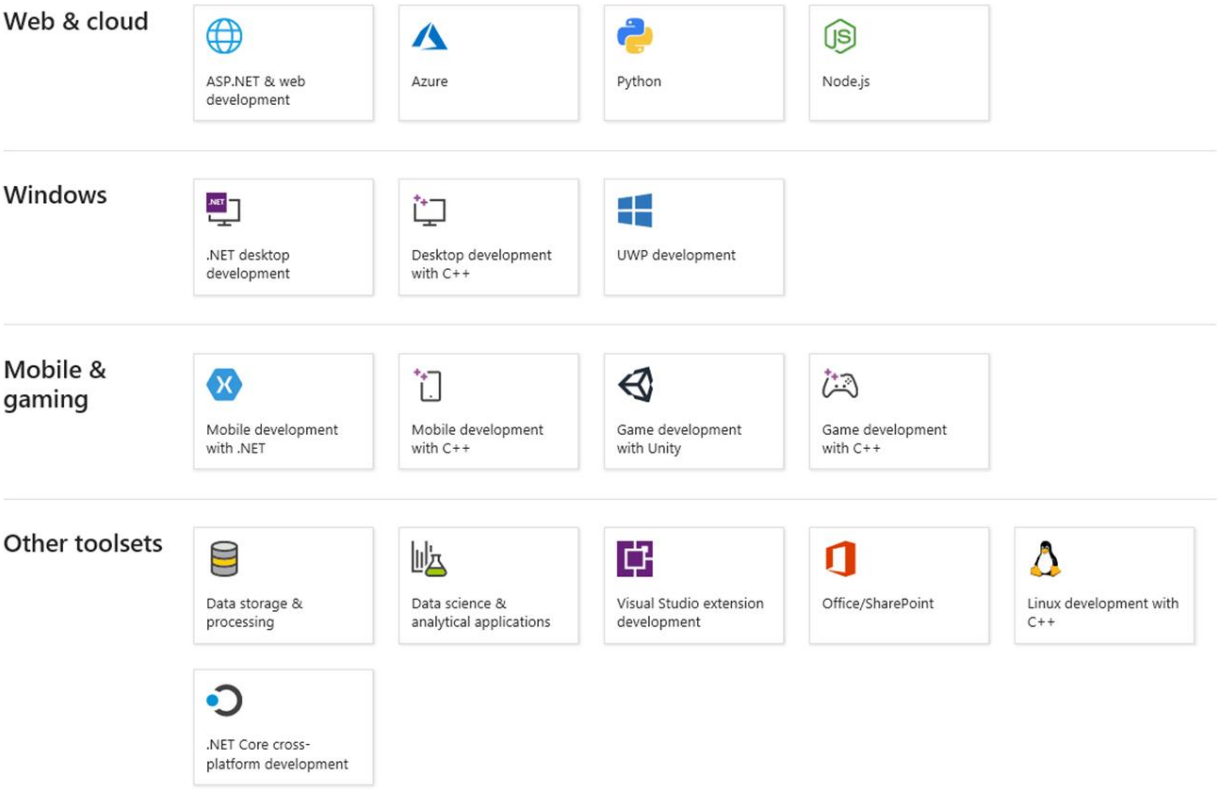
Prof. dr Zoran S. Veličković, dipl. inž. el.

.NET TEHNOLOGIJE

praktikum laboratorijskih vežbi



Niš, novembar 2021.

Student: _____

Broj indeksa: _____

Broj poena: _____

Potvrđuje: _____

NAPOMENE

LABORATORIJSKA VEŽBA 6: NASLEĐIVANJE I POLIMORFIZAM

Cilj vežbe: Upoznavanje sa osnovnim pojmovima vezanim za nasleđivanje klase i polimorfizam. Primena virtualnih i apstraktnih metode i klasa u programskom jeziku C#. Redefinisanje i kreiranje novih metoda kroz proces nasleđivanja.

Zadatak 1: Kreiranje klase `Tacka` sa svojstvima za postavljanje i čitanje

Kreirati projekt tipa „Console App (.NET Core)“ pod imenom Vezba_6_Z1. Način kreiranja ovog tipa projekta je opisan u prethodnim vežbama. U okviru projekta kreirati klasu `Tacka` koja sadrži koordinate date tačke (`x` i `y`) tipa `private double`, podrazumevani konstruktor bez argumenata, kao i parametarski konstruktor kojim se postavljaju vrednosti koordinata tačke. Takođe, kreirajte svojstva (`X` i `Y`) za pristup i definisanje koordinata tačke, kao i metodu `Distanca`, koja izračunava razdaljinu u odnosu na podrazumevanu tačku - početak koordinatnosg sistema. U glavnom programu formirati dve tačke, odštampati njihove koordinate i rastojanje između njih.

