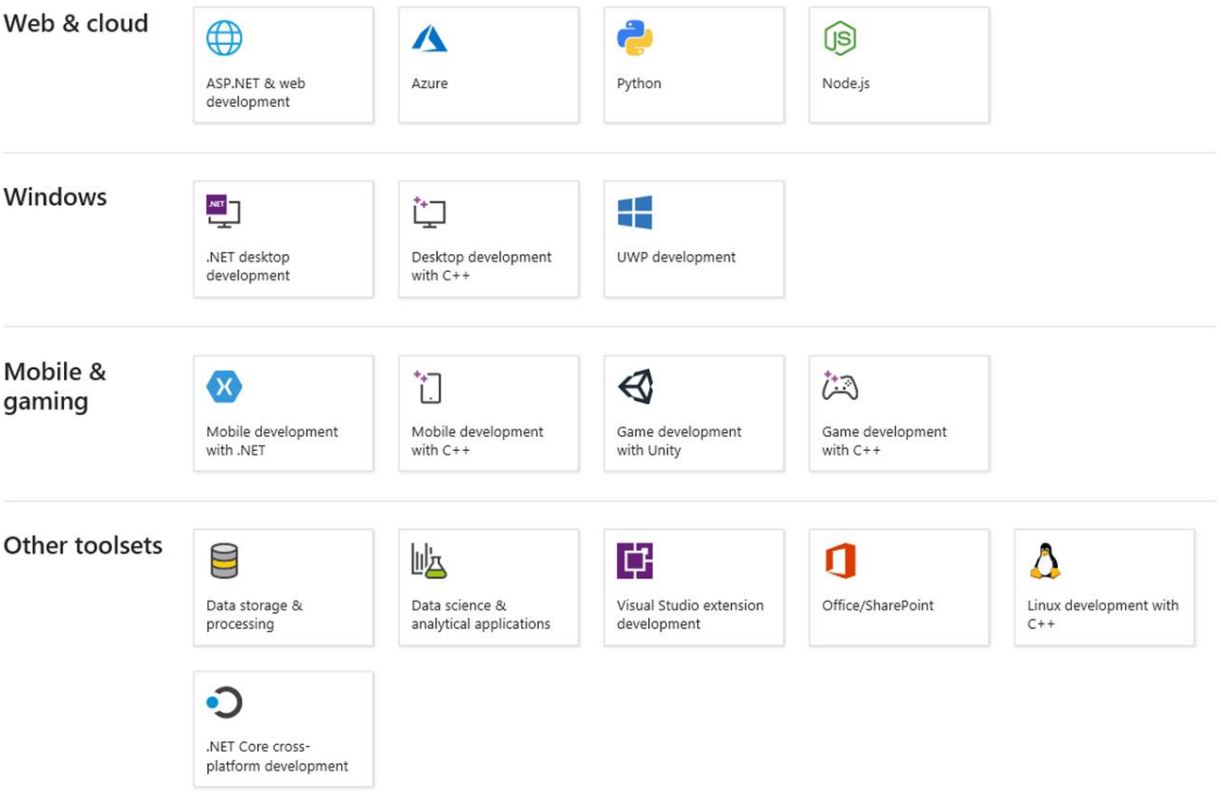
Prof. dr Zoran S. Veličković, dipl. inž. el.

.NET TEHNOLOGIJE

praktikum laboratorijskih vežbi



Niš, novembar 2021.

PREDGOVOR

Praktikum za laboratorijske vežbe iz predmeta ”NET tehnologije” je namenjen studentima studijskih programa Savremene računarske tehnologije Akademije tehničko-vaspitačkih strukovnih studija odseka u Nišu. Osnovni cilj laboratorijskih vežbi je da se teorijska poglavlja upotpune praktičnim znanjima. Iz ovog razloga, laboratorijske vežbe predstavljaju OBAVEZNI deo nastave. Laboratorijske vežbe su prilagođene raspoloživim tehničkim resursima i pokrivaju najvažnija teorijska poglavlja. Koncepcija ovog Praktikuma je da se studenti postepeno uvode u sve složenije zadatke uz istovremeno davanje korisnih uputstava i saveta pri samoj realizaciji vežbe. Da bi se postigao maksimalni učinak ovih vežbi, neophodna je teorijska priprema studenata. Spremnost studenata za laboratorijske vežbe proverava predmetni nastavnik ili asistent. Nespremnim studentima neće se dozvoliti izrada laboratorijskih vežbi. Preduslov započinjanja naredne laboratorijske vežbe je kompletiranje prethodne.

U Praktikumu su izloženi osnovni teorijski principi .NET tehnologija. Posebno su obrađene sledeće oblasti: koncep .NET-a, tipova .NET aplikacija, sintaksa, operatori i naredbe u C#, rad sa stringovima u .NET-u, Windows kontrole, klase i objekti u C#, nasleđivanje i polimorfizam, strukture, interfejsi i delegati, obrada izuzetaka, ...

Sve oblasti su propraćene programskim kodom za primere koji se odnose na izloženu materiju. Kao ilustracija rezultata primenjenih tehnologija, dat je izgled pojedinih prozora u VS-u. Zadaci za vežbu sa odgovarajućim programskim kodom su prikazani nakon teorijskog dela iza koga slede zadaci za samostalni rad. Na kraju vežbe nalazi se spisak pitanja na koje studenti treba daju odgovore. Vežbe se obavljaju na računaru u računarskim laboratorijama Akademije. Realizaciju vežbi je moguće obaviti u razvojnim i izvršnim okruženjima Visual Studio-a, Studio Code-a ili Visual Studia za MAC. Većina laboratorijskih vežbi je realizovana u radnom okruženju .NET Core, a deo u radnom okruženju .NET Framework.

Za efikasnu izradu vežbi potrebno je proučiti odgovarajuću oblast i odgovoriti pismenim putem na postavljena pitanja u samom Praktikumu. Takođe, studenti su obavezni da poseduju sopstvene kopije realizovanih programskih celina na nekom od raspoloživih memorijskih skladišta. Individualni specifični zahtevi za pojedine vežbe biće dati studentima prilikom realizacije vežbi. Ovo će omogućiti da studenti steknu specifična individualna znanja i ovladavaju veštinama potrebnim za samostalno obavljanje poslova i zadataka iz oblati razvoja aplikacija u

.NET-u. Vežbe se delom sastoje od već pripremljenih kodova na kojima studenti treba da razviju sopstvene aplikacije. Takođe, predviđene su vežbe kod kojih studenti treba sami da oforme odgovarajući kod, izgled i funkcionalnost aplikacije.

Student ne sme početi laboratorijsku vežbu pre nego što za to dobije dozvolu. U toku izrade laboratorijskih vežbi, studenti moraju poštovati pravila o radu u laboratoriji kako bi se izbegle situacije koje mogu naneti štetu bilo studentima, bilo opremi.

Posle završenih laboratorijskih vežbi, studenti polažu završni kolokvijum koji se odnosi na odbranu predatih izveštaja. Preduslov za polaganje ispita ".NET tehnologije" jesu OBAVLJENE i OVERENE laboratorijske vežbe od strane predmetnog nastavnika ili asistenta, čime se potvrđuje da je student uspešno obavio predviđene predispitne obaveze u laboratoriji.

