ПРОГРАМИРАЊЕ – ПРОГРАМСКИ ЈЕЗИК C#

У следећим задацима заокружите број испред траженог одговора

71.	Дати су типови промењљивих у програмском језику C#. Одредити како се назива променљива која је дефинисана унутар неког метода. Заокружити број испред очекиваног одговора: 1. Глобална променљива 2. Статичка променљива 3. Блоковска променљива 4. Локална променљива	**1**
72.	Одредити какви могу бити чланови класе (поља и методе) у програмском језику C#. Заокружити број испред очекиваног одговора: 1. Локални и глобални 2. Процедурални и непроцедурални 3. Статички (класни) и нестатички (објектни) 4. Спољашњи и унутрашњи	**1**
73.	Одредити која поља су заједничка и јединствена за све креиране објекте неке класе дефинисане у објектно оријентисаном програмском језику C#. Заокружити број испред очекиваног одговора: 1. Јавна 2. Приватна 3. Објектна 4. Инстанцна 5. Статичка	**1**
74.	У програмском језику C# класа може да садржи статичка и не-статичка (инстанцна) поља. Дате су изјаве које се односе на статичка поља класе и међу њих је уметнута једна изјава која се односи на не-статичка (инстанцна) поља класе. Заокружити број испред изјаве која се односи на не-статичка поља класе: 1. Поље које се може се користити без конструисања иједног објекта те класе 2. Поље које има исту вредност за све креиране објекте неке класе 3. Поље чија се вредност може разликовати за сваки појединачни објекат неке класе 4. Поље које се може користити унутар статичких метода класе, као и унутар метода инстанце	**1**
75.	Заокружити број испред исказа који представља исправан наставак дате тврдње: При креирању објеката изведене класе... 1. извршава се само конструктор изведене класе 2. прво се извршава конструктор родитељске класе, али само ако је позван кључном речју ***base*** 3. обавезно се прво извршава конструктор изведене, а потом конструктор родитељске класе 4. обавезно се прво извршава конструктор родитељске, а потом конструктор изведене класе	**1**

76. У програмском језику C# користи се службена реч ***base***. Проценити који од наредних исказа који дефинишу дату службену реч **НИЈЕ** тачан.

Заокружити број испред очекиваног одговора:

1. Службена реч ***base*** може послужити за позивање конструктора родитељске класе.
2. Службена реч ***base*** може послужити за позивање приватних метода родитељске класе којима се другачије не може приступити.
3. Службена реч ***base*** може послужити за позивање заклоњеног метода родитељске класе.
4. Службена реч ***base*** може послужити за позивање заклоњеног поља родитељске класе.

1

77. Дат је код програма у програмском језику C#:

```
namespace TestPrimer {
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine(fun(17));
        }
        public int fun(int n) { return n; }
        public void fun(int n){ Console.WriteLine(n); }
    }
}
```

Анализирати код и заокружити број испред очекиваног одговора:

1. Програм има грешку, јер се не може одредити коју верзију преоптерећеног метода fun(…) треба позвати.
2. Програм има грешку, јер је друга верзија преоптерећеног метода fun(…) дефинисана али се нигде не позива.
3. Програм се нормално извршава и приказује 17 једанпут.
4. Програм се нормално извршава и приказује 17 двапут.

2

78. Дат је код програма у програмском језику C# који формира и штампа елементе низа **a**. Анализирати дати код и проценити шта ће се догодити након његовог извршавања.

```
namespace TestPrimer {
    class Program {
        static void Main(string[] args){
            int[] a = new int[5];
            for (int i = 0; i < a.Length; i++) a[i] = i;
            Console.Write(a[i] + " ");
        }
    }
}
```

Заокружити број испред очекиваног одговора:

1. Програм приказује бројеве 0 1 2 3 4 на екрану.
2. Програм има грешку, јер ће у последњој наредби Console.**Write** метода **Main** покушати приступ непостојећем елементу a[5].
3. Програм приказује број 5 на екрану.
4. Програм има грешку, јер променљива ***i*** у последњој наредби Console.Write у методу Main неће имати дефинисану вредност.