Programski kod klase `Tacka` prikazan je na slici 6-1.

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

namespace Vezba_6_Z1
{
    class Tacka
    {
        private double x;
        private double y;

        public Tacka()
        {
            x = 0;
            y = 0;
        }

        public Tacka(int a, int b)
        {
            x = a;
            y = b;
        }

        public double X
        {
            get
            {
                return x;
            }
            set
            {
                x = X;
            }
        }

        public double Y
        {
```

```
            get
            {
                return y;
            }
            set
            {
                y = Y;
            }
        }

        public double Distanca(Tacka T2)
        {
            double rastojanje = Math.Sqrt(Math.Pow((this.x + T2.x), 2) + Math.Pow((this.y +
T2.y), 2));

            return rastojanje;
        }
    }
}
```

Slika 6-1. Programski kod klase `Tacka`.

Programski kod klase `Program` za testiranje napisane klase je prikazan na slici 6-2.

```
namespace Vezba_6_Z1
{
    class Program
    {
        static void Main(string[] args)
        {
            Tacka T1 = new Tacka();
            Tacka T2 = new Tacka(10, 5);
            Console.WriteLine("Koordinate tacke T1 su [" + T1.X + "," + T1.Y + "]");
            Console.WriteLine("Koordinate tacke T2 su [" + T2.X + "," + T2.Y + "]");

