Vežbe 1

OISISI 2023/2024

Sadržaj

O predmetu

Cilj vežbi je izrada timskog (dva člana) projekta. Vežbe će biti fokusirane na projekat, gde se svaki termin nadograđuje na sledeći.

Projekat je informacioni sistem studentske službe, izrađen kao desktop aplikacija u WPF (.NET) tehnologiji

Postoje dva domaća zadatka, odnosno kontrolne tačke koje predstavljaju delove projekta i nose po 5 bodova

Odbrana projekta nosi 30 bodova.

O predmetu - glavne veštine

Timski rad - socijalni i organizacioni aspekti timskog rada, upotreba git-a

Tehnologije - C# i WPF, posebno korisno za dalje predmete kao što su SIMS, HCI i PSW

Objektno orijentisano programiranje - napredne veštine i obrasci OOP, primenjene na projektu

Rad sa modelom podataka i skladištenjem

Izrada korisničkih aplikacija - interakcija sa korisnikom, desktop aplikacije

Uvod

C# je programski jezik kreiran od strane kompanije Microsoft

C# se izvršava u okviru .NET Framework-a

Okruženje (IDE - Integrated Development Environment) koje ćemo koristiti zove se *Visual Studio*

Zvanična web stranica jezika

Konvertovanje tipova podataka (Kastovanje)

```
int mojBroj = 10;

double mojDouble = mojBroj;                    // Implicitno iz int u double


double mojDouble = 5.2;

int mojBroj = (int) mojDouble;                 // Eksplicitno iz double u int


int mojBroj = 10;

Console.WriteLine(Convert.ToString(mojBroj));    // int u string
```

casting dokumentacija

Konstante, enumeracija

Konstante se definišu pomoću rezervisane reči const

```
const int broj = 123;
```

const dokumentacija

Enumeracija se definišu pomoću rezervisane reči enum

```
enum Dani {Pon, Uto, Sre, Cet, Pet, Sub, Ned};
```

enum dokumentacija

Namespace

Sav kod u .NET arhitekturi se realizuje u okviru imenskog prostora

```
namespace mojProstor {

    // kod...
}
```

namespace dokumentacija >> proći kroz projekat Uvod <<

Klase

```
class Automobil {

  string boja;                        // polje 1

  int prosecnaBrzina = 100;           // polje 2

  public void ispisiBoju() {          // metoda

     Console.WriteLine("Boja automobila je: " + boja);

  }

}
```

klase dokumentacija

Kreiranje instance klase

```
Automobil mojAuto1 = new Automobil();
Automobil mojAuto2 = new Automobil();
```

Konstruktor

```
class Automobil
{
  public string boja;
  public Automobil()  // Konstruktor
  {
    boja = "crna";   // Konstruktor postavlja inicijalnu vrednost boje
  }
}
```

Modifikatori pristupa

public - dostupan svim klasama

private - dostupan samo u okviru klase

protected - dostupan u okviru iste klase ili klase naslednice (kasnije učimo više o nasleđivanju)

internal - dostupan u okviru istog assembly

dokumentacija o modifikatorima pristupa

Class Properties

Definiše se na sledeći način:

```
class Osoba {
  private string ime;  // polje
  public string Ime    // property (svojstvo)
  {
    get { return ime; }    // get metoda
    set { ime = value; }  // set metoda
  }
}
```

property dokumentacija

Nasledjivanje

Sve klase u C# su izvedene klase pošto nasleđuju **System.Object** klasu.

C# dozvoljava nasleđivanje klasa i implementiranje interfejsa.

Dozvoljeno je nasleđivanje samo jedne klase, dok je moguće implementirati više interfejsa.

Klasa koja se nasleđuje je roditeljska klasa (parent), a klasa koja je nasleđuje je child klasa.

Prilikom redefinicije funkcije mora se upotrebiti ključna reč **override**.

nasleđivanje dokumentacija

```
class Vozilo    // roditeljska klasa
{
   string boja = "crna";
   int prosecnaBrzina = 100;
   string brend = "Mercedes";

   public void ispisiBoju()
   {
      Console.WriteLine("Boja automobila je: " + boja);
   }
}
```

```
class Automobil : Vozilo  // klasa naslednica (child)
{
  public string klasa = "B";
}
class Program
{
  static void Main(string[] args)
  {
    Automobil auto = new Automobil();
    auto.ispisiBoju();
    Console.WriteLine(auto.brend);}
 }
```

Interfejs

Interface predstavlja apstrakciju klase.
Interface definiše skup definicija (naziv, povratni tip, argumente) metoda.

Klase koje koriste neki interfejs kažemo da implementiraju taj interfejs.

Kada klasa implementira neki interface ona bi trebala da implementira sve metode definisane tim intefejsom.

Interfejsi se koriste u statički tipiziranim jezicima (kao što su Java, C#, i drugi) da omoguće polimorfizam među klasama koji ne pripadaju istoj hijerarhiji.

Interfejs se deklarišu uz pomoć ključne reči **interface**. Sve metode definisane u interfejsu su javne (public)

interface dokumentacija

```
interface IAuto {
  void ispisiModel();
  void ispisiBrend();
}

class Automobil : IAuto {
  public void ispisiModel() {
    Console.WriteLine("Model je B");
  }

  public void ispisiBrend() {
    Console.WriteLine("Model je Mercedes");}
}
```

Organizacija koda u .NET framework-u

- .NET framework organizuje kod u hijerarhiji koja je karakteristična njemu
- Na vrhu hijerarhije se nalazi **solution**
  - **Solution** prosto predstavlja skup .**NET** projekata koji logički pripadaju jednoj celini
  - Kroz **IDE** moguće je raditi na svim projektima iz jednog solutiona paralelno
  - Solution takođe pruža mogućnost jednostavnog međusobnog uvezivanja projekata
    - Ovo može biti korisno ako želimo da iskoristimo neke klase ili metode koje su definisane u jednom projektu u drugom
- Ispod solutiona je **projekat**
  - **Projekat** je skup fajlova (izvorni kod, resursni fajlovi) koji zajedno nakon **build** procesa proizvode najčešće izvršiv fajl (**executable**) ili .NET biblioteku (**library**)
    - C# je kompajliran jezik, pa samim tim svi projekti moraju prolaziti kroz **build** proces pre izvršavanja, nezavisno od toga dal je krajnji rezultat biblioteka ili izvršiv fajl

dokumentacija o projektima i solution-ima

Zadaci

Za pomoć pri izradi možeš se poslužiti ovim dokumentom

- Napraviti konzolnu aplikaciju koja učitava prirodan broj *n* i kao rezultat ispiše *n*-ti član Fibonačijevog niza.
- Napraviti konzolnu aplikaciju koju učitava string i provjerava da li je string palindrom.
- Napraviti konzolnu aplikaciju. Dodati klasu automobil koja ima atribute: model, marka, boja, godina proizvodnje, vrsta goriva i motor. Vrsta goriva je enumeracija koja može da ima vrijednosti dizel ili benzin. Atribut motor je tipa motor. Motor je klasa sa poljima naziv, kubikaža i snaga. Napravi po par objekata klasa motor i automobil. Za obje klase implementirati *ToString* metodu i ispisati objekte na konzoli. Implementirati *Equals* metodu i provjeriti jednakost kreiranih objekata.