2

79. У програмском језику C# дата је декларација низа:

int к;
int[ ] brojevi = {5, 12, 37, 7, 27, 33, 36};

На основу дате декларације одредити шта је резултат позива
k=Arrays.BinarySearch(brojevi, 37);

Заокружити број испред очекиваног одговора:

1. k=0, јер метод BinarySearch прво изврши сортирање низа у опадајућем редоследу, па онда тражи задату вредност
2. метод BinarySearch баца изузетак увек када је низ неуређен и програм „пуца“
3. k=2, јер се тражени елеменат налази на позицији 2
4. k добија неочекивану вредност јер низ мора бити сортиран у растућем поретку пре позива методе BinarySearch
5. k=6, јер метод BinarySearch прво изврши сортирање низа у растућем редоследу, па онда тражи задату вредност

2

80. У програмском језику C# дата је декларација једне стринг и једне целобројне променљиве, као и део кода:

```
string str = "Primer";
int broj = 66;
Console.WriteLine(str + broj + 65);
Console.WriteLine(broj + 65 + str);
```

Анализирати код и проценити шта ће се приказати на екрану након његовог извршења.
Заокружити број испред очекиваног одговора:

1. Primer6665
131Primer
2. Primer6665
6665Primer
3. Primer131
131Primer
4. PrimerBA
BAPrimer

2

81. Дат је код у програмском језику C#, који дефинише рекурзивни метод. Анализирати код и одредити резултат извршавања задатог метода.

```
public long fun(int n){
            return n * fun(n - 1);
}
```

Заокружити број испред очекиваног одговора:

1. Резултат позива fun(3) је 1.
2. Резултат позива fun(3) је 2.
3. Резултат позива fun(3) је 6.
4. Позив fun(3) изазива грешку јер производи бесконачан ланац позива истог метода fun(…).

2

82. Дат је код у програмском језику C#, који дефинише рекурзивни метод. Анализирати код и одредити резултат који ће се приказати на екрану.

```
namespace TestPrimer {
    class Program {
        static void Main(string[] args) {
            fun(2);
        }
        public static void fun(int n) {
            while (n > 1) {
                Console.Write((n - 1) + " ");
                fun(n - 1);
            }
        }
    }
}
```

Заокружити број испред очекиваног одговора:

1. Програм на екрану не приказује ништа
2. Програм на екрану приказује 1 2 3
3. Програм на екрану приказује 3 2 1.
4. Програм на екрану бесконачно приказује 1 1 1 1 1 ….
5. Програм на екрану бесконачно приказује 2 2 2 2 2 ….

2

83. У програмском језику C# дат је рекурзивни метод који проверава да ли је неки стринг палиндром. Да би код био комплетиран потребно је допунити трећи ред условом **if** наредбе.

```
1. public static bool palindrom(String s)
2. {
3.     if (s.Length <= 1) return true;  //bazni slučaj
4.     else if (____________________) return false;
5.     else return palindrom(s.Substring(1, s.Length - 2));
6. }
```

Заокружити број испред очекиваног одговора:

1. `s[0] != s[s.Length - 1]`
2. `s[0] != s[s.Length]`
3. `s[1] != s[s.Length - 1]`
4. `s[1] != s[s.Length]`

2

84. У програмском језику C# дат је рекурзивни метод који проверава да ли је неки string палиндром. Да би код био комплетиран потребно је допунити седми ред.

```
1. public static bool Palindrom(String s){
2.     return Palindrom(s, 0, s.Length - 1);
3. }
4. public static bool Palindrom(String s, int levi, int desni){
5.     if (desni <= levi) return true; // bazni slucaj
6.     else if (s[levi] != s[desni]) return false;
7.     else return _____________________;
8. }
```

Заокружити број испред очекиваног одговора:

1. `Palindrom(s)`
2. `Palindrom(s, levi, desni)`
3. `Palindrom(s, levi + 1, desni - 1)`
4. `Palindrom(s, levi + 1, desni)`
5. `Palindrom(s, levi, desni - 1)`