            double distanca = T1.Distanca(T2);
            Console.WriteLine("Razdaljina izmedju T1 i T2 je " + distanca);
        }
    }
}
```

Slika 6-2. Programski kod klase za testiranje klase `Tacka`.

Na slici 6-3 prikazan je izgled VS-a projekta kreiranog za Zadatak 1. Opišite arhitekturu kreiranog projekta.

_____

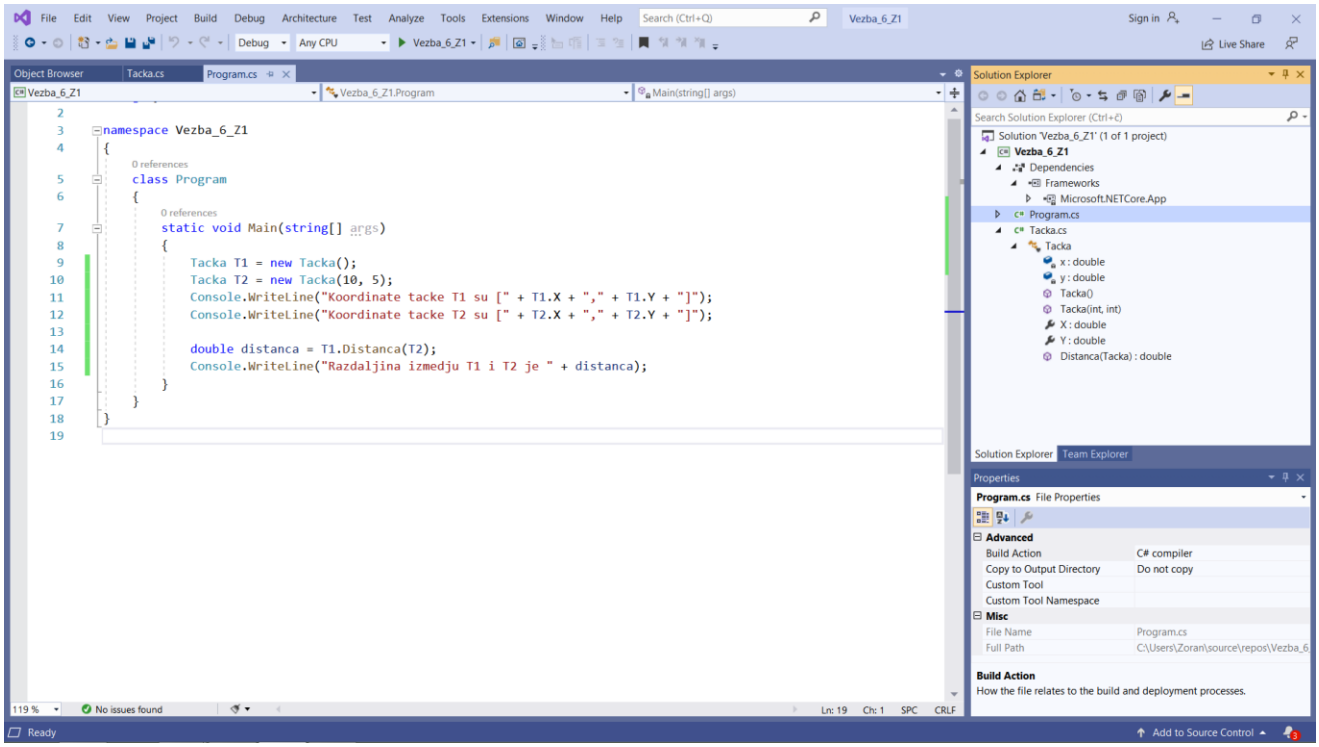
_____

_____

_____

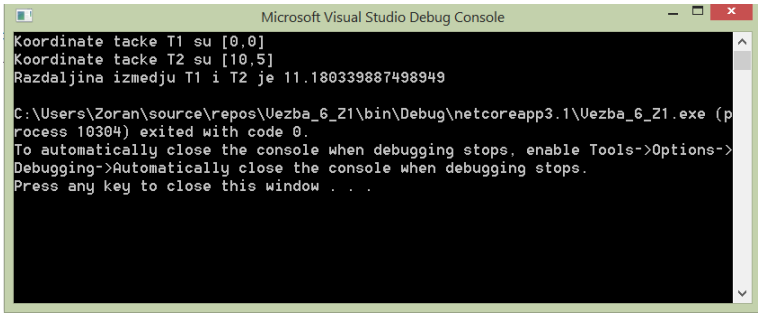
_____

_____



Slika 6-3. Izgled VS projekta kreiranog za Zadatak 1.

Na slici 6-4 prikazan je konzolni prozor u kojoj aplikacija ispisuje tražene vrednosti.



Slika 6-4. Izgled konzolnog prozora posle starovanja Zadataka 1.

Zadatak 2: Kreiranje univerzalne metode `Distance`

Kreirati univerzalnu metodu `Distance` koja računa rastojanje između dve proizvoljne tačke. Napisati programski iskaz kojim se realizuje ova funkcija.

Zadatak 3: Kreiranje sopstvenih objekata klase `Tacka`

Koristeći već napisani kod, kreirajte potreban broj objekata klase `Tacka` koji definišu a) kvadrat sa dužinom stranice 3 b) pravougaonik čiji je odnos stranica 3:1 c) jednakostranični trougao sa dužinom stranice 2. Zadajte koordinate tačaka koje zadovoljavaju tražene uslove u predviđenom prostoru.

a)

b)

c)

Koje korekcije u izvornom kodu treba načiniti da se zadovolje traženi uslovi?

_____

_____

_____

Proverite da li prikazane vrednosti odgovaraju zahtevu zadatka startovanjem programa. U predviđenom prostoru štampajte rezultate koji potvrđuju ispunjenost uslova.

Teorijska pitanja 1

Kako je omogućen pristup koordinatama tačke iz glavnog programa?

_____

_____

_____

Kako se obavlja preklapanje konstruktora i čemu preklapanje konstruktora služi?

_____

_____

_____

Zašto je uveden koncept modifikatora pristupa u OOP-u?

_____

_____

_____

Koje objekte formiramo u glavnom programu i u kojim linijama koda?

Koje metode sadrži klasa `Tačka`?

Koje matematičke funkcije obavljaju metode `Math.Sqrt` i `Math.Pow`?

Na šta se donose vrednosti `this.x` i `this.y` u funkciji `public double Distanca(Tacka T2)`?

U čemu je razlika između modifikatora pristupa: `public`, `protected internal`, `protected`, `internal` i `private`?

Zadatak 4: Kreiranje klase `Krug` nasleđivanjem klase `Tacka`

Kreirati projekt tipa „Console App (.NET Core)“ pod imenom Vezba_6_Z3. U projektu izvesti iz klase `Tacka` (Zadatak 1) klasu `Krug`. Klasa `Krug` treba da poseduje polje `radius` koji sadrži informaciju o poluprečniku kruga, kao i metode za izračunavanje površine i obima kruga. Zatim, iz klase `Krug` izvesti klasu `Valjak` koja treba da poseduje polje `visina`. Preklopiti metodu za izračunavanje površine i dodati metodu za izračunavanje zapremine. U glavnom programu formirati po dva objekta tipa `Tačka`, `Krug` i `Valjak` i odštampati vrednosti njihovih parametara.

NAPOMENA: Možete iskoristiti *copy-paste* tehniku da se formira početna poziciju (klasa `Tacka`) za kreiranje projekta za Zadatak 3.

Programski kod klase `Krug` je prikazan na slici 6-5, dok je na slici 6-6 prikazan kod klase `Valjak`.

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

