna
  - Enkapsulacija (dostopnost oz. skrivanje informacij znotraj objekta)
- Objektno programiranje – programiranje, ki omogoča delo z objekti

Objekti, ki jih že poznamo

- V naših programih smo že ves čas uporabljali različne objekte, pa to nismo vedeli
  - Objekt **System.Console** predstavlja *standardni izhod*. Kadar želimo kaj izpisati na zaslon, pokličemo metodo *Write* na objektu **System.Console**, kadar pa želimo uporabnikov vnos preko tipkovnice, pa pokličemo metodo ReadLine().
  - Objekt tipa **Random** predstavlja generator naključnih števil. Metoda *Next(a, b)*, ki jo izvedemo nad objektom tega tipa nam vrne neko naključno celo število med *a* in *b*.

Razred – osnova za tvorjenje objektov

- Objekti so torej zelo uporabni
  - Tvorimo jih iz programskih struktur, ki se imenujejo “razred”
  - Uporabljamo lahko obstoječe razrede: Console, Random, Convert, …
  - Pišemo lahko svoje razrede, iz katerih tvorimo lastne objekte

- Razred – je abstraktna definicija objekta
  - V razredu zapišemo lastnosti, stanja in obnašanje objektov
  - V razred zapišemo, katere **podatke** bomo hranili v objektih te vrste in katere **metode** oz. odzive objektov na sporočila. Stanje objekta torej opišemo s spremenljivkami (rečemo jim tudi **polja ali komponente**), njihovo obnašanje oz. odzive pa z **metodami**.

Razred - objekt

- Če želimo nek razred uporabiti, moramo iz njega običajno ustvariti vsaj en **primerek/instanco** .
- Razred je v bistvu šablona. (kanglica – potičke!!!)
  - Objekt – je primerek, ***instanca*** razreda
  - Ustvarimo ga z besedico **new**

```
imeRazreda primerek = new imeRazreda();
```

- Objekt je računalniški model predmeta ali stvari iz vsakdanjega življenja (avto, oseba, datum…). Objekt je kakršenkoli skupek podatkov, s katerimi želimo upravljati. Osnovni pristop objektnega programiranja je torej:

  objekt = podatki + metode za delo s podatki.

- Objekt je "črna škatla“, ki sprejema in pošilja sporočila. Jedro objekta sestavljajo njegove spremenljivke, okrog katerih se nahajajo njegove metode.

Klasično in objektno programiranje

- Ko želimo kak podatek obdelati v klasičnem programiranju pokličemo ustrezno metodo z argumenti, ki naj jih ta metoda obdela. Če smo npr. napisali metodo *Vsota*, ki vrne celo število, ima pa dva celoštevilska parametra, bomo to metodo poklicali npr. takole:

```
int skupaj=Vsota(23,45);
```

- Pri objektnem programiranju pa metode pripadajo objektom (ki jih izpeljemo oz. ustvarimo iz razredov). Pred imenom metode moramo napisati še ime objekta, ki mu ta metoda pripada. Recimo, da smo napisali razred *Ulomek*, znotraj tega razreda pa smo napisali metodo *Vsota*. Iz razreda *Ulomek* smo nato ustvarili objekt *U1*. Metodo *Vsota* objekta *U1* pokličemo sedaj takole:

```
int skupaj=U1.Vsota(23,45); //pred imenom metode
                            //zapišemo še ime objekta
```

Razred – osnova za tvorjenje objektov

- Primer razreda

```
public class MojR //razred z imenom MojR
{
    private string mojNiz;
    public MojR(string nekNiz)
    {
        mojNiz = nekNiz;
    }

    public void Izpisi()
    {
        Console.WriteLine(mojNiz);
    }
}
```

Definicija razreda

```
public static void Main(string[] arg)
{
    MojR prvi;          //oznaka (ime) objekta
    prvi= new MojR("Pozdravljen, moj prvi objekt v C#!");//ustvarjanje objekta
    prvi.Izpisi();   //ukaz objektu
}
```

