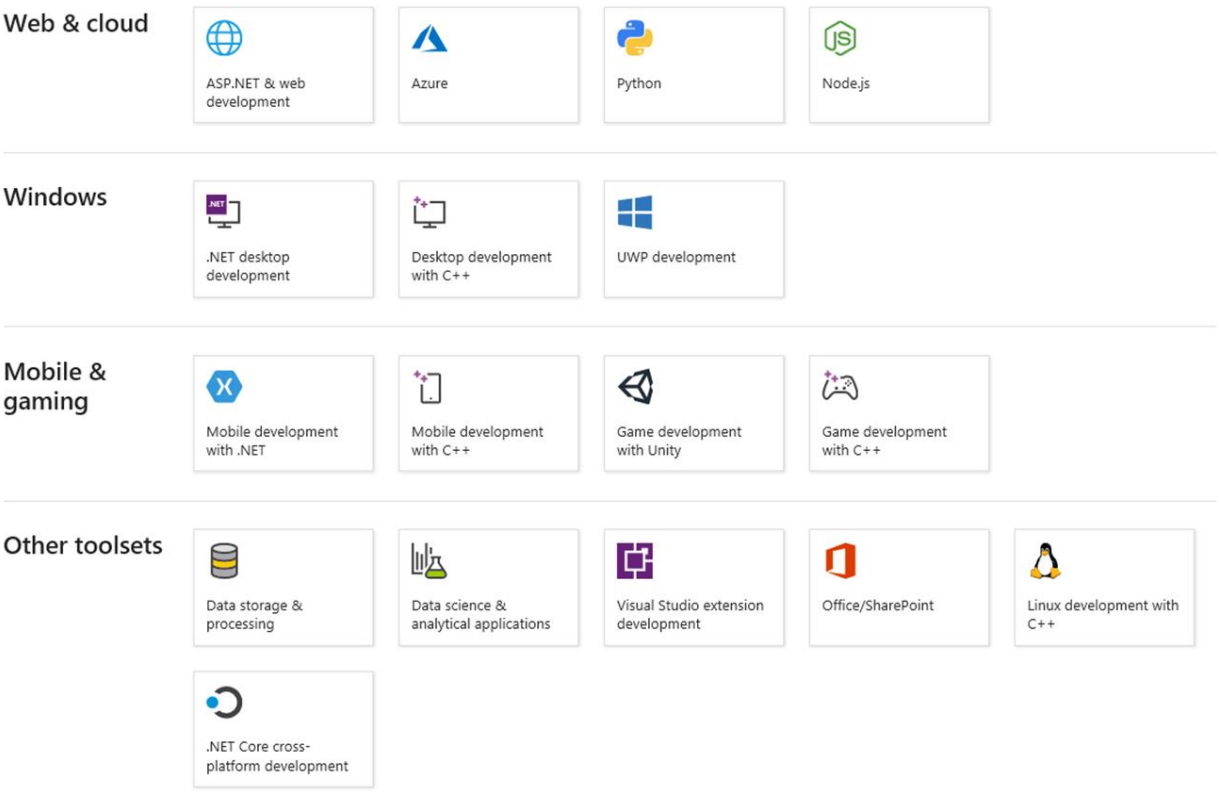
Prof. dr Zoran S. Veličković, dipl. inž. el.

.NET TEHNOLOGIJE

praktikum laboratorijskih vežbi



Niš, novembar 2021.

LABORATORIJSKA VEŽBA 7: GENERICI I INTERFEJSI

Cilj vežbe: Upoznavanje sa osnovnim pojmovima vezanim za generike i interfejse u .NET-u. Motivacija za kreiranje generičkih klasa i metoda. Razumevanje pojma interfejsa i primena u .NET-u.

Zadatak 1: Kreiranje preklopljenih metoda

Kreirati projekt tipa „Console App (.NET Core)“ pod imenom Vezba_7_Z1. U okviru projekta kreirati klasu `Prikazi` koja sadrži preklopljene metode `DisplayArray()`. Prva preklopljena metoda prikazuje sadržaj celobrojnog niza, druga niza tipa double, a treća niz karaktera. Za pristupanje članovima nizova koristiti `foreach` naredbu. U glavnom programu kreirati tri odgovarajuća niza i primeniti preklopljene metode. Na slikama od 7-1 do 7-3 prikazan je sadržaj ovog projekta.