Niš, Novembar 2021. Autor

SADRŽAJ

LABORATORIJSKA VEŽBA 9: DELEGATI I DOGAĐAJI U .NET-u

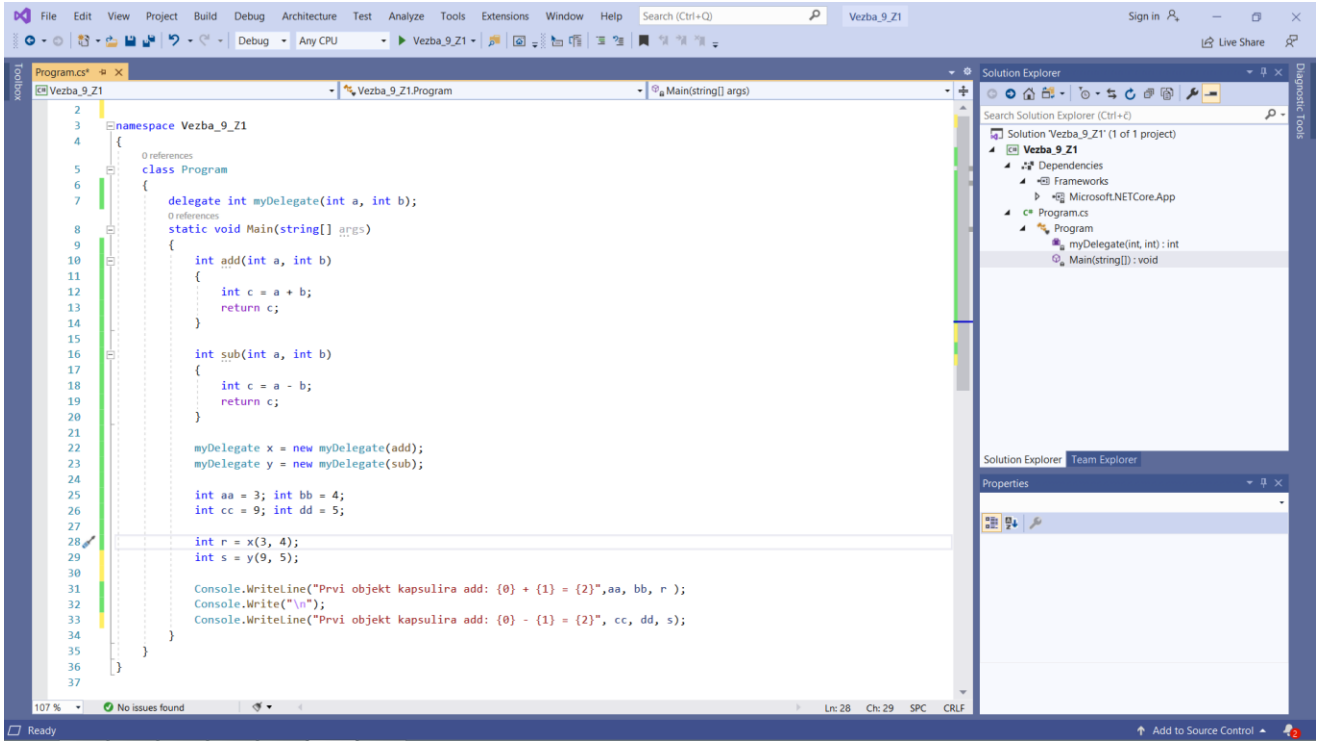
Cilj vežbe: Upoznavanje sa osnovnim pojmovima vezanim za delegate i događaje u .NET-u. Koncept i realizacija višesmernih delegata. Izrada i povezivanje metode za obradu događaja. Upoznati se sa generičkim delegatima i izradom korisničkih delegata i događaja.

Zadatak 1: Primer jednostavnog delegata

Izraditi jednostavn delegat `myDelegate` koji može predstavljati sve metde koje imaju `int` povratni tip sa potpisom koji se sastoji od dva `int` tipa. Proizvoljno izabrati tip aplikacije. Posle kreiranja delegata kreirati dva objekta koji kapsuliraju metode sabiranja `add` i oduzimanja `sub`.

Prikazati rezultate dobijene izvršavanjem ovog programa zamenom parametara koji odgovaraju Vašem broju indeksa. Rezultate formatirano prikazati na komandnoj liniji.

Na slici 9-1 prikazan je izgled realizacije ovog zadatka u VS 2019 za konzolnu aplikaciju realizovanu u .NET Core-u.



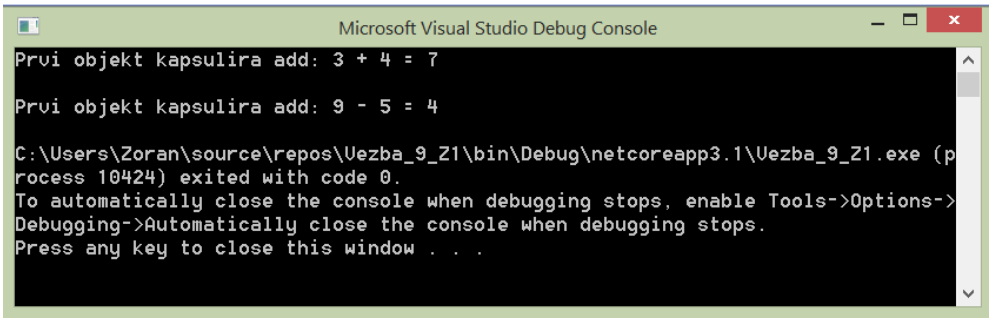
Slika 9-1. Projekat programajednostavnog delegata u VS-u.

Na slici 9-2 prikazan je izvorni kod za realizaciju Zadatka 1 kao konzolne aplikacije. Na slici 9-3 prikazan je izgled komandnog prozora posle startovanja aplikacije.

```
using System;

namespace Vezba_9_Z1
{
    class Program
    {
```

```
        delegate int myDelegate(int a, int b);
        static void Main(string[] args)
        {
            int add(int a, int b)
            {
                int c = a + b;
                return c;
            }

            int sub(int a, int b)
            {
                int c = a - b;
                return c;
            }

            myDelegate x = new myDelegate(add);
            myDelegate y = new myDelegate(sub);

            int aa = 3; int bb = 4;
            int cc = 9; int dd = 5;

            int r = x(3, 4);
            int s = y(9, 5);

            Console.WriteLine("Prvi objekt kapsulira add: {0} + {1} = {2}",aa, bb, r );
            Console.Write("\n");
            Console.WriteLine("Prvi objekt kapsulira sub: {0} - {1} = {2}", cc, dd, s);
        }
    }
}
```

Slika 9-2. Izvorni kod programa za Zadatak 1.



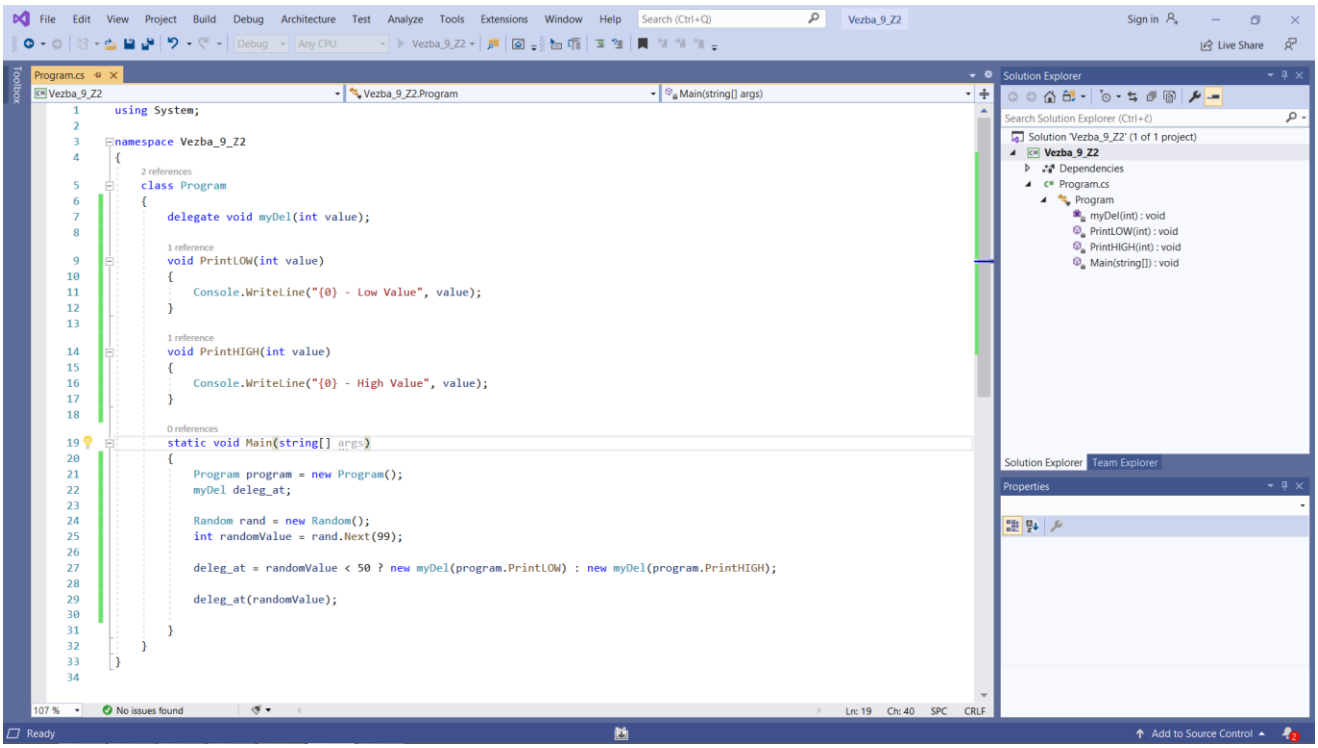
Slika 9-3. Izgled konzolnog prozora posle startovanja aplikacije Zadatak 1.

Prostor za prikaz konzolnog prozora za sa Vašim parametrima

Zadatak 2: Delegat `myDel`

Izraditi jednostavn delegat `myDel` koji može predstavljati sve metde koje imaju `void` povratni tip sa potpisom koji se sastoji od jednog `int` tipa. Proizvoljno izabrati tip aplikacije. Delegat predstavlja metode koje formatirano štampaju predefinisani tekst u funkciji vrednosti parametra. Vrednost parametra kreirati klasom `Random` u opsegu od 0-100. Na osnovu slučajno generisane vrednosti parametara u glavnom programu kreirati odgovarajući objekt, a potom pozvati kapsuliranu metodu u tom objektu.

Na slici 9-4 prikazan je izgled realizacije ovog zadatka u VS 2019 za konzolnu aplikaciju realizovanu u .NET Core-u.



