

## 8. Klase, objekti i tipovi u C# jeziku



M. Zekić-Sušac

1

## Što ćete naučiti u ovom poglavlju?

- Popis osnovnih ključnih riječi u C# jeziku
- Kako se kreiraju vlastite klase
- Kako se kreiraju instance klase
- Što je konstruktor i kako se koristi
- Kako se kreiraju metode i što su to statičke metode
- Kako se koristi ključna riječ "this"

M. Zekić-Sušac

2

## Ključne riječi C# jezika

Svaki programski jezik sastoji se od niza ključnih riječi koje imaju svoje unaprijed određeno značenje i mogu se koristiti za pisanje naredbi programa. U C# jeziku neke od ključnih riječi su:

- |           |             |
|-----------|-------------|
| ■ class   | ■ protected |
| ■ do      | ■ public    |
| ■ else    | ■ return    |
| ■ false   | ■ static    |
| ■ for     | ■ string    |
| ■ if      | ■ this      |
| ■ int     | ■ true      |
| ■ new     | ■ using     |
| ■ null    | ■ void      |
| ■ private | ■ while     |

M. Zekić-Sušac

3

## Kreiranje vlastitih klasa

- Ključne riječi koriste se u naredbama programa. Osim korištenja ugrađenih ključnih riječi, u naredbama koristimo i **proizvoljne nazive** za klase, objekte, metode i varijable.
- Do sada smo u primjerima uvijek koristili klase koje su već ugrađene u razvojni alat C# (npr. klasa Form koja je označavala formu u Windows aplikaciji). Međutim, u aplikaciji programeri često trebaju kreirati vlastite klase po potrebi.
- Kako kreirati vlastite klase? Za odgovor na to pitanje, potrebno je razumjeti osnovne koncepte objektnog programiranja – što su tipovi, klase, instance, objekti.

M. Zekić-Sušac

4

## Što je tip u objektnom programiranju?

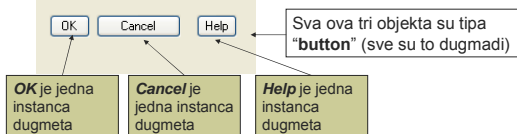
- Bit objektnog programiranja je kreiranje novih **tipova**.

**Tip** predstavlja neku stvar (eng. thing).

Ponekad je stvar abstraktna (npr. tablica), a ponekad je vrlo konkretna (npr. dugme u nekom prozoru).

Tip definira opća svojstva i ponašanje neke stvari.

Npr. dugme (kontrola **button**) je jedan tip objekta.



5

## Nasljeđivanje kod tipova

Ako smo kreirali tri dugmeta (OK, Cancel i Help), sva tri nasljeđuju svojstva koja ima tip "button":

- svojstvo visine, širine
- text koji će pisati na dugmetu
- položaj dugmeta na formi,
- i druga svojstva zajednička svim instancama koje pripadaju tipu "button"

Ta se karakteristika objektnog programiranja zove **NASLJEĐIVANJE**.

**NASLJEĐIVANJE** je mogućnost da instance preuzimaju sva svojstva tipa kojem pripadaju.

M. Zekić-Sušac

6

## Vrste tipova

U C# jeziku se koriste ova 4 tipa:

- Klase, pri čemu se instance klase zovu objekti,
- Enumeracije (eng. enums),
- Strukture (eng. structs) i
- Delegati (eng. delegates).

od kojih se najčešće koriste **KLASE**, pa će biti detaljnije objašnjene u nastavku.

M. Zekić-Sušac

7

## Klase, objekti (instance)

- **KLASA** je neki tip promatranih objekata.
- Klasa je adekvatna skupini ili vrsti kod bioloških bića, npr. skupina pasa, mačaka, studenata, radnika, artikala, itd. Objekt je **instanca** neke klase, npr.

1. Pas = klasa (ili skupina objekata)  
Rex = instanca klase Pas (konkretn pas iz klase Pas)
2. Student = klasa  
Pero Perić = instanca klase Student
3. Button = klasa  
OK button koji napravimo u aplikaciji = instanca klase button



## Kreiranje klase

Kada kreiramo neku vlastitu klasu, potrebno je definirati:

- njezina **svojstva (properties)** ili **variable** kojima će dodjeljivati vrijednosti (ako se radi o javnim svojstvima, tada se koristi izraz "svojstvo" (eng. property), a ako se radi o privatnim varijablama koje će se koristiti samo u toj klasi, tada govorimo koristimo izraz "varijabla")
- **metode** koje određuju njezino ponašanje.