```
namespace Vezba_6_Z3
{
    class Krug : Tacka
    {
        private double radius;

        // konstruktor 1
        public Krug(): base()
        {
            radius = 0;
        }

        // konstruktor 2
        public Krug(int x, int y, double rad): base(x, y)
        {
            radius = rad;
        }

        // dodatna metoda
        public virtual double Povrsina()
        {
            double P;
            P = Math.Pow(this.radius, 2) * Math.PI;
            return P;
        }

        // dodatna metoda
        public virtual double Obim()
        {
            double O;
            O = radius * 2 * Math.PI;
            return O;
        }
    }
}
```

Slika 6-5. Programski kod izvedene klase Krug.

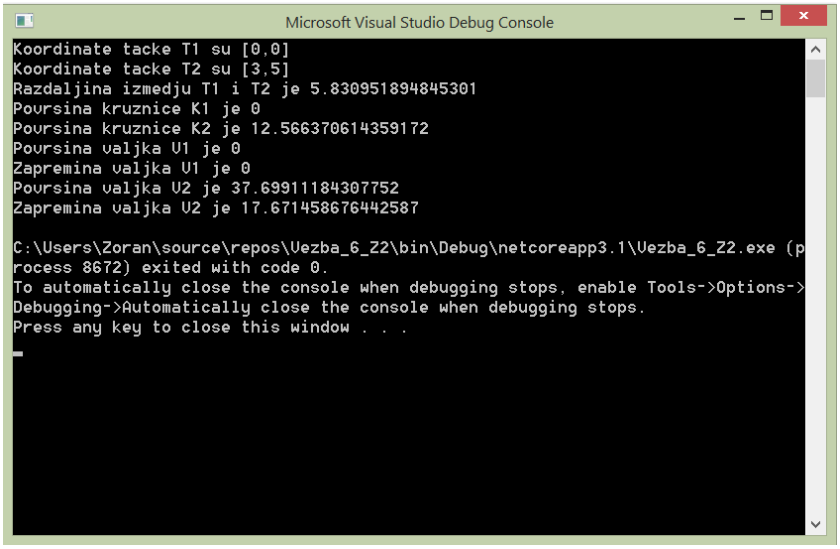
```
using System;
using System.Collections.Generic;
using System.Text;

namespace Vezba_6_Z3
{
    class Valjak : Krug
    {
        public double visina;

        // Konstruktor 1
        public Valjak() : base()
        {
            visina = 0;
        }

        // Konstruktor 2
        public Valjak(int x, int y, double rad, double H) : base(x, y, rad)
        {
            this.visina = H;
        }
```

```
        // preklopljena metoda
        public override double Povrsina()
        {
            double P;
            P = base.Povrsina() * 2 + base.Obim() * visina;
            return P;
        }