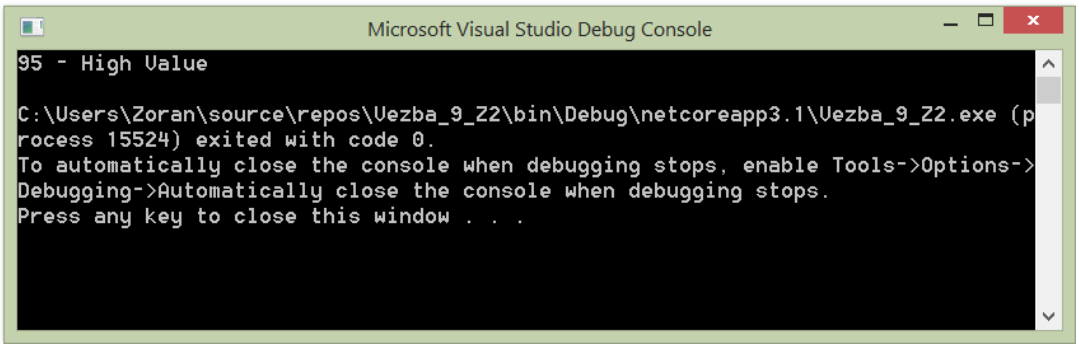
Slika 9-4. Projekat programa za Zadatak 2 u VS-u.

Na slici 9-5 prikazan je izvorni kod za realizaciju Zadatka 2 kao konzolne aplikacije, dok je na slici 9-6 prikazan izgled komandnog prozora posle startovanja aplikacije dva puta.

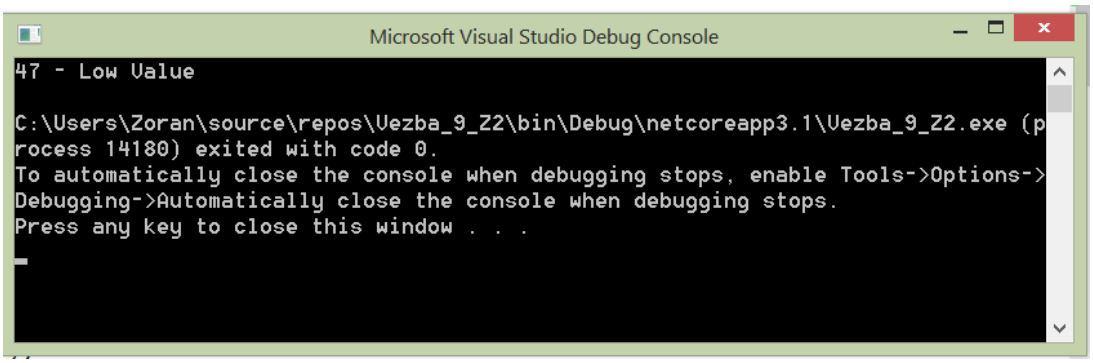
```
using System;

namespace Vezba_9_Z2
{
    class Program
    {
        delegate void myDel(int value);

        void PrintLOW(int value)
        {
            Console.WriteLine("{0} - Low Value", value);
        }

        void PrintHIGH(int value)
        {
            Console.WriteLine("{0} - High Value", value);
        }

        static void Main(string[] args)
        {
            Program program = new Program();
            myDel deleg_at;

            Random rand = new Random();
            int randomValue = rand.Next(99);

            deleg_at = randomValue < 50 ? new myDel(program.PrintLOW) : new
myDel(program.PrintHIGH);

            deleg_at(randomValue);

        }
    }
}
```

Slika 9-5. Izvorni kod programa za Zadatak 2.

NAPOMENA: Sa obzirom da se objekt kreira u funkciji generisane slučjne vrednosti potrebno je više puta starovati aplikaciju kako bi se generisali pozivi obema kapsuliranim metodama. Na slici 9-6 prikazana su oba slučaja ispisa.



a)



b)

Slika 9-6. Izgled konzolnog prozora posle a) prvog b) petog startovanja aplikacije za Zadatak 2.

Startovati 10 puta aplikaciu i zapisati dobijene rezultate u predviđenom prostoru

Prostor za prikaz konzolnog prozora

Komentarišite dobijene rezultate.

_____

_____

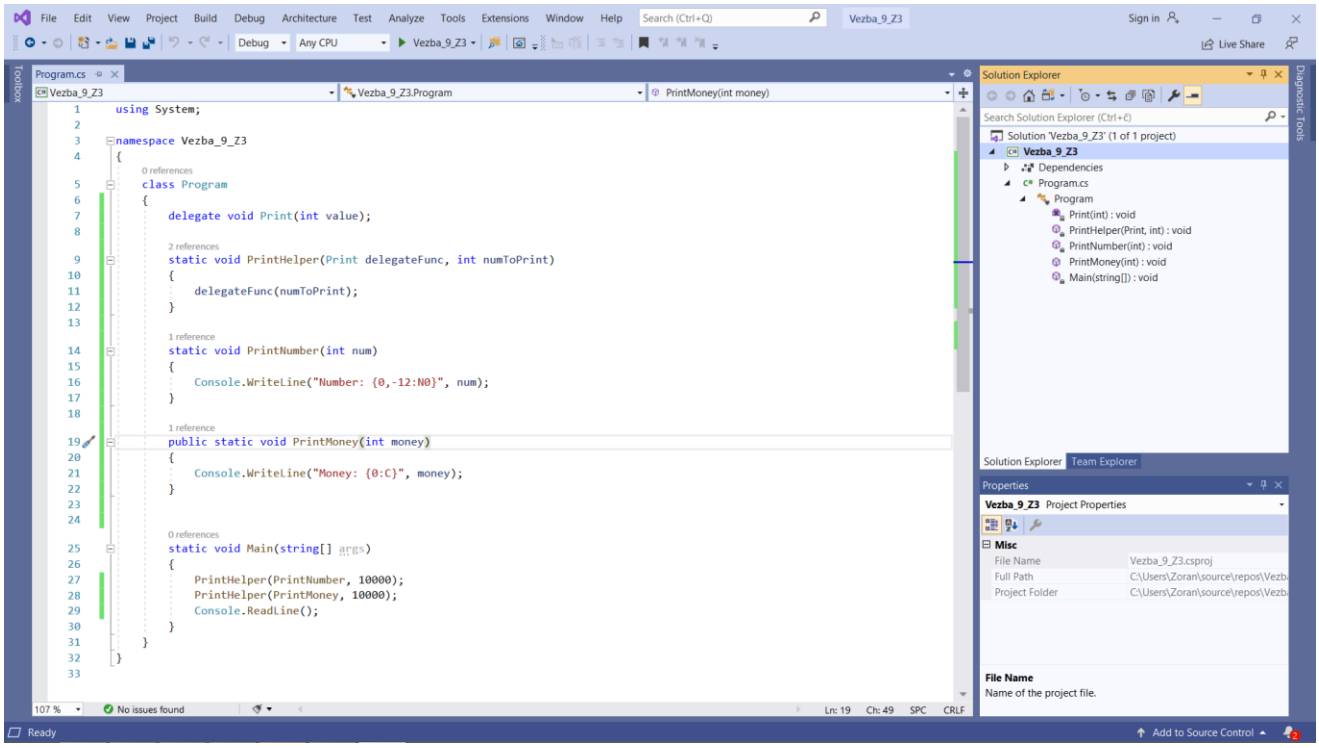
_____

Zadatak 3: Delegat kao parametar metode

Izraditi delegat `Print` koji kapsulira metode `PrintNumber` i `PrintMoney`. Delegat kapsulira ove metde koje imaju `void` povratni tip i parametar `int` tipa. Proizvoljno izabrati tip aplikacije. Potom kreirajte metodu `PrintHelper` koja za parametar ima kreirani delegat i `int` vrednost koju treba štamapti na različite - predefinisane načine.

U glavnom programu pozvati `PrintHelper` metodu sa odgovarajućim parametrima.

Na slici 9-7 prikazan je izgled realizacije ovog zadatka u VS 2019 za konzolnu aplikaciju realizovanu u .NET Core-u, dok je na slici 9-8 prikazan izvorni kod za realizaciju ovog zadatka.



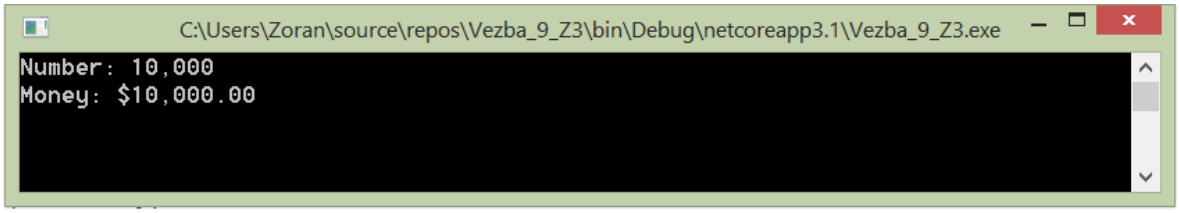
Slika 9-7. Projekat programa za Zadatak 3 u VS-u.

```
using System;

namespace Vezba_9_Z3
{
    class Program
    {
        delegate void Print(int value);

        static void PrintHelper(Print delegateFunc, int numToPrint)
        {
            delegateFunc(numToPrint);
        }

        static void PrintNumber(int num)
        {
            Console.WriteLine("Number: {0,-12:N0}", num);
        }

        public static void PrintMoney(int money)
        {
            Console.WriteLine("Money: {0:C}", money);
        }


        static void Main(string[] args)
        {
            PrintHelper(PrintNumber, 10000);
            PrintHelper(PrintMoney, 10000);
            Console.ReadLine();
        }
```

```
    }
}
```

Slika 9-8. Izvorni kod programa za Zadatak 3.