Na slici 7-1 prikazan je izvorni kod glavnog programa.

```
using System;
using System.Collections.Generic;

namespace Vezba_7_Z1
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] intArray = {1, 2, 3, 4, 5, 6};
            double[] doubleArray = { 1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7};
            char[] charArray = {'Z', 'D', 'R', 'A', 'V', 'O'};

            Console.WriteLine("Niz intArray sadrži:");
            Prikazi.DisplayArray(intArray);

            Console.WriteLine("Niz doubleArray sadrži:");
            Prikazi.DisplayArray(doubleArray);

            Console.WriteLine("Niz charArray sadrži:");
            Prikazi.DisplayArray(charArray);
        }
    }
}
```

Slika 7-1. Programski kod glavnog programa za Zadatak 1.

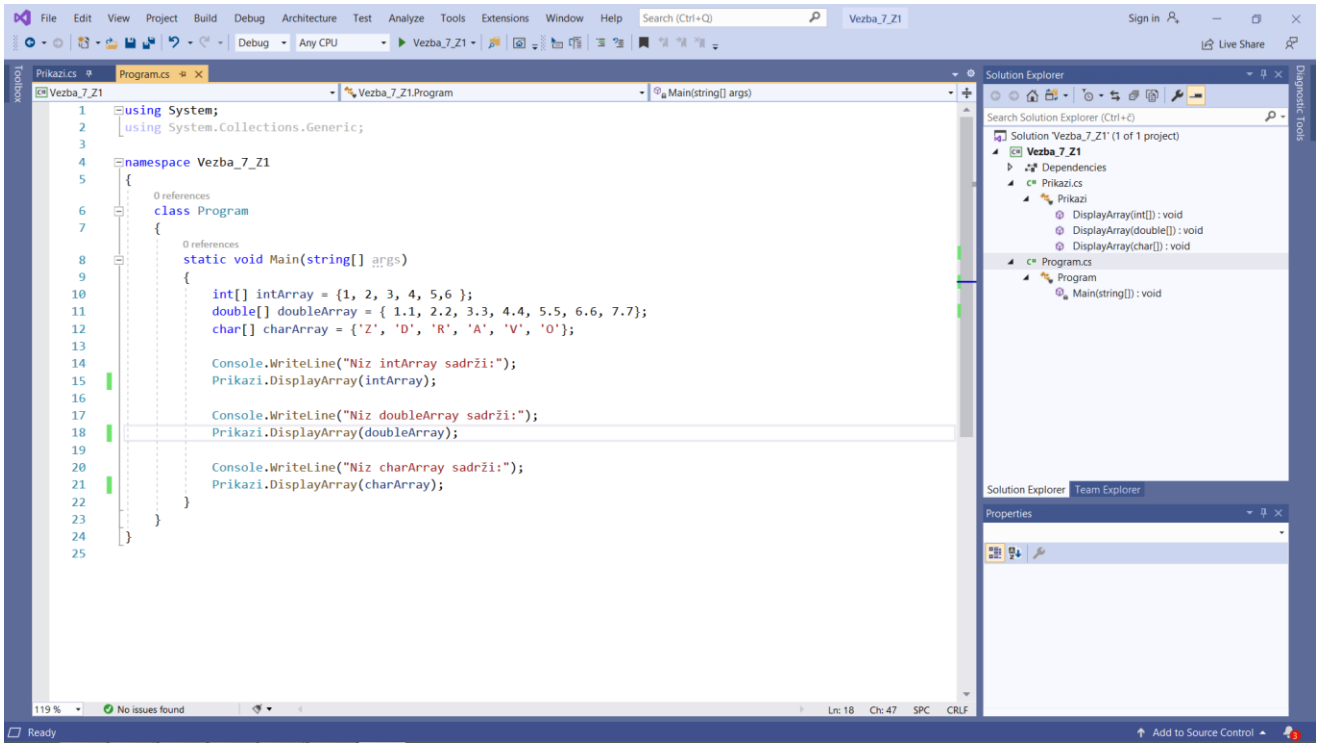
Na slici 7-2 prikazan je izvorni kod klase `Prikazi` koja sadrži sve tri preklopljene metode `DisplayArray()`.