Razred – razlaga razreda MojR

- Pred glavnim programom smo definirali nov razred z imenom *MojR*.
  - V vsakem objektu razreda MojR poznamo nek niz, ki ga hranimo v njem. Vse znanje, ki ga objekti tega razreda imajo je, da se znajo odzvati ne metodo *Izpis()* - izpišejo svojo vsebino
  - Glavni program *Main* (glavna metoda) je tisti del rešitve, ki opravi dejansko neko delo. Najprej naredimo primerek objekta iz razreda *MojR* (prvi) in vanj shranimo niz *"Pozdravljen, moj prvi objekt v C#!"*. Temu objektu nato naročimo, naj izpiše svojo vsebino.
  - Načelo združevanja - ***dostopnost*** (**enkapsulacija**). Potem, ko smo metode in polje združili znotraj razreda, smo pred temeljno odločitvijo, kaj naj bo javno, kaj pa zasebno. Vse, kar smo zapisali med zavita oklepaja v razredu, spada v notranjost razreda. Z besedicami **public, private** in **protected,** ki jih napišemo pred ime metode ali polja, lahko kontroliramo, katere metode in polja bodo dostopna tudi od zunaj:
    - Metoda ali polje je privatno (**private**), kadar je dostopno le znotraj razreda.
    - Metoda ali polje je javno (**public**), kadar je dostopno tako znotraj, kot tudi izven razreda.
    - Metoda ali polje je zaščiteno (**protected**), kadar je vidno le znotraj razreda, ali pa v **podedovanih** (**izpeljanih**) razredih.
    - Obstajajo tudi druge dostopnosti, a se z njimi ne bomo ukvarjali.

Ustvarjanje objektov

- Napoved ali deklaracija objekta

```
MojR prvi;  // NAPOVED objekta (prvi je NASLOV objekta)
Prvi=new MojR(“Pozdravljen!”); //inicializacija objekta
```

- Napoved in inicializacija objekta skupaj

```
MojR prvi = new MojR("Živjo, moj prvi objekt v C#!");
```

Sicer vedno rečemo, da v spremenljivki *prvi* hranimo objekt, a zavedati se moramo, da to ni čisto res. V spremenljivki prvi je shranjen naslov objekta.

Ustvarjanje objektov

- Naslov objekta

```
MojR prvi, drugi; /*v obeh spremenljivkah je shranjen le
    naslov objekta*/
```

Nimamo še nobenega objekta. Imamo le dve spremenljivki, v katerih lahko shranimo naslov, kjer bo operator *new* ustvaril nov objekt. Rečemo tudi, da spremenljivki *prvi* in *drugi* kažeta na objekta tipa *MojR*.

Če napišemo

```
prvi = new MojR("1.objekt");
```

V spremenljivko ***prvi*** smo shranili **naslov**, kjer je novo ustvarjeni objekt.

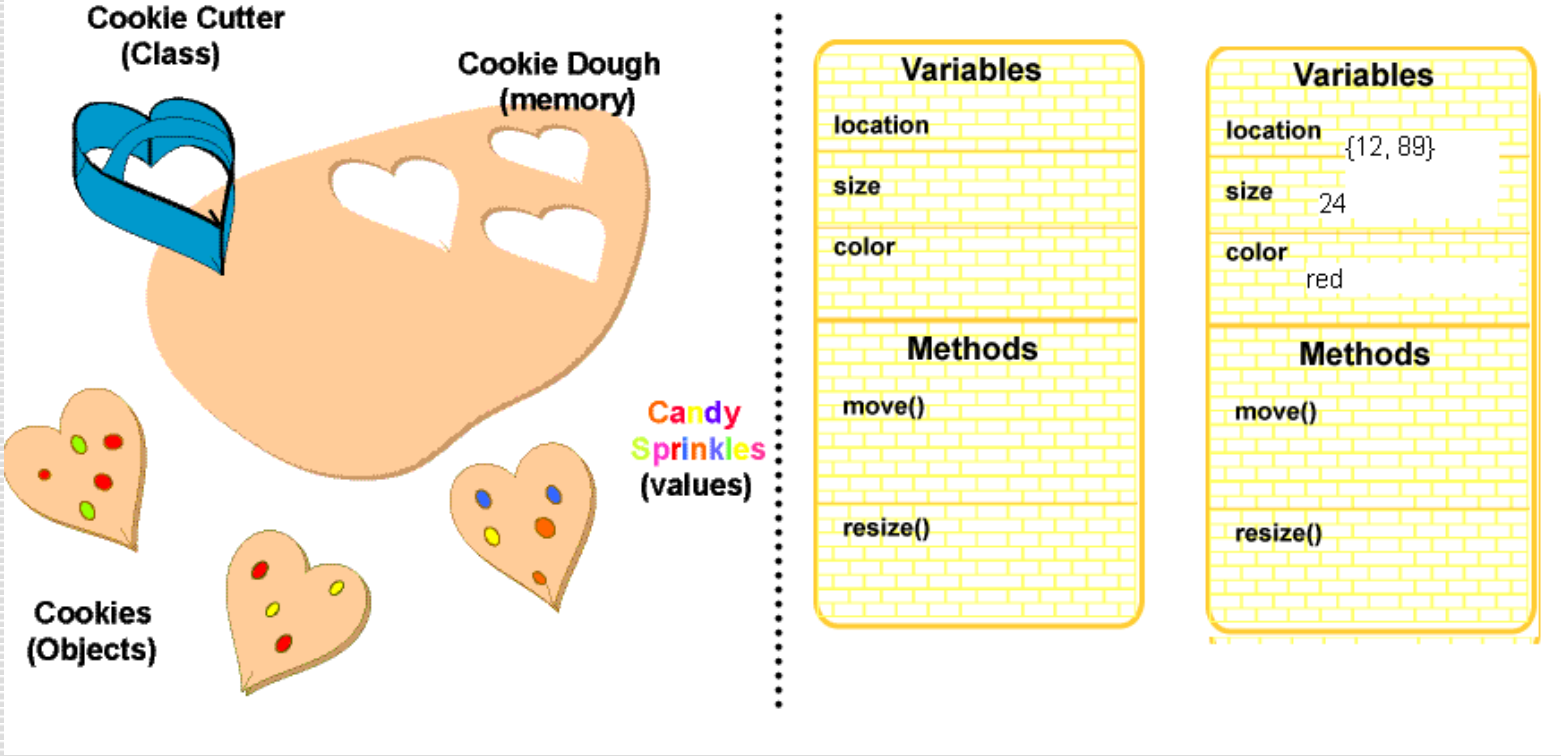
Ustvarimo še en objekt in njegov naslov shranimo v spremenljivko ***drugi***

```
drugi = new MojR("2. objekt");
```