        // nova metoda
        public double Zapremina()
        {
            double V;
            V = base.Povrsina() * visina;
            return V;
        }
    }
}
```

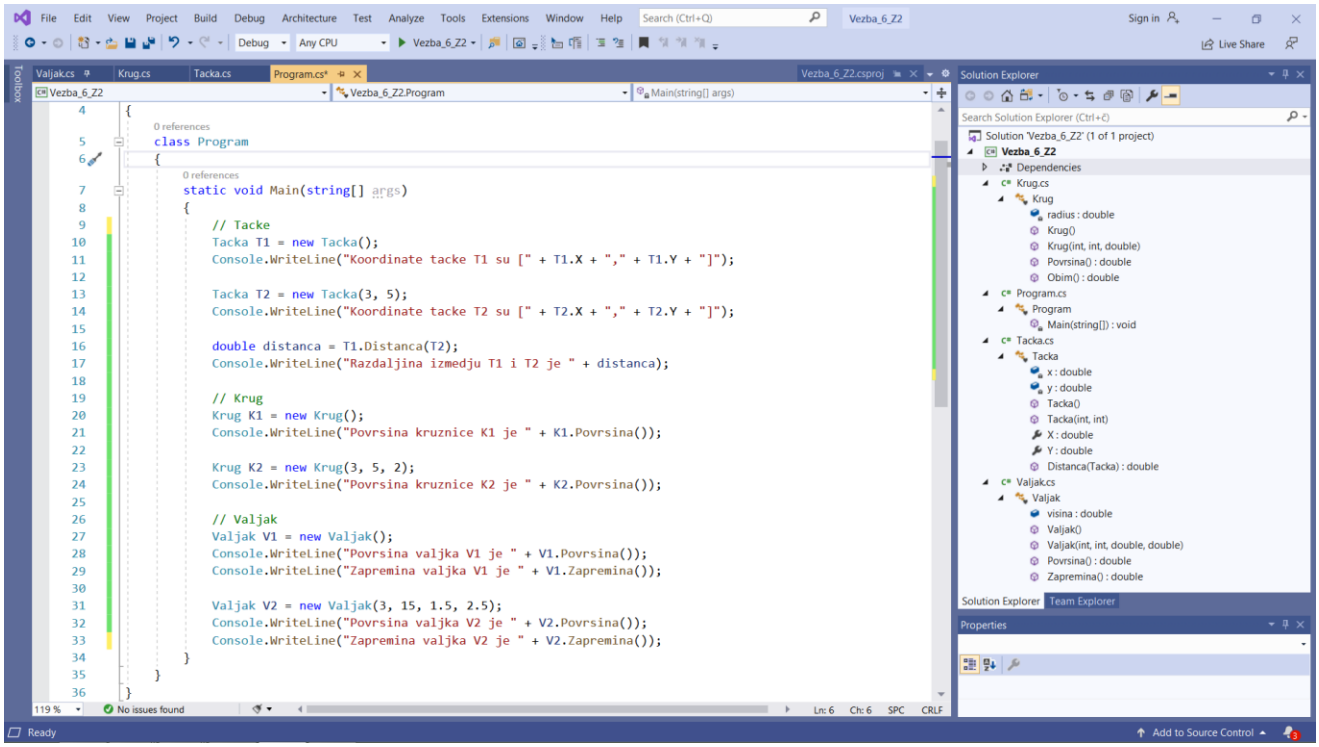
Slika 6-6. Programski kod izvedene klase Valjak.

```
using System;

namespace Vezba_6_Z3
{
    class Program
    {
        static void Main(string[] args)
        {
            // Tacke
            Tacka T1 = new Tacka();
            Console.WriteLine("Koordinate tacke T1 su [" + T1.X + "," + T1.Y + "]");

            Tacka T2 = new Tacka(3, 5);
            Console.WriteLine("Koordinate tacke T2 su [" + T2.X + "," + T2.Y + "]");

            double distanca = T1.Distanca(T2);
            Console.WriteLine("Razdaljina izmedju T1 i T2 je " + distanca);

            // Krug
            Krug K1 = new Krug();
            Console.WriteLine("Povrsina kruznice K1 je " + K1.Povrsina());

            Krug K2 = new Krug(3, 5, 2);
            Console.WriteLine("Povrsina kruznice K2 je " + K2.Povrsina());

            // Valjak
            Valjak V1 = new Valjak();
            Console.WriteLine("Povrsina valjka V1 je " + V1.Povrsina());
            Console.WriteLine("Zapremina valjka V1 je " + V1.Zapremina());

            Valjak V2 = new Valjak(3, 15, 1.5, 2.5);
            Console.WriteLine("Povrsina valjka V2 je " + V2.Povrsina());
            Console.WriteLine("Zapremina valjka V2 je " + V2.Zapremina());

        }
    }
}
```

Slika 6-6. Programski kod klase za testiranje klase `Tacka`, `Krug` i `Valjak`.



Slika 6-7. Izgled konzolnog prozora posle starovanja Zadataka 3.



Slika 6-8. Izgled VS projekta za Zadatak 3.

Na slici 6-8 prikazan izgled projekta koji treba realizaovati za Zadatak 3.

Objasnite programske iskaze:

```
class Valjak : Krug
```

```
public virtual double Povrsina()
```

```
public override double Povrsina()
```

```
Valjak V2 = new Valjak(3, 15, 1.5, 2.5);
```

```
Valjak V1 = new Valjak();
```

```
P = base.Povrsina() * 2 + base.Obim() * visina;
```

Zadatak 5: Kreiranje apstraktne klase `Figure`

Definisati apstraktnu klasu `Figura`, koja poseduje apstraktu metodu `Draw()`. Zatim, iz klase `Figura` izvesti klase `Krug`, `Trougao` i `Kvadrat` i u njima umesto implementacije metode `Draw()` odštampati opis posla. U glavnom programu formirati po jedan objekt od svih kreiranih tipova i za svaki pozvati metodu `Draw()`, a zatim formirati niz referenci na klasu figura, a potom pozvati metode `Draw()`. Zadatak 4 realizovati u projektu tipa „Console App (.NET Core)“. Na slikama 6-9 do 6-12 prikazani su programski kodovi svih klasa .

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

namespace Vezba_6_Z4
{
```

```
    abstract class Figura
    {
        public abstract void Draw();
    }
}
```

Slika 6-9. Programski kod klase `Figura`.