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Vezba_7_Z1
{
    class Prikazi
```

```
    {
        public static void DisplayArray(int[] inputArray)
        {
            foreach (int element in inputArray)
                Console.Write(element + " ");
            Console.WriteLine("\n");
        }

        public static void DisplayArray(double[] inputArray)
        {
            foreach (double element in inputArray)
                Console.Write(element + " ");
            Console.WriteLine("\n");
        }

        public static void DisplayArray(char[] inputArray)
        {
            foreach (char element in inputArray)
                Console.Write(element + " ");
            Console.WriteLine("\n");
        }
    }
}
```

Slika 7-2. Programski kod klase Prikazi za Zadatak 1.

Na slici 7-3 prikazan je izgled projekta za Zadatak 1 realizovan u VS-u 2019.



Slika 7-3. Projekat za Zadatak 1 realizovan u VS-u.

Objasnite programske iskaze iz klase `Program`:

```
int[] intArray = {1, 2, 3, 4, 5,6 };
double[] doubleArray = { 1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7};
char[] charArray = {'Z', 'D', 'R', 'A', 'V', 'O'};
```

```
Prikazi.DisplayArray(intArray);
Prikazi.DisplayArray(doubleArray);
Prikazi.DisplayArray(charArray);
```

Objasnite programske iskaze iz klase `Prikazi`:

```
  public static void DisplayArray(double[] inputArray)
  public static void DisplayArray(int[] inputArray)
  public static void DisplayArray(char[] inputArray)
```

```
   foreach (char element in inputArray)
     Console.Write(element + " ");
```

```
Console.Write(element + " ");
```

Objasnite pojam preklopljenih metoda za prikazani slučaj.

Ispisati sadržaj ekrana konzolnog ekrana u predviđenom prostoru.

Zadatak 2: Kreiranje generičke metode `DisplayArray<T>`

Kreirati generičku metodu `DisplayArray<T>` koja može da zameni sva tri tipa preklopljenih metoda iz Zadataka 1. Generičku metodu realizovati u klasi `Prikazi`. Sadržaj projekta je prikazan na slikama od 7-4 do 7-6.

Na slici 7-4 prikazan je izvorni kod klase `Program` .

```
using System;
using System.Collections.Generic;

namespace Vezba_7_Z2
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] intArray = {1, 2, 3, 4, 5, 6};
            double[] doubleArray = {1.1, 2.2, 3.3, 4.4, 5.5, 6.6};
            char[] charArray = { 'Z', 'D', 'R', 'A', 'V', 'O'};

            Console.WriteLine("Niz intArray sadrži: ");
            Prikazi.DisplayArray<int>(intArray);

            Console.WriteLine("Niz doubleArray sadrži: ");
            Prikazi.DisplayArray<double>(doubleArray);

            Console.WriteLine("Niz charArray sadrži: ");
            Prikazi.DisplayArray<char>(charArray);
        }
    }
}
```

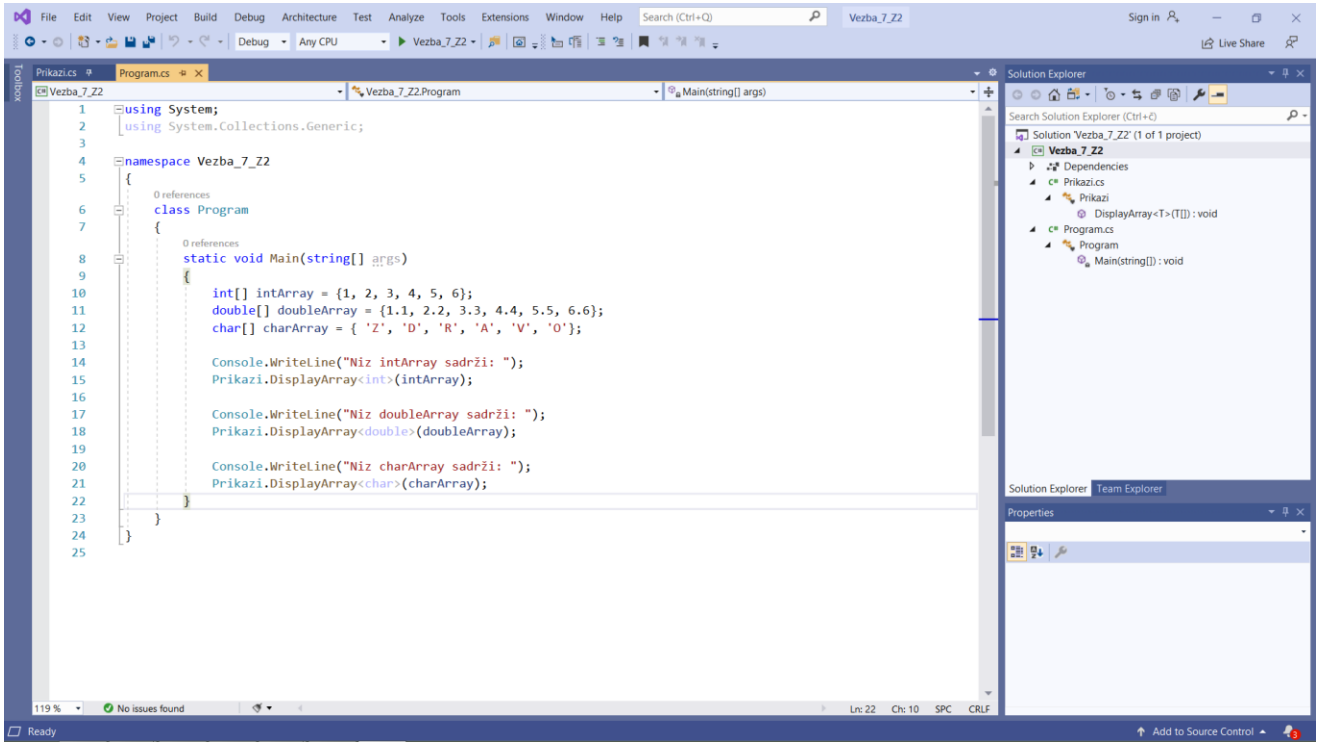
Slika 7-4. Programski kod klase `Program`.

Na slici 7-5 prikazan je izvorni kod klase `Prikazi`.

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Vezba_7_Z2
{
    class Prikazi
    {
        public static void DisplayArray<T>(T[] inputArray)
```

```
        {
            foreach (T element in inputArray)
                Console.Write(element + " ");
            Console.WriteLine("\n");
        }
    }
}
```