Na slici 9-9 prikazan je izgled konzolnog prozora posle startovanja ove aplikacije.



Slika 9-9. Izgled konzolnog prozora posle startovanja aplikacije za Zadatak3.

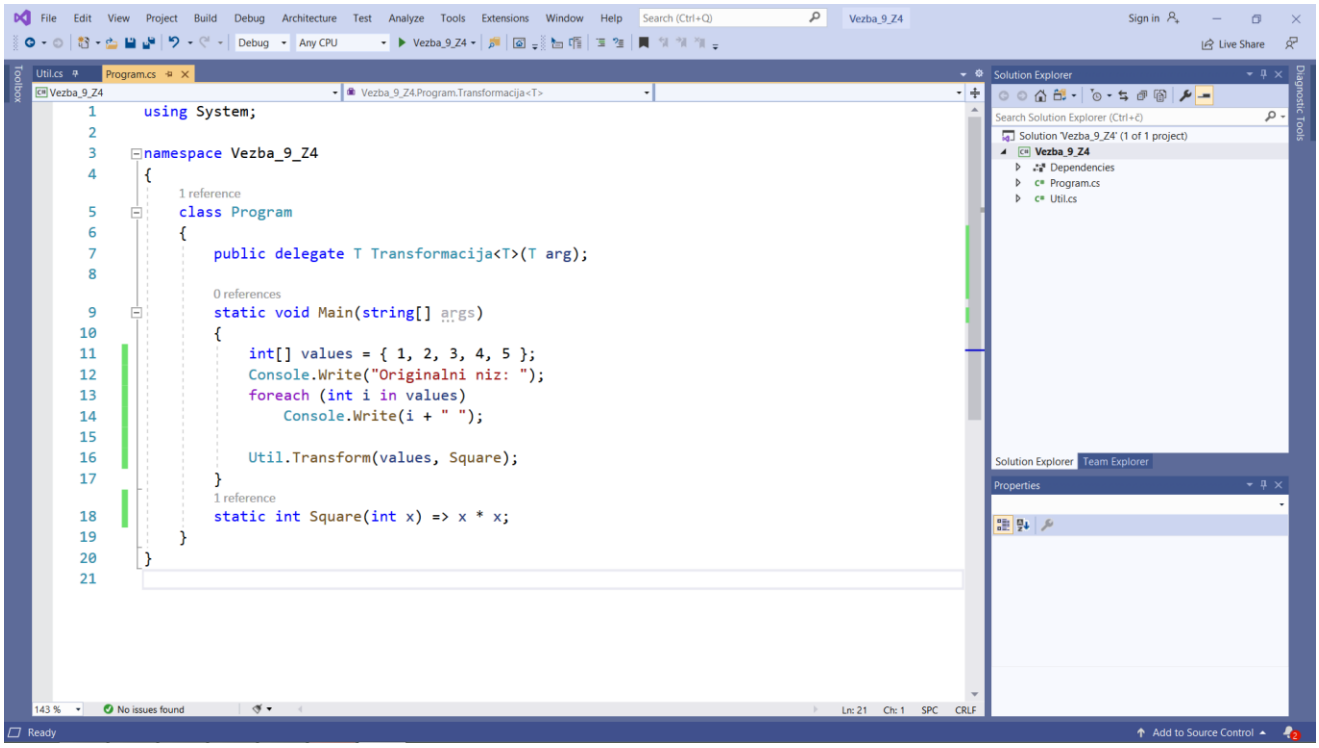
Komentarišite prikazane rezultate.

Zadatak 4: Generički delegat

Kreirati generički delegat `Transformacija<T>(T arg)` sa povratnim tipom `T` i generičkim parametrom tipa `T`. Kreirajte metodu `Square` koja zadovoljava uslove delegata i štampa transformisani niz - kvadrat vrednosti svakog člana. U glavnom programu štampati početni i transformisani niz. Na slici 9-10 prikazan je izvorni kod klase `Util` za realizaciju ovog zadatka, dok je na slici 9-11 prikazan izgled realizacije ovog zadatka u VS 2019 za konzolnu aplikaciju realizovanu u .NET Core-u.

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

namespace Vezba_9_Z4
{
    class Util
    {
        public static void Transform<T>(T[] values, Program.Transformacija<T> t)
        {
            Console.Write("\nTransformisani niz: ");

            for (int i = 0; i < values.Length; i++)
            {
                values[i] = t(values[i]);
                Console.Write(values[i] + " ");
            }

        }
    }
}
```

Slika 9-10. Izvorni kod programa klase `Util` *za Zadatak 4.*

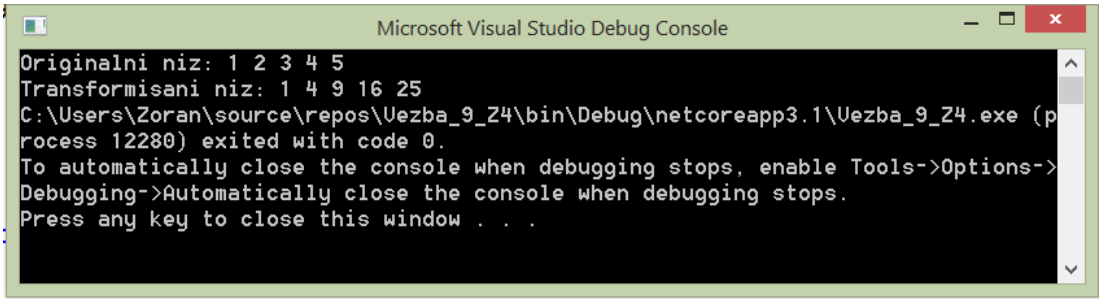


Slika 9-11. Projekat programa za Zadatak 4 u VS-u.

Na slici 9-12 prikazan je izvorni kod klase `Program`.

```
using System;

namespace Vezba_9_Z4
{
    class Program
    {
        public delegate T Transformacija<T>(T arg);

        static void Main(string[] args)
        {
            int[] values = { 1, 2, 3, 4, 5 };
            Console.Write("Originalni niz: ");
            foreach (int i in values)
                Console.Write(i + " ");

            Util.Transform(values, Square);
        }

        static int Square(int x) => x * x;

    }
}
```

Slika 9-12. Izvorni kod programa klase `Program` *za Zadatak 4.*



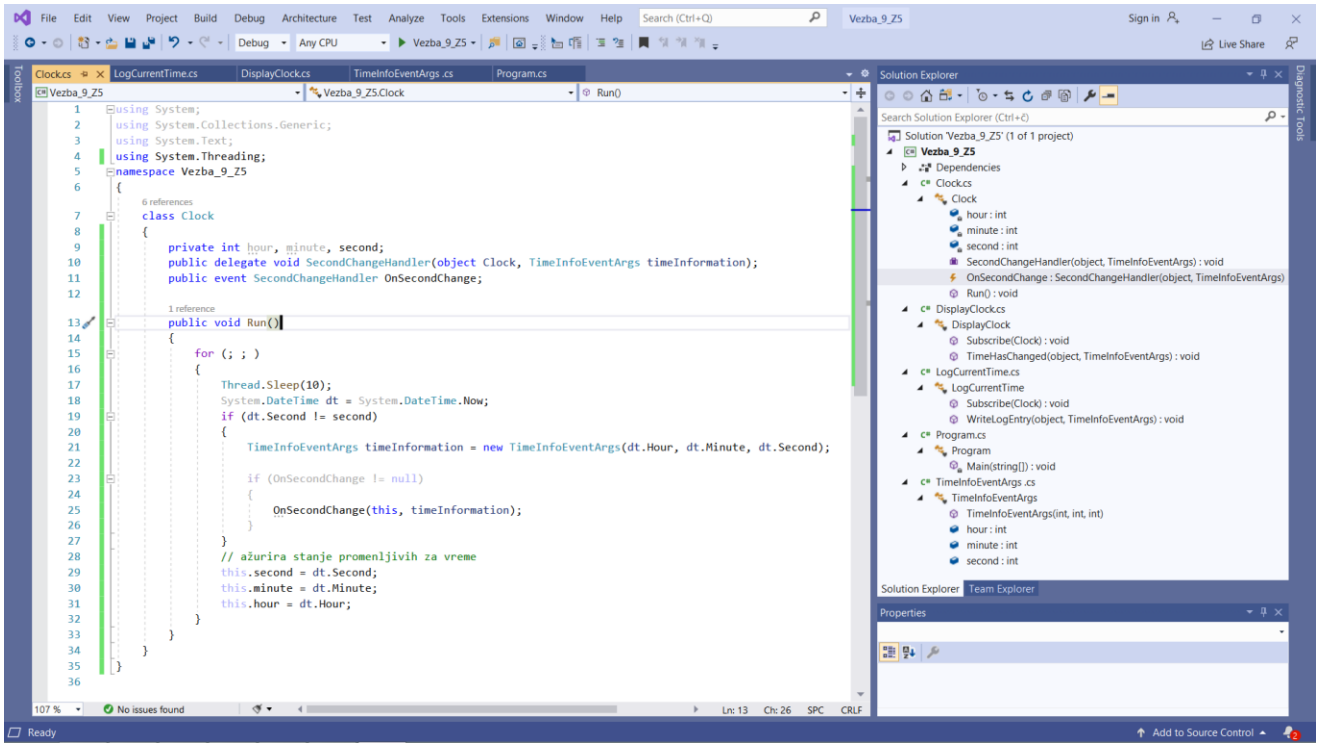
Slika 9-13. Izgled konzolnog prozora posle startovanja aplikacije za Zadatak3.

Komentarišite prikazane rezultate.

Zadatak 5: Kreiranje korisnički definisanog događaja `TimeInfoEventArgs`

Kreirati korisnički događaj `TimeInfoEventArgs` nasleđivanjem klase `EventArgs`. Kreirajte delegat `SecondChangeHandler` u klasi `Clock`.

Na slici 9-14 prikazan je izgled realizacije ovog zadatka u VS 2019 za konzolnu aplikaciju realizovanu u .NET Core-u, a izvorni kod za realizaciju ovog zadatka je dat na slici 9-15 prikazan.



Slika 9-14. Projekat programa za Zadatak 4 u VS-u.

Clock.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;
namespace Vezba_9_Z5
{
    class Clock
    {
        private int hour, minute, second;
        public delegate void SecondChangeHandler(object Clock, TimeInfoEventArgs
timeInformation);
        public event SecondChangeHandler OnSecondChange;

        public void Run()
        {
            for (; ; )
            {
                Thread.Sleep(10);
                System.DateTime dt = System.DateTime.Now;
                if (dt.Second != second)
                {
                    TimeInfoEventArgs timeInformation = new TimeInfoEventArgs(dt.Hour,
dt.Minute, dt.Second);

                    if (OnSecondChange != null)
                    {
                        OnSecondChange(this, timeInformation);
                    }
                }
                // ažurira stanje promenljivih za vreme
                this.second = dt.Second;
                this.minute = dt.Minute;
                this.hour = dt.Hour;
            }
        }
    }
}
```