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

namespace Vezba_6_Z4
{
    class Kvadrat : Figura
    {
        public override void Draw()
        {
            Console.WriteLine("Komande za crtanje kvadrata");
        }
    }
}
```

Slika 6-10. Programski kod klase `Kvadrat`.

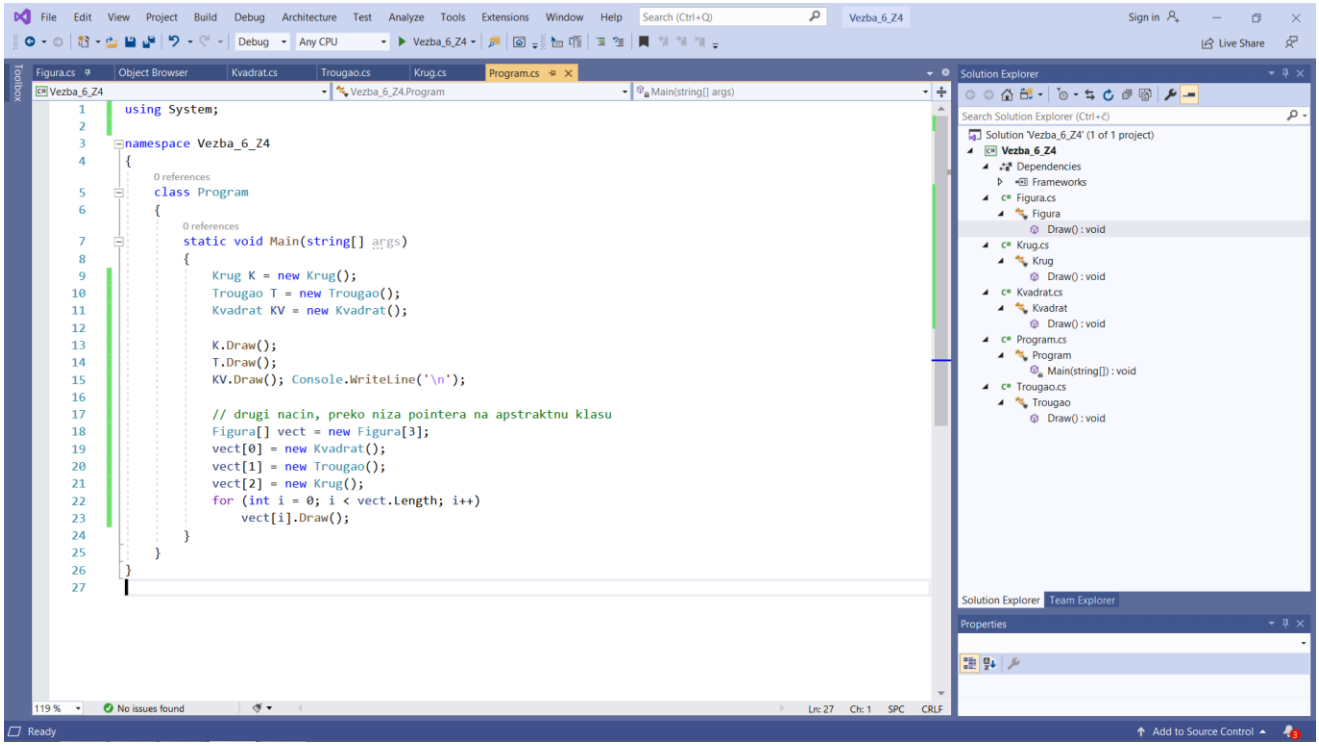
```
using System;
using System.Collections.Generic;
using System.Text;