Slika 7-5. Programski kod klase `Prikazi` .



Slika 7-6. Projekat za Zadatak 2 realizovan u VS-u.

Ispisati sadržaj ekrana konzolnog ekrana u predviđenom prostoru.

Uporedite programske kodove i dobijene rezultate iz Zadatka 1 i Zadatka 2.

Teorijska pitanja 1

Koji imenski prostor je obavezno učitati? U čemu je razlika između ovih zadataka?

Navesti dobre osobine generičkih klasa i metoda.

Objasnite notaciju koja se koristi kod generičkih metoda.

Objasnite način korišćenja generičkih metoda.

Navesti nedostatke korišćenja generičkog objekta klase `Object`.

Da li generička metoda može biti preklopljena? Da li generička metoda može biti preklopljena negeneričkom metodom?

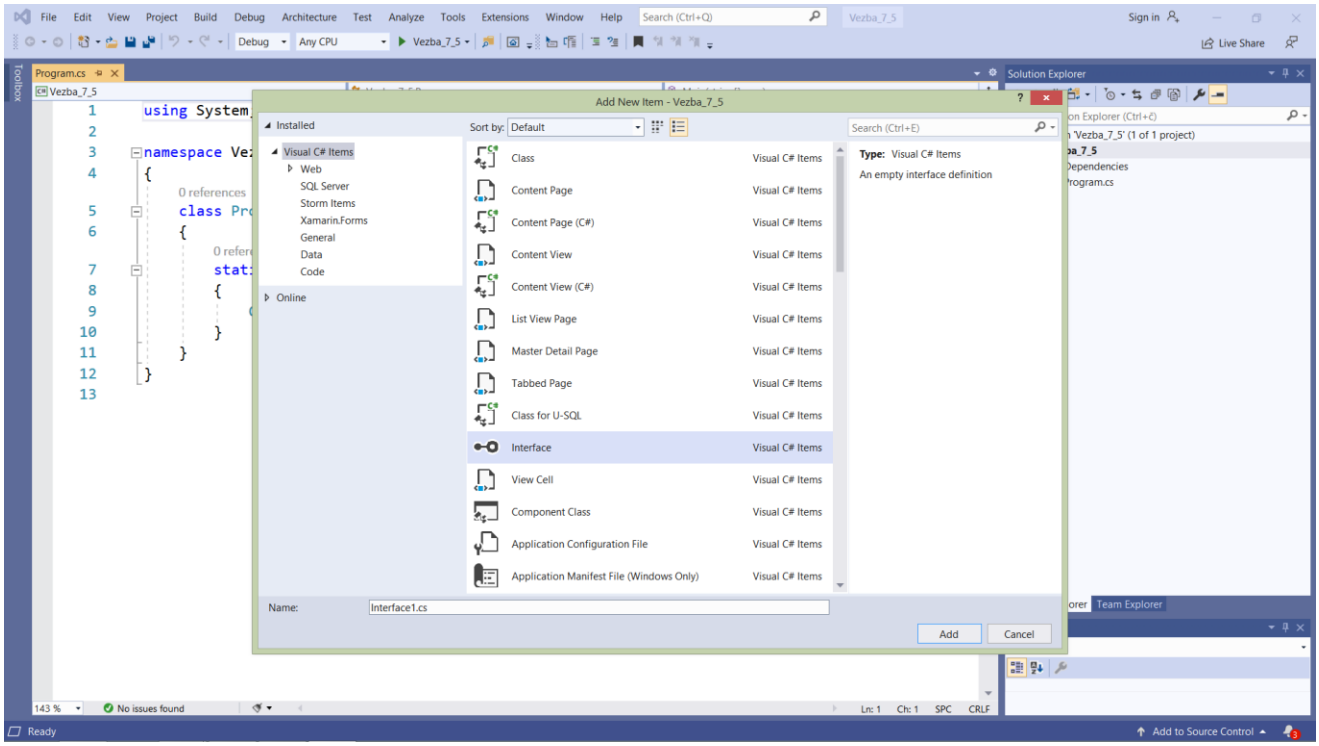
Zadatak 3: Kreiranje intefejsa `IFunkcija`