TimeInfoEventArgs.cs

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

namespace Vezba_9_Z5
{
    class TimeInfoEventArgs : EventArgs
    {
        public TimeInfoEventArgs(int hour, int minute, int second)
        {
            this.hour = hour;
            this.minute = minute;
            this.second = second;
        }
        public readonly int hour;
        public readonly int minute;
        public readonly int second;
    }
}
```

Slika 9-15. Izvorni kod programa za Zadatak 5.

Zadatak 6: Kreiranje pretplatničkih klasa na korisnički događaj

Kreirati pretplatničke klase `DisplayClock` i `LogCurrentTime` koje se pretplaćuju na kreirani događaj delegatom `SecondChangeHandler`.

Na slici 10-16 prikazani su izvorni kodovi klasa za zadatak 6. Na slici 9-17 prikazan je *Solutions explorer* prozora ovog zadatka. Izgled komandnog prozora posle starovanja je prikazan na slici 9-18.

LogCurrentTime.cs

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

namespace Vezba_9_Z5
{
    class LogCurrentTime
    {
       public void Subscribe(Clock theClock)
        {
            theClock.OnSecondChange += new Clock.SecondChangeHandler(WriteLogEntry);
        }

       public void WriteLogEntry(object theClock, TimeInfoEventArgs ti)
        {
            Console.WriteLine("Logging to file: {0}:{1}:{2}",
              ti.hour.ToString(), ti.minute.ToString(), ti.second.ToString());
        }
    }
}
```

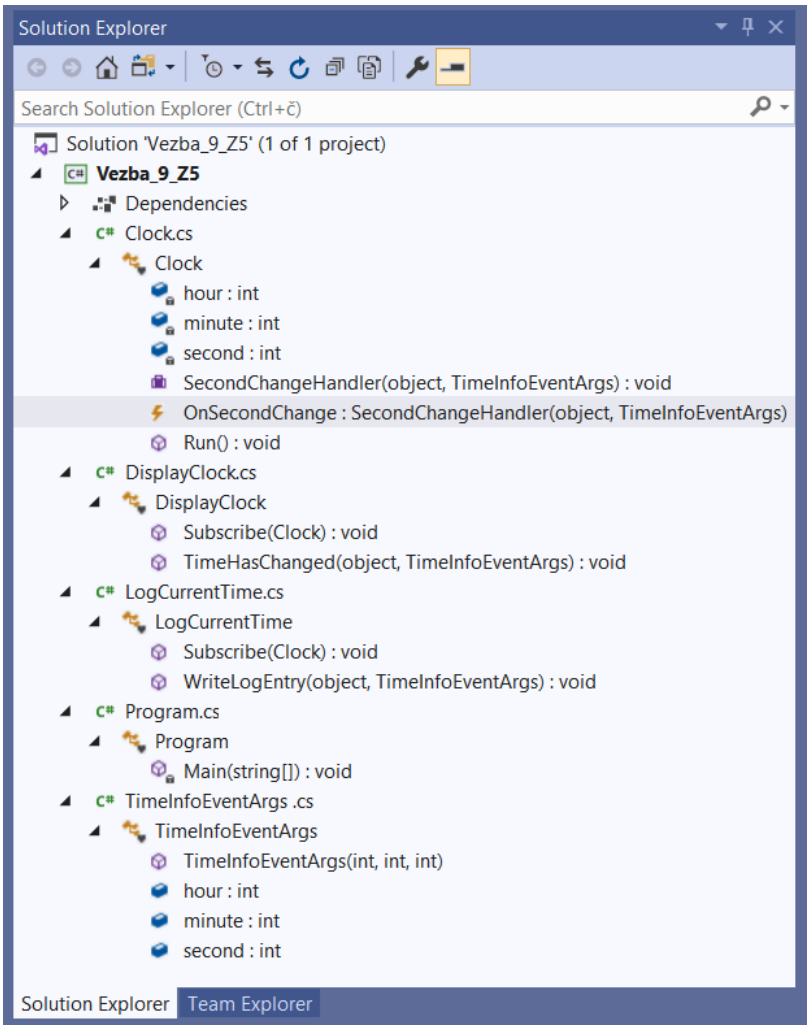
DisplayClock.cs

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

namespace Vezba_9_Z5
{
    class DisplayClock
    {
            public void Subscribe(Clock theClock)
            {
                theClock.OnSecondChange += new Clock.SecondChangeHandler(TimeHasChanged);
            }

            public void TimeHasChanged(object theClock, TimeInfoEventArgs ti)
            {
                Console.WriteLine("Current Time: {0}:{1}:{2}", ti.hour.ToString(),
ti.minute.ToString(), ti.second.ToString());
            }
        }
    }
```

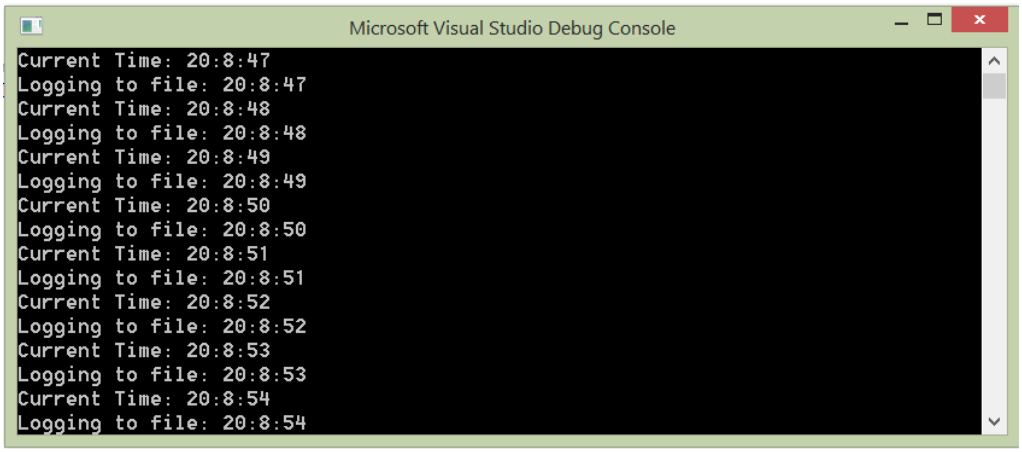
Program.cs

```
using System;

namespace Vezba_9_Z5
{

    class Program
    {
        static void Main(string[] args)
        {
            Clock theClock = new Clock();
            DisplayClock dc = new DisplayClock();
            dc.Subscribe(theClock);