namespace Vezba_6_Z4
{
    class Trougao : Figura
    {
        public override void Draw()
        {
            Console.WriteLine("Komande za crtanje trougla");
        }
    }
}
```

Slika 6-11. Programski kod klase `Trougao`.

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

namespace Vezba_6_Z4
{
    class Krug: Figura
    {
        public override void Draw()
        {
            Console.WriteLine("Komande za crtanje kruga");
        }
    }
}

using System;

namespace Vezba_6_Z4
{
    class Program
```

```
    {
        static void Main(string[] args)
        {
            Krug K = new Krug();
            Trougao T = new Trougao();
            Kvadrat KV = new Kvadrat();
            K.Draw();
            T.Draw();
            KV.Draw(); Console.WriteLine('\n');
            // drugi nacin, preko niza pointera na apstraktnu klasu
            Figura[] vect = new Figura[3];
            vect[0] = new Kvadrat();
            vect[1] = new Trougao();
            vect[2] = new Krug();
            for (int i = 0; i < vect.Length; i++)
                vect[i].Draw();
        }
    }
}
```

Slika 6-12. Programski kod za testiranje svih kreiranih klasa.



Slika 6-13. Izgled VS projekta za Zadatak 4.

U predviđenom prostoru ispišite tekst iz konzolnog prozora posle startovanja aplikacije.

Teorijska pitanja 2

Čemu služe apstraktne klase?

Kojom linijom koda definišemo apstraktnu klasu?

Čemu služi komanda `virtual`?

Čemu služi komanda `override`?

Objasnite pojam polimorfizma.

Objasnite pojam `seald` klasa.

Zadaci za samostalan rad

1. Proširiti klasu `Figura` dodavanjem apstraktnih metoda za izračunavanje površine i obima za svaku od zadatih figura. U klasama naslednicama dodati potrebna polja, konstruktore, svojstva za pristup i definisanje atributa. Implementirati odgovarajuće metode za izračunavanja površina i obima. U glavnom programu testirati ovako formirane klase.

2. Kreirati odgovarajuću *WinForm* aplikaciju (NET *Framework*) koja omogućava izbor figure za crtanje u padajućim listama.

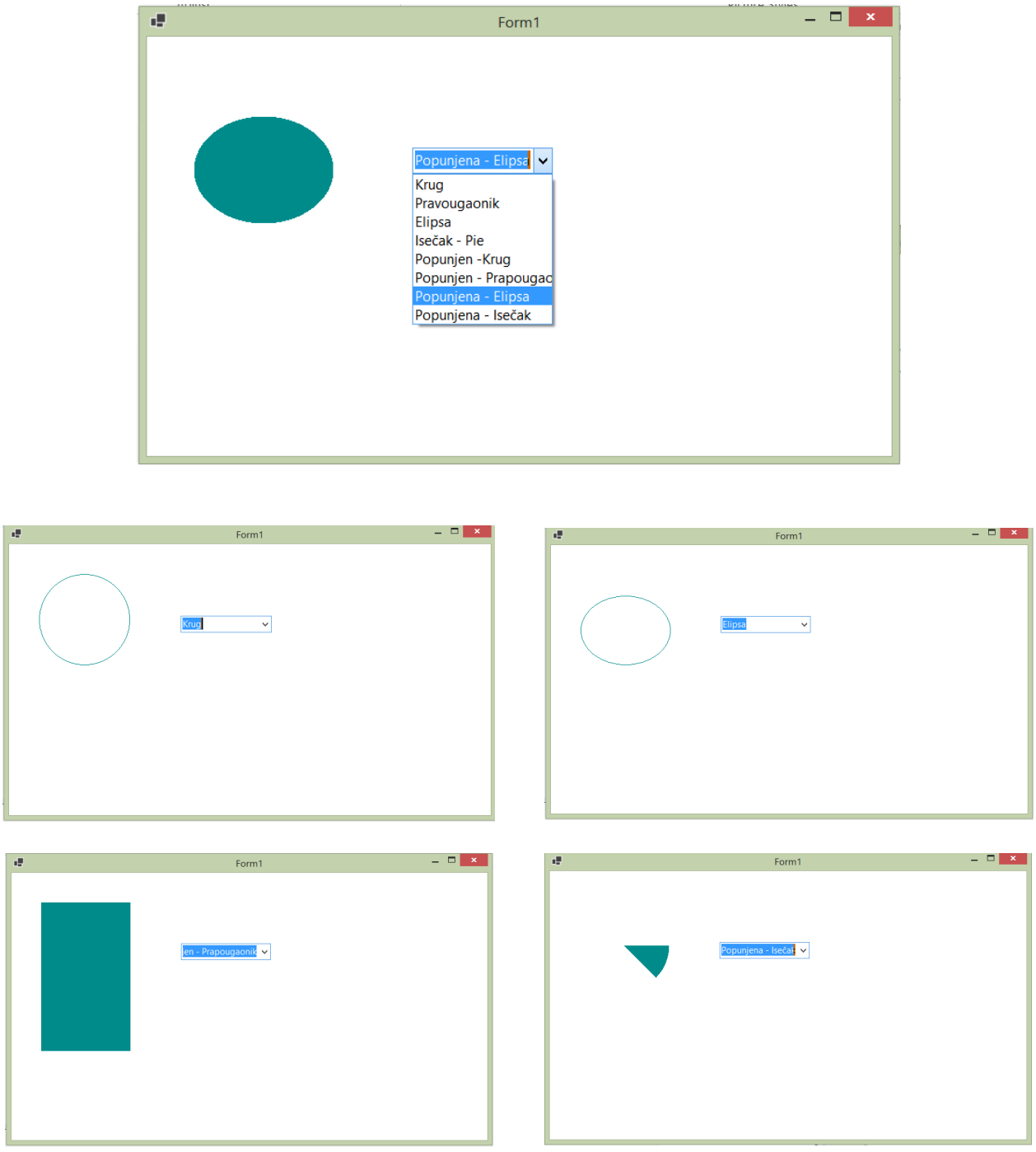
   NAPOMENA:
   Za samo crtanje treba koristite metode klase `CreateGraphics();`.
   Koristeći prozor V.S.-a ObjecBrowser, proučite izabrane metode i svojstva klase `System.Drawing` (GDI+ basic graphics functionality) ili posetite MS stranicu: https://docs.microsoft.com/en-us/dotnet/api/system.drawing?view=netframework-4.8