Keirajte projekat tipa „Console App (.NET Core)“ pod imenom Vezba_7_Z3. Ovom projektu dodajte (kreirajte) interfejs `IFunkcija` sa metodama `VrednostFunckije` koja kao argument ima zadatu - celobrojnu vrednost `x`, i `Stampaj` koja ispisuje formu odgovarajuće funkcije (npr. “`y = 2x+3`”). Na slici 7-7 prikazan je programski kod interfejsa. Uneti ovaj kod u projekat koji treba realizovati u VS-u koristeći dodatni alat za kreiranje interfejsa koji je prikazan na slici 7-8. Da bi

aktivirali alat za dodavanje interfejsa projektu potražite: `Project/Add New Item ...`. Keirajte projekat tipa „Console App (.NET Core)“ pod imenom Vezba_7_Z3.

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Vezba_7_Z3
{
    class Interfejs
    {
        interface IFunkcija
        {
            int Vrednost_Funkcije(int x);
            void Stampaj();
        }
}
```

Slika 7-7. Izvorni kod interfejsa IFunkcija.



Slika 7-8. Prozor za kreiranje interfejsa u VS-u.

Zadatak 4: Implementacija intefejsa `IFunkcija`

Na osnovu kreiranog interfejsa `IFunkcija` iz Zadtaka 3 formirati klase `Linearna` i `Kvadratna`. U izvedenim klasama kreirati odgovarajuće konstruktore koji inicijalizuju odgovarajući broj koeficijenata (kod linearne 2, a kod kvadratne funkcije 3). Implementirati interfejs `IFunkcija` i realizovati implementirane metode. U glavnom programu napraviti po dva objekta tipa `Linearna` i `Kvadratna` i pozvati kreirane metode. Na slikama od 7-9 i 7-10 prikazan su izvorni kodovi ovog projekta. Na slici 7-11 prikazan je izgled projekta ovog

zadatka u VS-u, dok je na slici 7-12 prikazan izgled konzolnog prozora posle startovanja aplikacije.

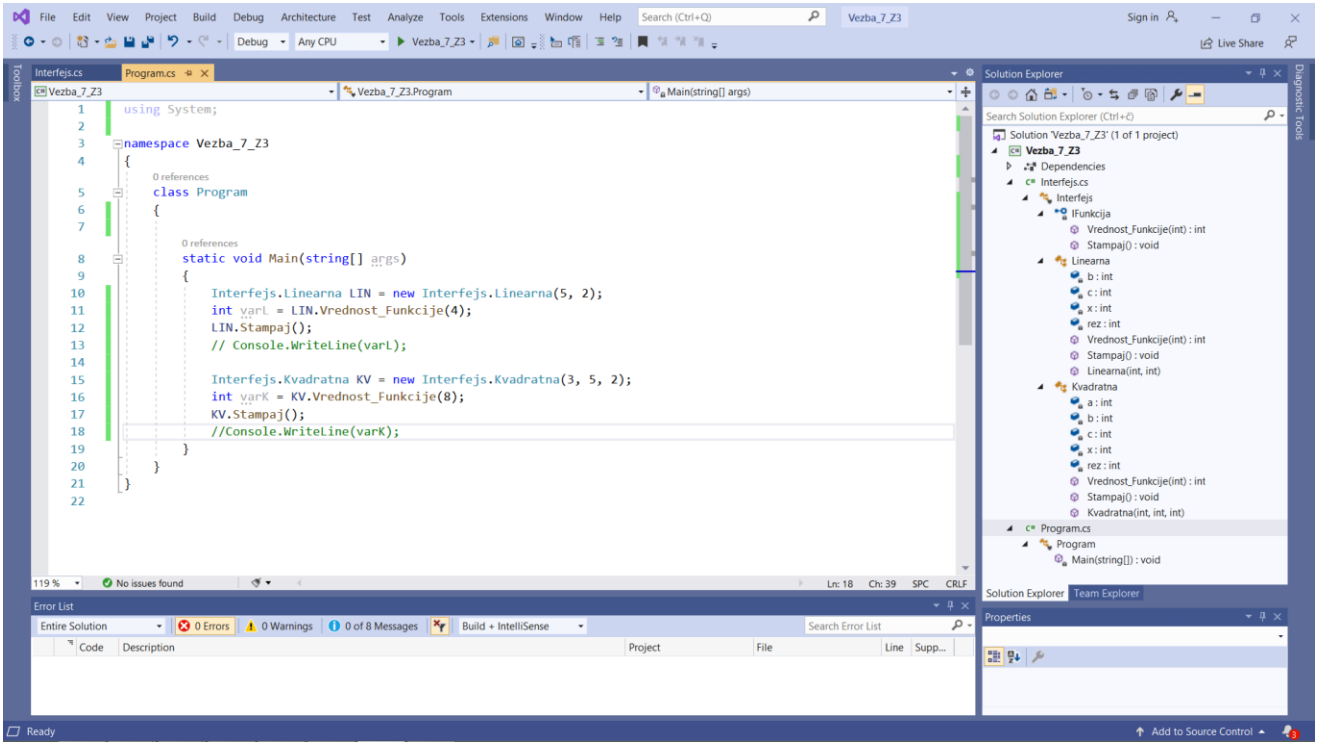
```
using System;
using System.Collections.Generic;
using System.Text;