Razred - objekti

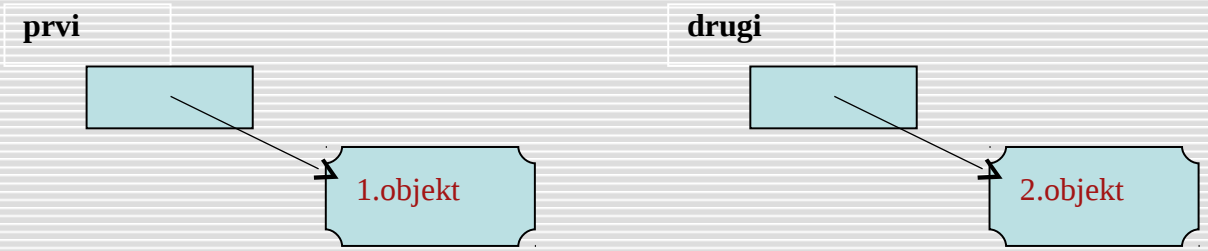
- Razred (*class*) je torej opis vrste objekta (načrt, kako naj bo objekt videti) – **šablona**, oz. opis ideje objekta. Primerek razreda (instanca) pa je konkretni objekt.

  Primer: modelček (razred), piškoti (objekti)



Stanje v pomnilniku

- V spremenljivkah *prvi* in *drugi* se nahaja ustrezen naslov objektov. Dejansko je na tistem mestu napisano nekaj v stilu *h002a22* (torej naslov mesta v pomnilniku)



In če sedaj napišemo

```
drugi = new MojR("Novo");
```

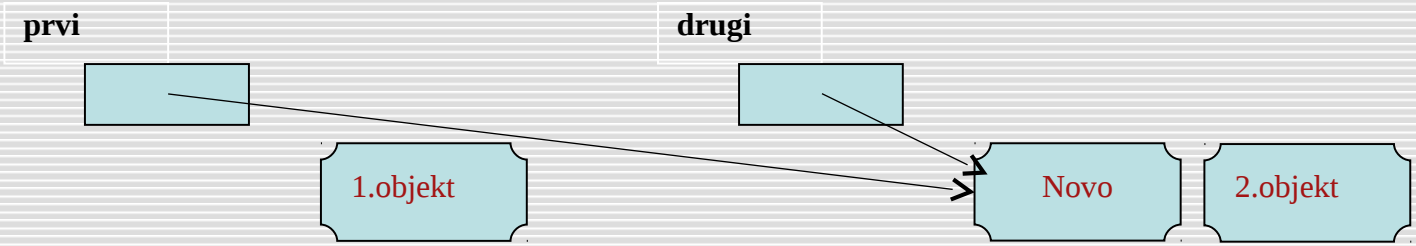
je vse v redu. Le do objekta z vsebino "2. objekt" ne moremo več! Stanje v pomnilniku je

Stanje v pomnilniku

- Če zapišemo še stavek

```
prvi = drugi;
```

smo s tem izgubili še dostop do objekta, ki smo ga prvega ustvarili in pomnilnik bo videti takole.



Sedaj tako spremenljivka *prvi* kot *drugi* vsebujeta naslov istega objekta. Do objektov z vsebino "1. Objekt" in "2. objekt" pa ne more nihče več. Na srečo bo kmalu prišel ***smetar*** in ju pometel proč. Smetar je poseben program v C#, za katerega delovanje nam ni potrebno skrbeti in poskrbi, da se po pomnilniku ne nabere preveč objektov, katerih naslova ne pozna nihče več.