            LogCurrentTime lct = new LogCurrentTime();
            lct.Subscribe(theClock);
            theClock.Run();
        }
    }
}
```

Slika 9-16. Izvorni kod programa za Zadatak 6.



Slika 9-17. Izgled Solutions explorer prozora realizovanog za Zadatak 6.



Slika 9-18. Izgled konzolnog prozora posle startovanja aplikacije za Zadatak 6.

Zadaci za samostalan rad

1. Na osnovu koda iz Zadatka 3 kreirati i kapsulirati metode koje štampaju datume i vreme na različite načine (shodno anglosaksonskoj i istočno-evropskoj notaciji).
2. Kreirati generičku klasu koja radi sa double tipovima, a izračunava kvadratni koren svakog člana niza.

U Nišu	POTVRĐUJE

LABORATORIJSKA VEŽBA 10: IZUZECI u C#

Cilj vežbe: Upoznavanje sa osnovnim pojmovima vezanim za izuzetke (engl. *exceptions*) u C#. Ovladati pojmovima „bacanja“ i „hvatanja“ izuzetaka kao i programskom arhitekturom za njihovu primenu. Proučiti izuzetke najznačajnijih klasa i način primene. Upoznati se sa primenom višestruke „`catch`“strukture.

Zadatak 1: Štampanje sadržaj tekstualnog fajla na komandnoj liniji

Koristeći tekst editor - Notepad kreirati tekstualni fajl `Primer.txt` i smestiti ga u proizvoljni folder. Kreirati konzolnu aplikaciju u .NET Core-u sa programskim kodom za čitanje i štampanje sadržaja tekstualnog fajla. Nazovite ovu aplikaciju Vezba_10_Z1. Na slici 10-1 prikazan je programski kod ovog zadatka.

```
using System;
using System.IO;
using System.Text;


namespace Vezba_10_Z1
{
    class Program
    {
        static void Main(string[] args)
        {
            string content =
File.ReadAllText(@"D:\PREDAVANJA\PREDAVANJA_ATVSS_2020_2021\NET_tehnologije\Praktikum_NET_2020
\Primer.txt");

       Console.WriteLine(content);
        }
    }
}
```

Slika 10-1. Programski kod za realizaciju Zadatka 1.

`Primer.txt`

```
Blago onom ko dovjeka živi imao se rašta i roditi.