Naredba za kreiranje nove klase počinje ključnom riječi **class**, zatim dolazi *naziv klase*, zatim deklaracija svojstava (properties) koje se definiraju za tu klasu, te metode koje pripadaju klasi.

M. Zekić-Sušac

9

## Kreiranje klase - nastavak

- Npr. ovako bismo kreirali vlastitu klasu:

```
class mojaKlasa
{
    //ovdje definicija svojstava ili varijabli,
    npr.
    int mojaVarijabla;
}
```

Gornjim naredbama kreirali smo klasu pod nazivom **mojaKlasa**, i njoj pripadajuću cjelobrojnu varijablu pod nazivom **mojaVarijabla** kojoj će se unutar klase mojaKlasa moći dodjeljivati vrijednosti.

M. Zekić-Sušac

10

## Kreiranje instance neke klase

- Npr. sljedećim naredbama definira se klasa ili skupina objekata **Student**, a unutar nje metoda pod nazivom **Main()** (Main() je glavna metoda za neki program, a nalazi se u datoteci Program.cs u okviru Solution Explorer-a). U metodi se definira instanca klase.

```
class Student
{
    // ovdje definiramo svojstva koja pripadaju toj klasi:
    int Ocjena;
    // ovdje definiramo metodu:
    public void Main()
    {
        Student Pero = new Student();
        Pero.Ocjena = 3;
    }
}
```

Kreiramo instancu Pero koja pripada klasi Student

Varijabli Ocjena koja pripada instanci Pero dodjeljuje se vrijednost 3

M. Zekić-Sušac

11

## Konstruktor

Naredba kojom se definira instanca neke klase izgleda ovako: (npr. za kreiranje instance Pero klase Student)

```
Student Pero = new Student();
```

Naredba sadrži na kraju zagrada (), pa izgled kao da poziva metodu. Zapravo, pri kreiranju nove instance poziva se jedna metoda koja se zove **konstruktor** (eng. constructor).

Zadatak konstruktora je kreirati instancu (objekt) neke klase i omogućiti da se toj instanci dodijeli vrijednost kasnije u programu.

Nakon što se konstruktorom kreira nova instanca, u memoriji se ostavlja prostor za vrijednosnu instancu. Ako ne specificiramo koji konstruktor pozivamo, CLR poziva standardni (default) konstruktor kojim se definira nova instanca klase i ne proslijeđuju mu se nikakve vrijednosti. Zato u zagradama kod ovog standardnog konstruktora ne pišemo ništa.

M. Zekić-Sušac

12

## Konstruktor - nastavak

Ukoliko se želi koristiti neki vlastiti određeni konstruktor, tada se on mora deklarirati u programu prije samog korištenja.

Primjer: deklariranje konstruktora koji će ispisivati sistemsko vrijeme

```
// constructor
public Time(System.DateTime dt)
{
    Year = dt.Year;
    Month = dt.Month;
    Date = dt.Day;
    Hour = dt.Hour;
    Minute = dt.Minute;
    Second = dt.Second;
}
```

Deklaracija konstruktora koji je javan (public) da bi se mogao koristiti u svim metodama, on će konstruirati instance klase Time, a proslijeđuje mu se varijabla dt koja je tipa System.DateTime

Kako bi ovaj primjer radio, potrebno je prije korištenja konstruktora i deklarirati i dodijeliti vrijednosti za varijable Year, Month, Date, Hour, Minute, Second.

M. Zekić-Sušac

13

## Konstruktor - nastavak

Nakon deklaracije, u nekoj metodi, npr. Main() može se koristiti deklarirani konstruktor da bi kreirao novu instancu klase Time. Deklarirani konstruktor koristi se u klasi Tester, u metodi Main():

```
public class Tester
{
    static void Main()
    {
        System.DateTime currentTime = System.DateTime.Now;
        Time t = new Time(currentTime);
        t.DisplayCurrentTime();
    }
}
```

Više o kreiranju vlastitih konstruktora vidi u Liberty, 2005.