Zgled – razred Krog

- Radi bi napisali program, ki bo prebral polmer kroga in izračunal ter izpisal njegovo ploščino. "**Klasično**" bi program napisali takole:

```
//  vnos podatka
Console.Write("Polmer kroga: ");
int r = int.Parse(Console.ReadLine());/*Parse pretvori
                                        string v int*/
//  izračun
double ploscina = Math.PI * r * r;
//  izpis
Console.WriteLine("Ploščina kroga: " + ploscina);
```

Zgled – razred Krog

- Radi bi napisali program, ki bo prebral polmer kroga in izračunal ter izpisal njegovo ploščino. “**Objektno**" bi program napisali takole:

```
class Krog
{
    public double polmer;    // javno polje razreda Krog
    public double Ploscina() // javna OBJEKTNA metoda razreda krog
    {
        return Math.PI * polmer * polmer;
    }
}
public static void Main(string[] arg)
{
    Krog k = new Krog(); // naredimo nov objekt tipa krog
    //POZOR: polmer kroga k je dobil privzeto vrednost 0!!!!
    Console.Write("Polmer: ");
    // do polj in metod objekta k dostopamo z operatorjem pika
    k.polmer = double.Parse(Console.ReadLine()); // polmer preberemo
    Console.WriteLine(k.Ploscina()); // izpis ploščine
}
```

Zgled : evidenca članov kluba

- Napišimo program, ki vodi evidenco o članih športnega kluba.
  - Podatki o članu obsegajo ime, priimek, letnico vpisa v klub in vpisno številke (seveda je to poenostavljen primer). Torej objekt, ki predstavlja člana kluba, vsebuje štiri podatke.

```
public class Clan //definicija razreda Clan
{
     public string ime;
     public string priimek;
     public int leto_vpisa;
     public string vpisna_st;
}
static void Main(string[] args)//glavni program
{
     Clan a = new Clan();
     a.ime = "Janez";     a.priimek = "Starina";
     a.leto_vpisa = 2000;     a.vpisna_st = "2304";
     Console.WriteLine("Član a:\n" + a.ime + " " + a.priimek +
          " " + a.leto_vpisa + " (" + a.vpisna_st + ")\n");
     Clan c = a; //objekt c ima enak naslov kot objekt a
     c.ime = "Andreja";
}
```

Zgled: Razred Zgradba

- Pogosto razrede uporabimo le zato, da v celoto združimo podatke o neki stvari. Takrat v razred "zapremo" le polja, ki imajo vsa javni dostop, metod pa v razredu ni. Napišimo razred *Zgradba* s poljema *kvadratura* in *stanovalcev*.

```
class Zgradba // deklaracija razreda Zgradba z dvema javnima poljema.
{
    public int kvadratura;
    public int stanovalcev;
}
static void Main(string[] args)
{   //ustvarimo objekt hiša
    Zgradba hiša = new Zgradba();   // nov objekt razreda Zgradba
    //POZOR: obe polji objekta “hiša” imata privzeto vrednost 0!!!
     hiša.stanovalcev = 4; //začetna vrednost za stanovalce hiše
    hiša.kvadratura = 2500;

    // izračun kvadrature za hišo
    int kvadraturaPP = hiša.kvadratura / hiša.stanovalcev;
    Console.WriteLine("Podatki o hiši:\n  " + hiša.stanovalcev + "
     stanovalcev\n  " +hiša.kvadratura + " skupna kvadratura\n  " +
     kvadraturaPP + " m2 na osebo");
}
```

Razred zgradba z objektno metodo

- Metodo za izračun kvadrature lahko zapišemo v razred in poznali jo bodo vsi objekti, izpeljani iz tega razreda

```
class Zgradba // deklaracija razreda Zgradba z dvema javnima poljema.
{
    public int kvadratura;  //polje
    public int stanovalcev; //polje
    public double Kvadratura()  //OBJEKTNA metoda
    {
       return Math.Round((double)kvadratura / stanovalcev,2);
    }
}
static void Main(string[] args)
{
    //ustvarimo objekt hiša
    Zgradba hiša = new Zgradba();   // nov objekt razreda Zgradba
    hiša.stanovalcev = 4; //začetna vrednost za stanovalce hiše
    hiša.kvadratura = 2500;
    // izračun kvadrature za hišo
    int kvadraturaPP = hiša.kvadratura / hiša.stanovalcev;
    Console.WriteLine("Kvadratura hiše: "+hiša.kvadratura + " m2 na osebo");
           Console.ReadKey();
}
```

Razred Zgradba

- Vrednosti polj posameznega objekta seveda lahko tudi preberemo, torej jih vnese uporabnik

```
Console.Write("Kvadratura: ");
hiša.kvadratura = Convert.ToInt32(Console.ReadLine());
Console.Write("Število stanovalcev: ");
hiša.stanovalcev = Convert.ToInt32(Console.ReadLine());
```