Klasa/Struktura	Objašnjenje
Bitmap	
Brush	
Brushes	

Pre korišćenja ove klase treba kreirati grafički objekt `myGraphics` na sledeći način `Graphics myGraphics = base.CreateGraphics()`. Za crtanje kruga koristiti metodu `DrawEllipse()` objekta `myGraphics`. Takođe, kreirajte objekt `myPen` na sledeći način `Pen myPen new Pen(Color.DarkRed)`. Samo crtanje ktuga se izvodi programskim iskazom:

```
myGraphics.DrawEllipse(myPen, 50, 50, 150, 150);.
```

Po stratovanju projekta treba da se realizuju prozori kao na sledećim slikama.



Slika 6-14. Izgled .NET Framework aplikacije za samostalni rad.

Izvorni kod realizovane forme je dat na slici 6-15.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace Vezba_6_Z4_Winform
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            // kreiranje grafičkog objekta, Pen i SolidBrush
            Graphics myGrapics = base.CreateGraphics();
            //kreiranje olovke - pen-a korišćenjem boje DarkCyan
            Pen myPen = new Pen(Color.DarkCyan);
            // kreiranje kista sa odgovarajućom bojom
            SolidBrush mySolidBrush = new SolidBrush(Color.DarkCyan);
            // obriši oblast iscrtavanja - bela boja
            myGrapics.Clear(Color.White);

            switch(comboBox1.SelectedIndex)
            {
                case 0:
                    myGrapics.DrawEllipse(myPen, 50, 50, 150, 150);
                    break;
                case 1:
                    myGrapics.DrawRectangle(myPen, 50, 50, 150, 150);
                    break;
                case 2:
                    myGrapics.DrawEllipse(myPen, 50, 85, 150, 115);
                    break;
                case 3:
                    myGrapics.DrawPie(myPen, 50, 50, 150, 150, 0, 45);
                    break;
                case 4:
                    myGrapics.FillEllipse(mySolidBrush, 50, 50, 150, 150);
                    break;
                case 5:
                    myGrapics.FillRectangle(mySolidBrush, 50, 50, 150, 250);
                    break;
                case 6:
                    myGrapics.FillEllipse(mySolidBrush, 50, 85, 150, 115);
                    break;
                case 7:
                    myGrapics.FillPie(mySolidBrush, 50, 50, 150, 150, 0, 45);
                    break;
            }
            myGrapics.Dispose();
        }
    }
}
```

Slika 6-15. Izvorni kod za realizaciju forme sa padajućom listom.

3. Prethodnoj formi dodati polja za zadavanje parameara pojedinim grafičkim formama. Broj polja treba da je u funkciji grafičke forme. U predviđenom delu prikazati izvorni kod realizovanog zadatka.

U Nišu	POTVRĐUJE