namespace Vezba_7_Z3
{
    class Interfejs
    {
        interface IFunkcija
        {
            int Vrednost_Funkcije(int x);
            void Stampaj();
        }

        public class Linearna : IFunkcija
        {
            int b; int c; int x; int rez;

            public int Vrednost_Funkcije(int xx)
            {
                this.x = xx;
                this.rez = (b * x) + c;
                return rez;
            }

            public void Stampaj()
            {
                Console.WriteLine("f(x)={0}*x+{1}", b, c);
                Console.WriteLine("f({0})={1}*{0}+{2}={3}", x, b, c, rez);
            }

            public Linearna(int bb, int cc)
            {
                this.b = bb;
                this.c = cc;
            }
        }

        public class Kvadratna : IFunkcija
        {
            int a; int b; int c; int x; int rez;
            public int Vrednost_Funkcije(int xx)
            {
                this.x = xx;
                this.rez = (a * x * x) + (b * x) + c;
                return rez;
            }

            public void Stampaj()
            {
                Console.WriteLine("f(x)={0}x^2+{1}x+{2}", a, b, c);
                Console.WriteLine("f({0})={1}*{0}^2+{2}*{0}+{3}={4}", x, a, b, c, rez);
            }

            public Kvadratna(int aa, int bb, int cc)
```

```
            {
                this.a = aa;
                this.b = bb;
                this.c = cc;
            }
        }
    }
}
```

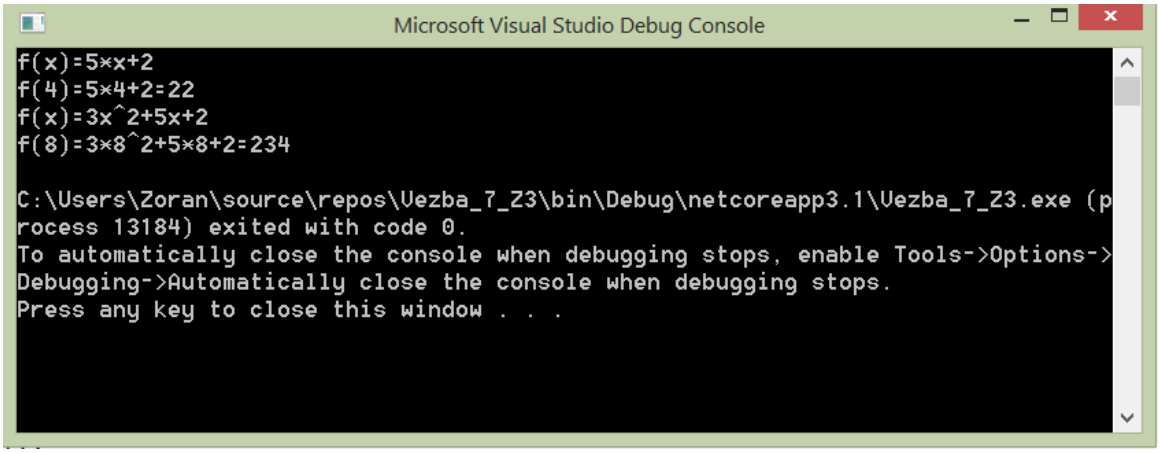
Slika 7-9. Projekat za Zadatak 2 realizovan u VS-u.