M. Zekić-Sušac

14

## Metode

**Metoda** je neka funkcija koja se nalazi unutar klase. Ona je član (eng. member) klase.

- Metode definiraju što klase rade i kako se klase ponašaju.
- Metodama koje sami kreiramo možemo dati proizvoljne naziv, npr. Ispis(), Racunanje(), itd.
- Ako želimo napisati odgovor na događaj klik miša na neku kontrolu (button ili dr.), dvostrukim klikom miša program će kreirati metodu i dati joj standardni naziv, npr. button1\_Click()

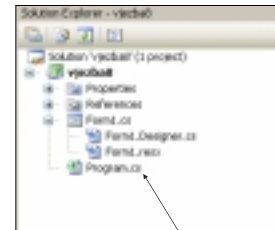
U svakoj aplikaciji postoji prva, glavna metoda iz koje se pozivaju sve ostale metode te aplikacije. Ta se metoda zove **Main()**. Metoda Main() može se u C#-u vidjeti u prozoru **Program.cs** u Solution Explorer-u)

M. Zekić-Sušac

15

## Metoda Main()

- Otvorite u C# Express-u novi projekt (Windows aplikaciju) pod nazivom Vjezba8.
- U Solution Explorer prozoru kliknite na Program.cs
- Pojavit će se programski kod za glavni program aplikacije koji poziva sve metode te aplikacije



Glavni program aplikacije može se vidjeti kada kliknemo dvaput na "Program.cs" u prozoru Solution Explorer-a

M. Zekić-Sušac

16

## Metoda Main() - nastavak

Glavni program aplikacije izgleda ovako:

```
namespace vjezba8
```

```
{
    static class Program
    {
        /// <summary>
        /// ovdje dolazi opis što aplikacija radi
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

M. Zekić-Sušac

17

## Metoda Main() – nastavak

- Metoda Main() se automatski poziva od strane CLR-a (Common Language Runtime-a) prilikom izvršavanja programa.
- Za nas važna naredba u metodi Main() je ona koja pokreće aplikaciju, i otvara prvu formu aplikacije. To je naredba:  
**Application.Run(new Form1());**
- Gornjom naredbom pokreće se aplikacija otvaranjem prve forme Form1.

M. Zekić-Sušac

18

## Kreiranje vlastite metode

- Kada se kreira neka metoda, ona također ima svoju deklaraciju. U deklaraciji se definira tip podatka koji će metoda vraćati, zatim naziv metode i u zagradi parametre koji će se proslijeđivati metodi. Npr. ovako:

```
int myMethod(int velicina);
```

Ovdje se određuje da će metoda nakon izvršavanja vratiti cjelobrojnu vrijednost

Naziv metode je myMethod (može biti proizvoljan)

Ovdje se određuje da će se varijabla s kojom će se u metodi računati zvati "velicina" i bit će cjelobrojna

- Neke metode uopće ne vraćaju nikakve vrijednosti. Takve metode samo nešto izvode, i tada je tip metode **void**, a u zagradi se ne piše ništa, npr. ovako:

```
void myVoidMethod( );
```

M. Zekić-Sušac

19

## Metode tipa **void**

- U C# se prilikom korištenja metode uvijek mora deklarirati tip koji metoda vraća ili napisati ključnu riječ "void" ako metoda ne vraća nikakvu vrijednost.
- Npr. do sada smo u primjerima koristili metode koje ne vraćaju vrijednosti, ali su bile standardne metode za klik miša na neko dugme, pa su njihove deklaracije izgledale ovako:

```
private void button1_Click(object sender, EventArgs e)
```

"private" znači da se metoda koristi samo unutar jedne klase u kojoj je deklarirana

"void" znači da metoda ne vraća nikakvu vrijednost

"button1\_Click" je naziv metode

Metodi se u zagradi proslijeđuju parametri koji su standardni za neku "click" metodu

M. Zekić-Sušac

20

## Komentari u programskom kodu

Komentari se u programski kod umeću zbog više razloga:

- Kako bi olakšali programeru snalaženje u kodu (zbog mogućih kasnijih ispravki)
- Kako bi omogućili izradu adekvatne programske dokumentacije

Postoji tri osnovna načina upotrebe komentara:

1. **C++ stil:**  
*// ovo je C++ stil pisanja komentara i vrijedi do kraja  
// jedne linije*
2. **C stil:**  
*/\* ovo je C stil pisanja komentara.  
Komentar se može pisati kroz više linija  
a vrijedi do znaka \*/*
3. **Komentari za XML dokumente**

M. Zekić-Sušac

21

## Operator točka (.)

Operator . koristi se:

- Za pristup metodi ili svojstvu unutar neke klase
- Kako bi ograničio naziv klase na određeni prostor za nazive (namespace), npr. u naredbi **using System.Windows.Forms;** koristi se prostor za nazive System, u njemu podprostor Windows, a u njemu podprostor Forms

Primjer:

```
Student.Pero.Visina=175;
```

koristi instancu Pero klase Student, i njegovom svojstvu Visina dodjeljuje vrijednost 175.

M. Zekić-Sušac

22

## Korištenje statičnih članova

- Pri korištenju metode Main() vidjeli smo da naredba sadrži ključnu riječ "static":

```
static void Main()
```

Riječ "static" znači da se ta metoda može pozvati samostalno bez prethodnog pozivanja klase ("Možete je koristiti bez da u ruci imate instancu (te klase" – Liberty, 2005, str. 67)

Svojstva i metode neke klase mogu biti instance ili statični članovi. Ako su instance, moraju se pozivati preko operatora ".", tako da se najprije navede klasa kojoj pripadaju, npr. ako definiramo klasu Student, i u okviru nje instancu Pero, a zatim neku metodu Upisi() (kojom će se studenti upisivati na fakultet), tada se ta metoda poziva:

- a) ako prethodno nije definirana kao statična:  

```
Student.Pero.Upisi();
```
- b) ako je prethodno definirana kao statična:  

```
Student.Upisi();
```

M. Zekić-Sušac

23

## Ključna riječ **this**

Ključna riječ "**this**" odnosi se na trenutnu instancu neke klase (ili drugog objekta).

- "**this**" je skriveni pokazivač (pointer) na svaku nestatičnu metodu neke klase

Postoje 3 načina upotrebe **this**:

1. način: U svrhu kvalificiranja instance ako se ona zove isto kao i parametar koji se proslijeđuje nekoj metodi, pa tada "**this**" omogućava da se odredi na koju se vrijednost misli

Npr.

```
public void SomeMethod (int hour)  
{  
    this.hour = hour;  
}
```

Ovoj metodi se proslijeđuje cjelobrojni parametar **hour**. Međutim, u metodi se koristi i instanca (ili svojstvo) **hour** kojoj treba dodijeliti vrijednost proslijeđenog parametra. **This** određuje da se na lijevoj strani jednakosti misli na instancu **hour** koja pripada aktualnoj klasi.

(Liberty, 2005)

M. Zekić-Sušac

## Ključna riječ **this** - nastavak

2. način: U svrhu proslijeđivanja trenutno aktivnog objekta kao parametra drugoj metodi

```
public void FirstMethod(OtherClass otherObject)
{
    otherObject.SecondMethod(this);
}
```

Ovaj primjer uspostavlja dvije klase; jedna je klasa koja ima metodu FirstMethod(), a druga je klasa OtherClass koja ima metodu SecondMethod().

Ako unutar prve metode želimo pozvati metodu koja pripada drugoj klasi, tada tu drugu metodu moramo pozvati iz te druge klase. Tu klasu ne možemo koristiti pod njezinim nazivom, nego moramo proslijediti parametar – varijablu koja je tipa OtherClass.

3. način: korištenjem indeksera (detaljnije pogledaj u Liberty,2005)

M. Zekić-Sušac

25

## Primjer korištenja **this**

■ Primjer ako **this** koristimo u svrhu određivanja čije svojstvo (property) će se koristiti:

```
label4.Text = this.ImePrezimeStudenta;
```

Gornjom naredbom u tekstu labele **label4** će se ispisati vrijednost svojstva **ImePrezimeStudenta** koje pripada aktualnoj klasi (u kojoj se nalazi ta metoda i naredba)