P.P. Njegoš
```

Slika 10-2. Tekstualni sadržaj fajla `Primer.txt`.

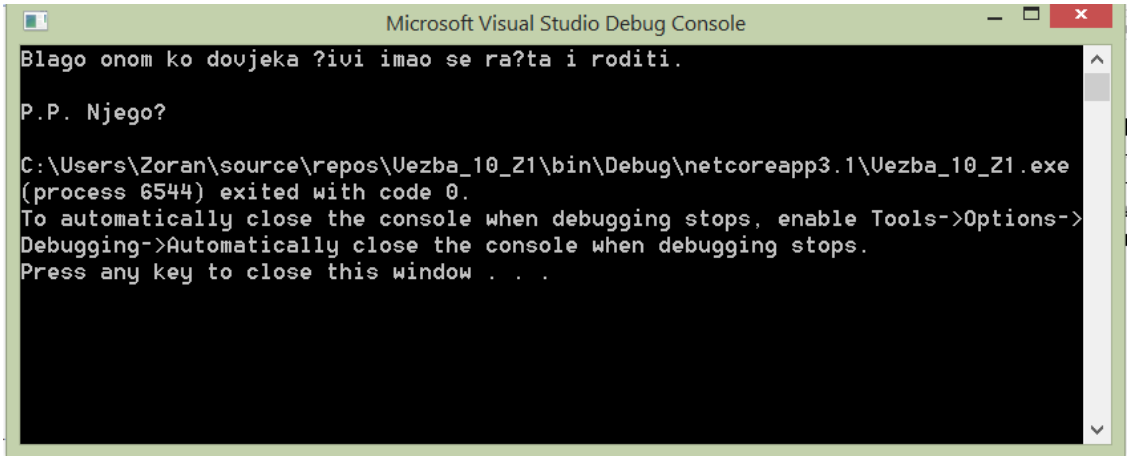
Komentarisati programski kod i korišćene klase sa slike 10-1.

_____

_____

_____

Stratovati realizovani projekt u okruženju VS-a. Na slici 10-3 je prikazan izgled komandnog prozora posle startovanja.



Slika 10-3. Tekstualni sadržaj fajla `Primer.txt`.

Komentarisati prikaz konzolnog prozora sa slike 10-3.

_____

_____

_____

Zadatak 2: Bacanje izuzetka pri pokušaju čitanja nepostojećeg fajla

Preimenujte fajl `Primer.txt` kreiran u Zadatku 1 na `Primer_1.txt` . Sada stratujte izvršenje projekta u okruženju VS-a kreiranog u Zadatku 1. Pažljivo pogledajte odziv VS-a.

Na slici 10-4 prikazan je prozor koji obaveštava korisnika da se desio izuzetak koji nije obrađen. Prikazan je red u izvornom programskom kodu koji je izazvao izuzetak.

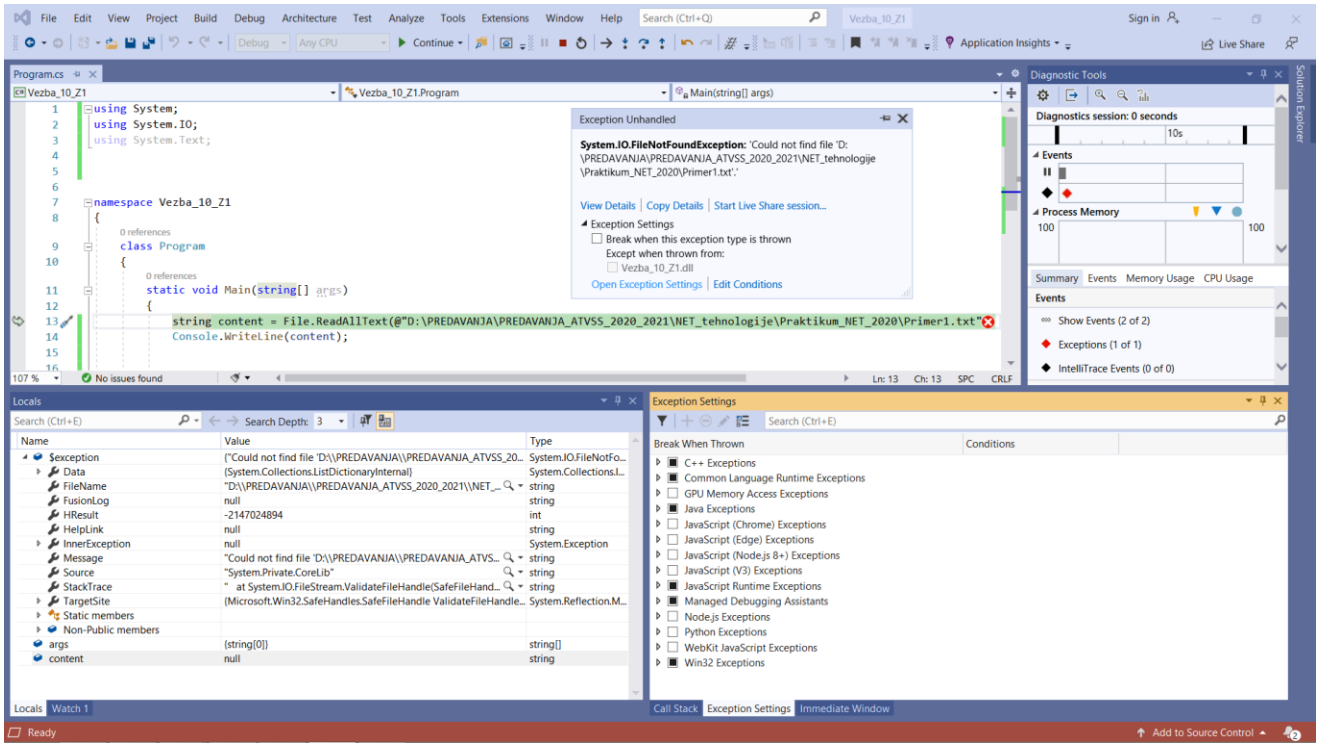
Dodatne informacije o bačenom izuzetku se mogu dobiti na linkovima u okviru ove stranice.

Komentarisati prikaz prozora sa slike 10-4.

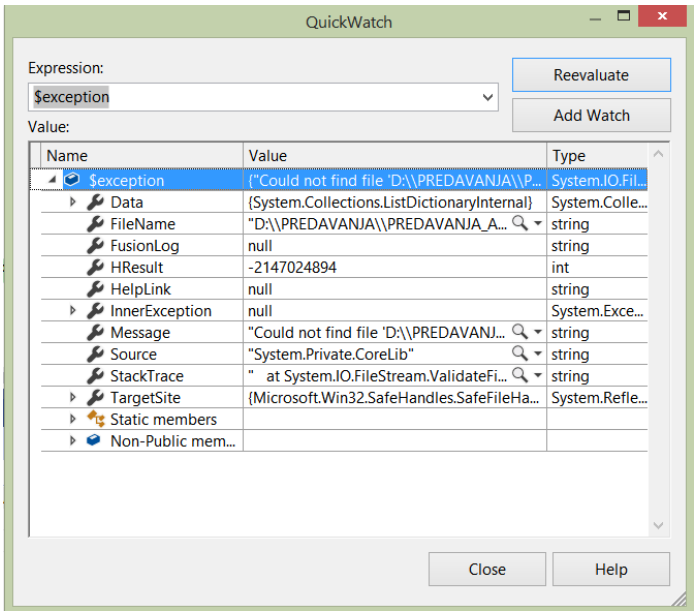
_____

_____

_____



Slika 10-4. VS prozor koji se otvara prilikom pojave neobrađenog izuzetka.



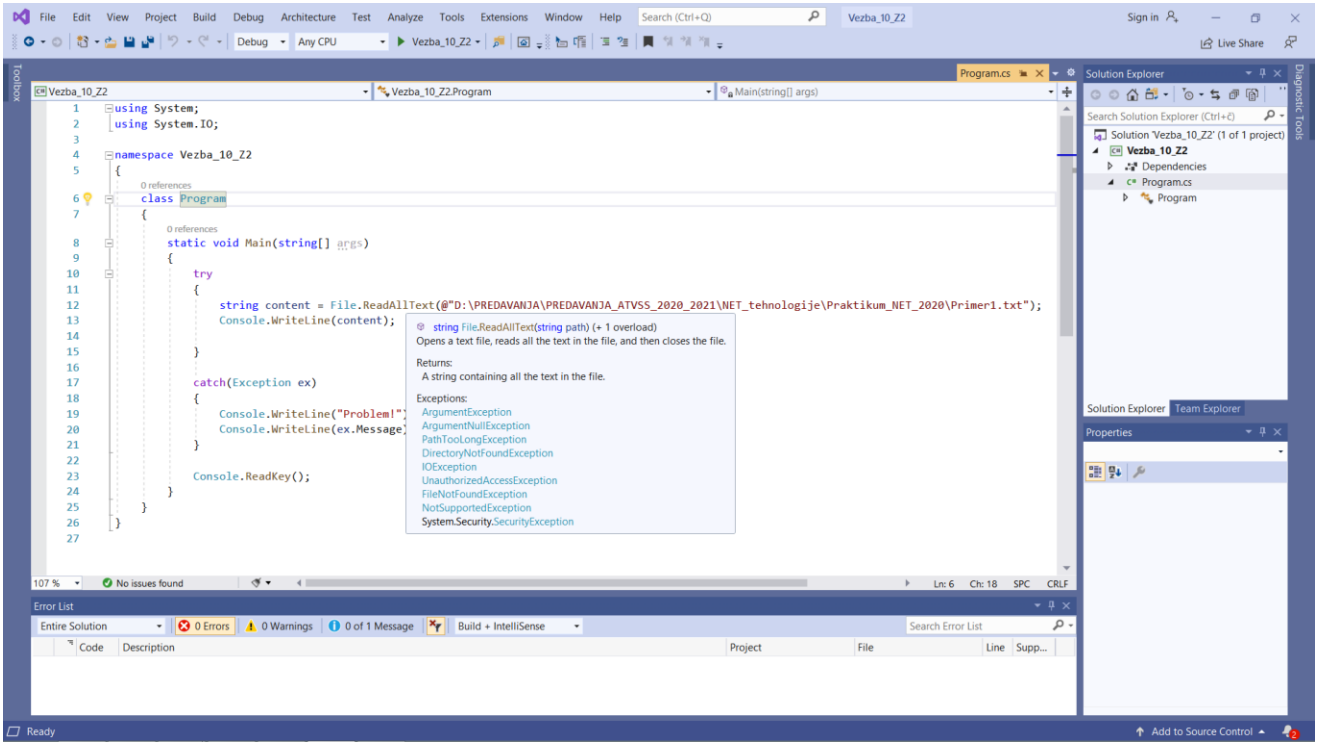
Slika 10-5. VS prozor sa dodatnim informacijama o bačenom izuzetku.

Nabrojte svojstva bačenog izuzetka sa slike 10-5.

_____

_____

Na slikama 10-6 i 10-7 prikazani su pomoćni prozori VS-a vezani za izuzetke koje može baciti metoda `ReadAllText` klase `File`.



Slika 10-6. Alat Intellisense prikazuje preklopljene metode i izuzetke koje može generisati metoda `ReadAllText`.

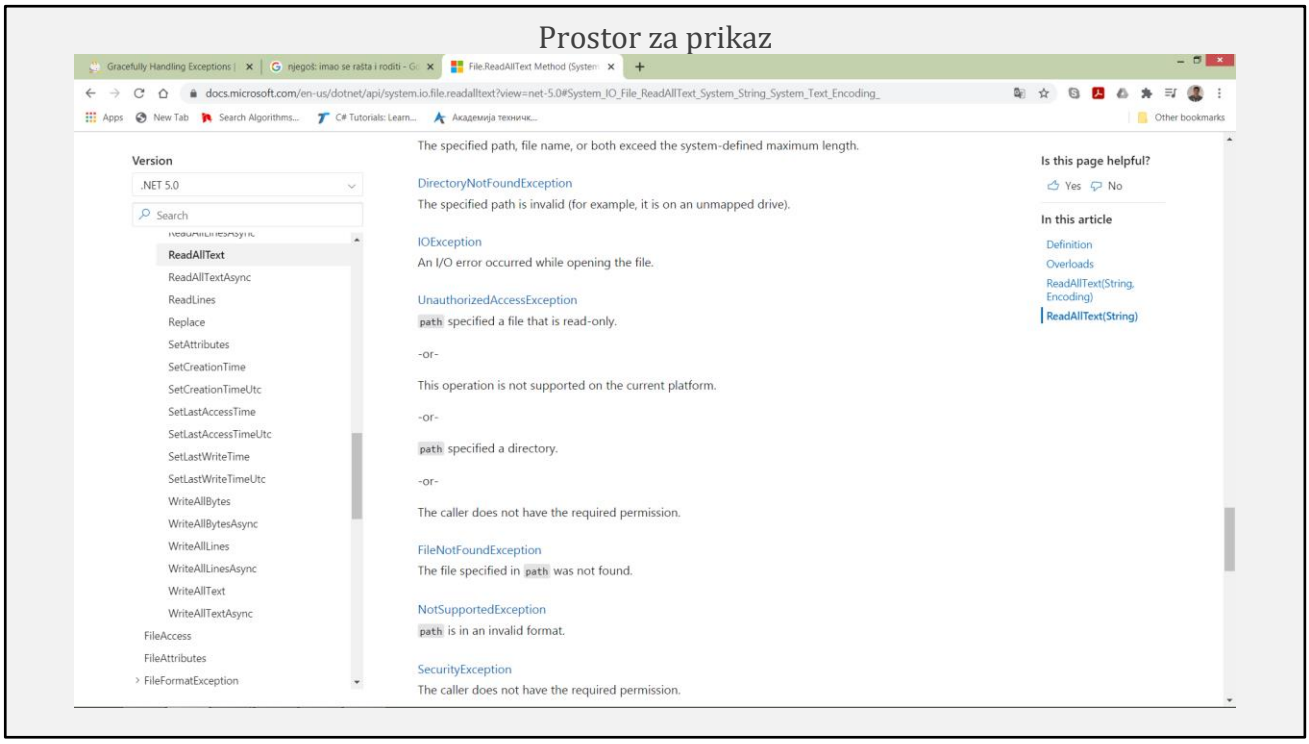
Nabrojte sve izuzetke koje može baciti metoda `ReadAllText`.

_____

_____

_____

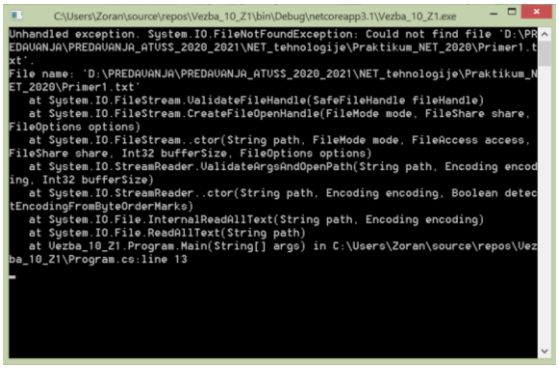
Pronađite na MS sajtu informacije vezane za metoda `ReadAllText`. Posebno prikazati izuzetke vezane za ovu metodu.



Slika 10-7. Prozor koji se otvara prilikom stratovanja aplikacije izvan VS-a .

Zadatak 3: Stratovanje programa iz korisničkog okruženja

Stratujte kreirani program izvan VS-a. Da bi se to realizovalo, postavite se u `bin` folder, pronađite izvršnu verziju i stratujte. Na slici 10-5 prikazan je komandni prozor dobijen posle stratovanja aplikacije kao i kompletan tekst ovog prozora.