```
using System;

namespace Vezba_7_Z3
{
    class Program
    {

        static void Main(string[] args)
        {
            Interfejs.Linearna LIN = new Interfejs.Linearna(5, 2);
            int varL = LIN.Vrednost_Funkcije(4);
            LIN.Stampaj();

            Interfejs.Kvadratna KV = new Interfejs.Kvadratna(3, 5, 2);
            int varK = KV.Vrednost_Funkcije(8);
            KV.Stampaj();
        }
    }
}
```

Slika 7-10. Projekat za Zadatak 2 realizovan u VS-u.



Slika 7-11. Projekat za Zadatak 4 realizovan u VS-u.



Slika 7-12. Projekat za Zadatak 4 realizovan u VS-u.

Izmenite koeficijente jednačina da odgovaraju broju Vašeg indeksa. Prikazati dobijena rešenja u za to predviđenom prostoru.

Zadatak 5: Kreiranje i implementacija generičkog intefejsa `IEquatible<T>`

Formirati klasu `Car`, koja se satoji od privatnih polja `Make`, `Model` i `Year`, tipa string, podrazumevanog konstruktora bez parametara, konstrukotra sa parametrima i konstruktora za kopiranje, kao i svojstava `_Make`, `_Model` i `_Year`. Klasa `Car` treba da implementira ugrađeni interfejs `IEquatible<T>`, koji se sastoji od generičke metode `bool Equals(T obj)`, koja poredi jednakost dva objekta. U glavnom programu formirati nekoliko objekata tipa `Car` i testirati ih na jednakost.

Kreirajte projekat tipa „Console App (.NET Core)“ pod imenom Vezba_7_Z5. Da bi dodali interfejs projektu potražite `Project/Add Neww Item ...`. Popunite identifikator interfejsa i unesite programski kod prikazan na slikama 7-13 do 7-15.

Na slici 7-13 prikazan je programski kod generičkog interfejsa `IEquatible<T>`.

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Vezba_7_5
{
    interface IEquatible<T>
    {
        bool Equals(T obj);
    }
}
```

Slika 7-13. Programski kod interfejsa `IEqutable`.

Na slici 7-14 prikazan je programski kod klase `Car` koja realizuje interfejsa `IEquatible<T>`.

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Vezba_7_5
{
    class Car : IEquatible<Car>
    {
        private string Make;
        private string Model;
        private string Year;

        public Car()
        {
            Make = null;
            Model = null;
            Year = null;
        }

        public Car(Car c)
        {
            Make = c.Make;
            Model = c.Model;
            Year = c.Year;
        }

        public Car(string make, string model, string year)
        {
            Make = make;
            Model = model;
            Year = year;
        }

        public string _Make
        {
            get { return Make; }
            set { Make = value; }
        }