```
textBox1.Text = this.Ocjena;
```

Gornjom naredbom u tekstu **textBox1** će se ispisati vrijednost svojstva **Ocjena** koje pripada aktualnoj klasi (u kojoj se nalazi ta metoda i naredba).

M. Zekić-Sušac

26

## Pitanja za ponavljanje

1. Što je tip u objektnom programiranju?
2. Što definira tip nekog objekta?
3. Ako koristimo izraze: **button**, **OK**, **Cancel**, **Help**, što je od toga tip, a što su instance objekta?
4. Ako koristimo izraze: **student**, **Ivo Ivić**, što je od toga tip, a što su instance objekta?
5. Što je nasljeđivanje u objektnom programiranju?
6. Objasnite pojam nasljeđivanja na primjeru tipa **student** i neke njegove instance kojoj dajte proizvoljno ime.
7. Što se nasljeđuje u objektnom programiranju (metode ili svojstva ili objekti ili dr.?)
8. Navedite 4 glavna tipa koja se koriste kod objektnog programiranja.
9. Što je klasa u objektnom programiranju?

M. Zekić-Sušac

27

## Pitanja za ponavljanje

10. Što je instanca u objektnom programiranju?
11. Što možemo definirati za neku klasu?
12. Za što se koriste metode, tj. što se pomoću njih određuje ili opisuje?
13. Koji je opći oblik naredbe za kreiranje neke nove klase?
14. Definirajte novu klasu pod nazivom **Student** koja će imati jednu cjelobrojnu varijablu **OcjenaStudenta** i jednu tekstualnu varijablu **ImePrezimeStudenta**.
15. Kako se zove naredba kojom se kreira nova instanca neke klase?
16. Što je zadatak konstruktora?
17. Napišite konstruktor naredbu kojom se kreira instanca **Rex** koja pripada klasi **Pas**.
18. Što se kreira naredbom: **Student Student1 = new Student;**

M. Zekić-Sušac

28

## Pitanja za ponavljanje

19. Što je tip u objektnom programiranju?
20. Kakva je to metoda **Main()** i u kojoj datoteci aplikacije se nalazi?
21. Što je metoda u objektnom programiranju?
22. Koji je glavni program iz kojeg se pokreće aplikacija i poziva prva forma aplikacije: **Program.cs**, **Form1.cs**, ili **References**?
23. Kada se pokreće metoda **Main()**?
24. Koja je naredba metode **Main()** koja pokreće aplikaciju i poziva prvu formu?
25. Što se definira u deklaraciji neke metode?
26. Što znači ključna riječ **void** u deklaraciji metode?
27. Napišite deklaraciju metode koja se zove **Izracunaj()**, koja je privatna za klasu u kojoj se nalazi, koja ne vraća nikakvu vrijednost i ne proslijeđuju joj se parametri.
28. Koja su tri načina pisanja komentara u C# jeziku?

M. Zekić-Sušac

29

## Pitanja za ponavljanje

29. Da li je ispravno napisan komentar:  
`// Ovdje počinje deklaracija novog svojstva za artikl1 //`
30. Napišite ispravno komentar "Ovdje počinje deklaracija svojstva za artikl1" kroz dvije linije koda u C++ stilu.
31. Napišite ispravno komentar "Ovdje počinje deklaracija svojstva za artikl1" kroz dvije linije koda u C stilu.
32. Za što se koristi operator **..**?
33. Koristeći operator **..** dodijelite svojstvu **Ime** instance **Student1** klase **Student** vrijednost "Ivo".
34. Koristeći operator **..** dodijelite svojstvu **Placa** instance **Djelatnik1** klase **Djelatnik** vrijednost "2500.20".
35. Što znači ključna riječ **static** za neku metodu?
36. Ako u naredbi napišemo ključnu riječ **this**, na što se ona odnosi?
37. Što će napraviti naredba: **label4.Text = this.ImePrezimeStudenta;**
38. Napišite naredbu kojom ćete u okviru za tekst **textBox1** ispisati vrijednost svojstva **Placa** koje pripada instanci klase u kojoj se nalazite (koristite **this**).

M. Zekić-Sušac

30

## [ Literatura ]

- J. Liberty, Programming C#, Fourth edition, O'Reilly and Associates, 2005.
- Prijevod: C# programiranje, A. Dragosavljević (ur.), Dobar plan, 2005.
- Predavanja i zadaci na web stranici kolegija
- Web izvori