```
System.IO.FileNotFoundException
  HResult=0x80070002
  Message=Could not find file
'D:\PREDAVANJA\PREDAVANJA_ATVSS_2020_2021\NET_tehnologij
e\Praktikum_NET_2020\Primer1.txt'.
  Source=System.Private.CoreLib
  StackTrace:
   at
System.IO.FileStream.ValidateFileHandle(SafeFileHandle
fileHandle)
   at System.IO.FileStream.CreateFileOpenHandle(FileMode
mode, FileShare share, FileOptions options)
   at System.IO.FileStream..ctor(String path, FileMode
mode, FileAccess access, FileShare share, Int32
bufferSize, FileOptions options)
   at
System.IO.StreamReader.ValidateArgsAndOpenPath(String
path, Encoding encoding, Int32 bufferSize)
   at System.IO.StreamReader..ctor(String path, Encoding
encoding, Boolean detectEncodingFromByteOrderMarks)
   at System.IO.File.InternalReadAllText(String path,
Encoding encoding)
   at System.IO.File.ReadAllText(String path)
   at Vezba_10_Z1.Program.Main(String[] args) in
C:\Users\Zoran\source\repos\Vezba_10_Z1\Program.cs:line
13
```

Slika 10-8. Prozor koji se otvara prilikom stratovanja aplikacije izvan VS-a .

Komentarisati prikaz prozora sa slike 10-8.

_____

_____

_____

Zadatak 4: Hvatanje i obrada izuzetka

Da bi se sprečilo nasilno prekidanje izvršenja, neophodno je hvatanje i opsluživanje izuzetka. Programski kod koji rešava problem i zadatka 1 i 2 je prikazan na slici 10-6 . Korigujte programski kod glavnog programa dodavanjem `try-catch` strukture kako je to prikazano na slici 10-9. Potencijalno opasni programski kod (kod koji može baciti izuzetak) se stavlja u `try` blok, dok `catch` blok sadrži programski kod za obradu izuzetka i sprečava nasilno završavanje programa.

Komentarisati programski kod sa slike 10-9.

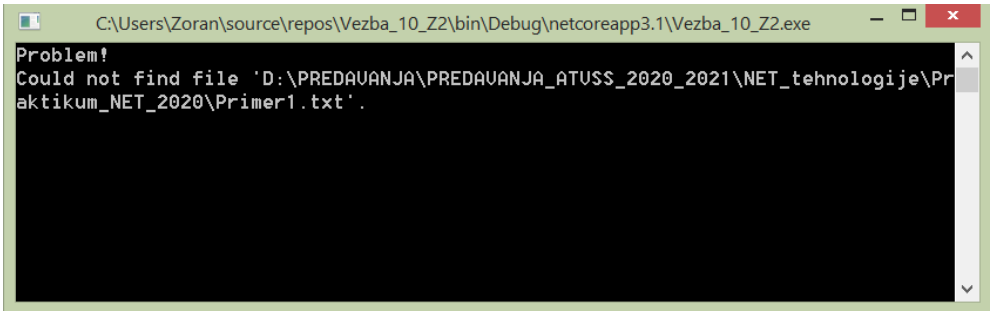
_____

_____

```
using System;
using System.IO;

namespace Vezba_10_Z2
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                string content =
File.ReadAllText(@"D:\PREDAVANJA\PREDAVANJA_ATVSS_2020_2021\NET_tehnologije\Praktikum_NET_2020
\Primer1.txt");
                Console.WriteLine(content);

            }

            catch(Exception ex)
            {
                Console.WriteLine("Problem!");
                Console.WriteLine(ex.Message);
            }

            Console.ReadKey();
        }
    }
}
```

Slika 10-9. Programski kod za hvatanje i obradu izuzetka zadatka1 i 2.

Na slici 10-10 prikazan je prozor aplikacije posle stratovanja korigovane aplikacije.



Slika 10-10. Programski kod za hvatanje i obradu izuzetka zadatka1 i 2.

Komentarisati prikaz sa slike 10-10.

Zadatak 5: Hvatanje i obrada više izuzetka

Korigovati programski kod iz zadatka 4 da može obraditi više različitih tipova izuzetaka. Programski kod koji treba uneti u glavni program je prikazan na slici 10-11.

```
using System;
using System.IO;
using System.Text;

namespace Vezba_10_Z2
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                string content =
File.ReadAllText(@"D:\PREDAVANJA\PREDAVANJA_ATVSS_2020_2021\NET_tehnologije\Praktikum_NET_2020
\Primer1.txt", Encoding.UTF8);
                Console.WriteLine(content);

            }
            catch(FileNotFoundException ex)
            {
                Console.WriteLine("Problem!");
                Console.WriteLine("Fajl ne postoji");
            }
            catch(DirectoryNotFoundException ex)
            {
                Console.WriteLine("Problem!");
                Console.WriteLine("Direktorijum ne postoji");
            }
            catch(Exception ex)
            {
```

```
                Console.WriteLine("Problem!");
                Console.WriteLine(ex.Message);
            }
            Console.ReadKey();
        }
    }
}
```

Slika 10-11. Korigovan programski kod za hvatanje i obradu više izuzetka.

Komentarisati prikaz sa slike 10-11.

_____

_____

_____

_____

Dodati blok `finally` koji treba da zatvori već otvoren fajl za čitanje. Struktura bloka je prikazana na slici 10-12.

```
finally
{

    // ovde zatvoriti fajl

        ...
}
```

Slika 10-11. Blok `finally`.

Traženi programski kod prikazati u predviđenom prostoru.

Prostor za prikaz

Zadatak 6: Kreiranje korisničkog izuzetka `MyCustomException`

Kreirati korisnički izuzetak `MyCustomException` koji će prilikom deljenja baciti izuzetak kada je BROILAC jednak 0. U glavnom programu realizovati standarne slučajeve deljenja nulom, korisniči bačen izuzetak, kao i bilo koji drugi izuzetak. Takođe, kreirati `finally` blok koji treba da zatvori otvoreni fajl koji se otvara u na početku metode.

U projektu kreirati klasu `Test` sa metodom `TestFunc()` koja će bacati odgovarajuće izuzetke koje treba uhvatiti i obraditi. U ovoj metodi kreirajte dve double promenljive a i b kojima treba zadavati odgovarajuće vrednosti kako bi se izazvalo bacanje svih izuzetaka koji se hvataju.

Na slici 10-12 prikazan je glavni programski kod aplikacije kao i deklaracija korisničkog izizetka `MyCustomException`.

```
using System;

namespace Vezba_10_Z6
{
    class Program
    {
        public class MyCustomException: System.ApplicationException
        {
            public MyCustomException(string message) : base(message)
            {
                ;
            }

        }

        static void Main(string[] args)
        {
            Test t = new Test();
            t.TestFunc();
        }
    }
}
```

Slika 10-12. Glavni program aplikacije i dekleracija korisničkog izizetka.

Korisnički izuzetak se kreirana nasleđivanjem klase _____.

Na slici 10-13 prikazan je programski kod klase `Test` i metode `TestFunc()`. Programski kod sa slike 10-12 i 10-13 uneti u konzolni (Core) projekat kreiran na ranije prikazan način.

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

namespace Vezba_10_Z6
{
    class Test
    {
        public void TestFunc()
        {
            try
            {
```

```
                Console.WriteLine("Otvaranje fajla na ovom mestu");
                double a = 5;
                double b = 0;
                Console.WriteLine("{0} / {1} = {2}", a, b, DoDivide(a,b));
            }
            catch (System.DivideByZeroException e)
            {
                Console.WriteLine("\n Deljenje nulom - izuzetak! Poruka: {0}", e.Message);
                Console.WriteLine("\nHelpLink: {0}\n", e.HelpLink);
            }
            catch(Program.MyCustomException e)
            {
                Console.WriteLine("\n Korisnički izuzetak! Poruka: {0}", e.Message);
                Console.WriteLine("\nHelpLink: {0}\n", e.HelpLink);
            }
            catch
            {
                Console.WriteLine("Nepoznati izvor izuzetka");
            }
            finally
            {
                Console.WriteLine("Zatvori fajl ovde");
            }
        }

        public double DoDivide(double a, double b)
        {
            if (b == 0)
            {
                DivideByZeroException e = new DivideByZeroException();
                e.HelpLink = "http://vtsnis.edu.rs";
                throw e;
            }

            if(a == 0)
            {
                Program.MyCustomException e = new Program.MyCustomException("U ovom primeru
imenilac ne može biti nula!");
                e.HelpLink = "http://vtsnis.edu.rs";
                throw e;
            }
            return a / b;
        }
    }
}
```

Slika 10-12. Programski kod klase `Test`.

U predviđenom prostoru unesite tekst iz komandnog prozora posle stratovanja aplikacije.

Prostor za prikaz

Izmenom vrednosti parametara a i b u metodi `TestFunc()` odgovarajuće izuzetke.
Vrednosti parametara a = _____ i b = _____ izazivaju bacanje korisničkog izuzetka `MyCustomException`.

U predviđenom prostoru unesite tekst iz komandnog prozora posle stratovanja aplikacije.

Prostor za prikaz

zmenom vrednosti parametara a i b u metodi `TestFunc()` odgovarajuće izuzetke.
Vrednosti parametara.

Zadaci za samostalan rad

1. Kreirati sopstveni izuzetak `ObaCiniocaNula_Exception` koji će baciti izuzetak kada su i imenilac i brojilac jednaki 0.
2. Kreirati programski kod koji testira izuzetak `IndexOutOfRangeException`.

U Nišu	POTVRĐUJE