        public string _Model
        {
```

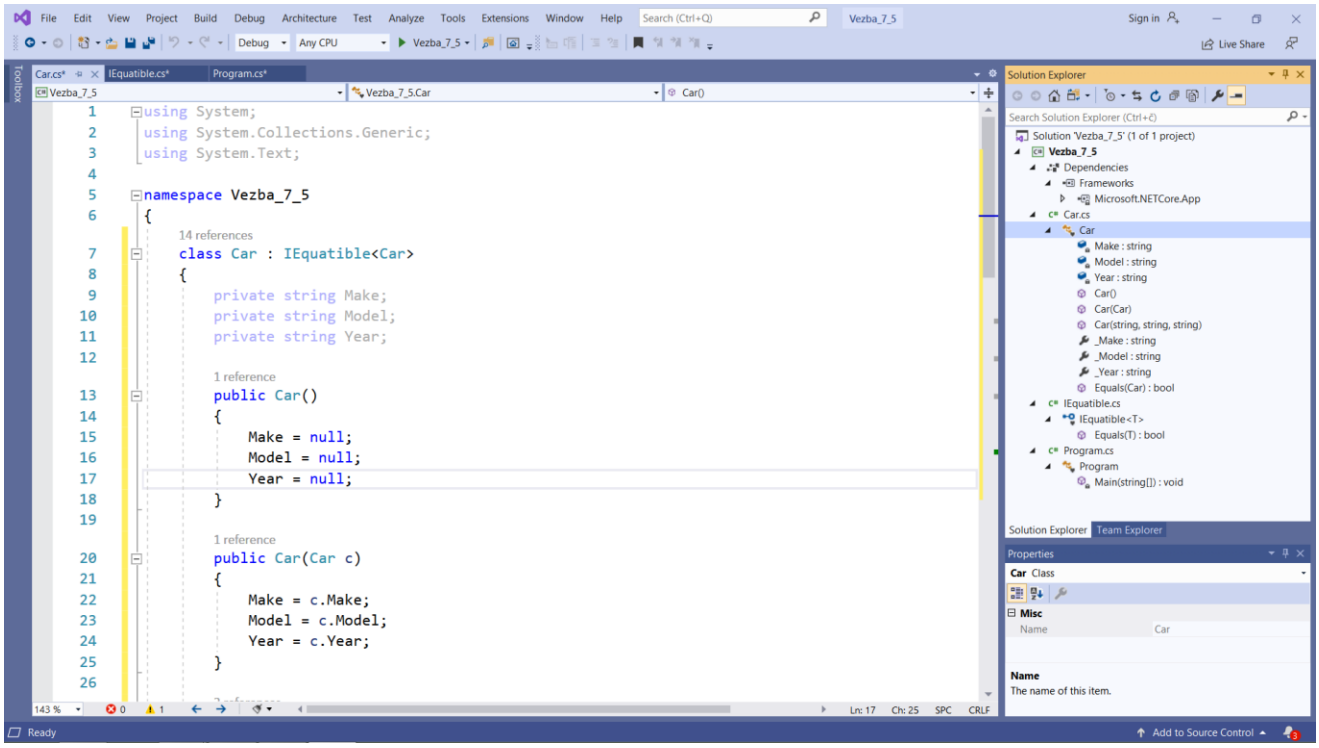
```
            get { return Model; }
            set { Model = value; }
        }

       public string _Year
        {
            get { return Year; }
            set { Year = value; }
        }

        public bool Equals(Car car)
        {
            if (this.Make == car.Make && this.Model == car.Model && this.Year == Year)
            {
                return true;
            }
            else
                return false;
        }
    }
}
```

Slika 7-14. Programski kod klase `Car` koja realizuje generički interfejs `IEqutable`.

Na slici 7-15 prikazan je programski kod za primenu klase `Car`.

```
using System;
namespace Vezba_7_5
{
    class Program
    {
        static void Main(string[] args)
        {
            Car c1 = new Car("Europe", "BMW", "1998");
            Car c2 = new Car("USA", "Chevrolete", "1998");
            Car c3 = new Car(c1);
            Car c4 = new Car();
            c4 = c1;

            bool rez1 = c1.Equals(c2);
            Console.WriteLine(rez1);

            c3._Make = "ASIA";
            c3._Model = "KIA";

            bool rez2 = c1.Equals(c3);
            Console.WriteLine(rez2);

            c4._Make = "ASIA";
            c4._Model = "KIA";

            bool rez3 = c1.Equals(c4);
            Console.WriteLine(rez3);
        }
    }
}
```

Slika 7-15. Programski kod za testiranje klase `Car`.



Slika 7-14. Projekat za Zadatak 5 realizovan u VS-u.

Ispisati sadržaj ekrana konzolnog ekrana u predviđenom prostoru.

Zašto su potrebni konstruktori za kopiranje?

Zbog čega promenljiva `rez3` na kraju programa ima vrednost `true`?

Zadaci za samostalan rad

1. Kreirati generičku klasu `Stack`. Klasa Stack treba da obezbedi metode `Push()` i `Pop()`. Konstruktorom definisati veličinu steka. Testirati realizovane klase za promenljive tipa `double` i `string`.

Prostor za programski kod

2. Kreirati generički interfejs `IBook<T>` sa sledećim metodama `add(T book)`, `delete()` i `T get()`. Kreirati klasu generičku klasu `Book<T>` koja implementira interfejs `IBook<T>`. U klasi za testiranje obaviti manipulaciju sa objektima `Book`.

Prostor za programski kod

U Nišu	POTVRĐUJE