2

85. У програмском језику C# дат је рекурзивни метод за бинарно претраживање сортираног целобројног низа. Да би код био комплетиран потребно је допунити девети ред (означен линијом) помоћу понуђеног одговора.

```
1.  public static int TraziBroj(int[] niz, int broj) {
2.      return TraziBroj(niz, broj, 0, niz.Length - 1);
3.  }
4.  public static int TraziBroj(int[] niz,int broj,int levi,int desni) {
5.    if(levi > desni) return -1; // broj nije nadjen u nizu
6.    int sredina = (levi + desni) / 2;
7.    if(broj < niz[sredina]) return TraziBroj(niz, broj, levi,sredina-1);
8.    else if(broj > niz[sredina]) return _____________________________;
9.    else return sredina;
10. }
```

Заокружити број испред очекиваног одговора:

```
1. TraziBroj(niz, broj, sredina + 1, levi)
2. TraziBroj(niz, broj, sredina - 1, levi)
3. TraziBroj(niz, broj, desni, sredina + 1)
4. TraziBroj(niz, broj, sredina + 1, desni)
```

2

86. Дат је код програма у програмском језику C#. Анализирати дати код и проценити његову тачност. Заокружити број испред понуђеног тачног исказа:

```
namespace TestPrimer {
    class Test {
        int x;
        public Test(string s){
            Console.WriteLine("Klasa Test");
        }
        static void Main(string[] args){
            Test t = null;
            Console.WriteLine(t.x);
        }
    }
}
```

1. Програм има грешку јер променљива x није иницијализована.
2. Програм има грешку јер класа Test нема подразумевани конструктор.
3. Програм има грешку јер се у некој класи не може декларисати променљива типа те исте класе, као што је то овде случај са променљивом t.
4. Програм има грешку јер променљива t није иницијализована и има вредност ***null*** у моменту када се приказује поље ***t.x*** .
5. Програм нема грешака и нормално се извршава, не приказујући ништа на екрану.

2

87. Дата је дефиниција класе у програмском језику C#. Проценити где у дефиницији класе (испред које методе) треба заменити знакове ***???*** службеном речју ***static***.

```
1.  public class Test {
2.      private int broj;
3.
4.      public ??? int kvadrant(int n) { return n * n; }
5.      public ??? int getBroj() { return broj; }
6.  }
```

Заокружити број испред тачне изјаве:

1. Метода ***kvadrant*** МОРА да буде статичка, док метода ***getBroj*** може и не мора.
2. Обе методе морају бити статичке.
3. Ни једна од дефинисаних метода није статичка.
4. Метода ***getBroj*** НЕ СМЕ да буде статичка, док метода ***kvadrant*** може и не мора.

2

88. Дата је дефиниција класе у програмском језику C# и састоји се од два конструктора, методе и поља x и y. У шестом реду написати конструктор копије објекта класе Point.

```
1.  public class Point {
2.      private double x, y;
3.      public Point() { x = 0; y = 0; }
4.      public void Set(double xx, double yy){ x=xx; y=yy; }
5.      public Point(Point p) {
6.          _____________________//Odgovor
7.      }
8.  }
```

2

Заокружити број испред очекиваног одговора:

1. `this(p.x, p.y);`
2. `this(p);`
3. `Set(p);`
4. `Set(p.x, p.y);`

89. Дат је код програма у програмском језику C# којим су дефинисане две класе: `class Program` која садржи `Main(string[] args)` методу и `class KlasaA`. Анализирати дати код и одредити да ли је код исправно написан. Понуђени одговори дају опис последица извршавања овог кода. Заокружити број испред тачног исказа.

```
class Program {
    public static void Main(string[] args){
        KlasaA a1 = new KlasaA ();
        KlasaA a2 = new KlasaA ();
        Console.WriteLine(a1.Equals(a2));
    }
}
class KlasaA {
    int x;
    public bool Equals(KlasaA a){
        return this.x == a.x;
    }
}
```

2

1. Програм има грешку, јер се изразом a1.Equals(a2) проверава једнакост објеката a1 и a2 различитог типа од Object.
2. Програм има грешку, јер се једнакост објеката a1 и a2 типа KlasaA проверава изразом a1 == a2.
3. Програм се извршава без грешке и приказује се true на екрану.
4. Програм се извршава без грешке и приказује се false на екрану.

90. Дат је код програма у програмском језику C# којим су дефинисане две класе: `class Program` која садржи `Main(string[] args)` методу и `class KlasaA`. Анализирати дати код и одредити да ли је код исправно написан. Понуђени одговори дају опис последица извршавања овог кода. Заокружити број испред тачног исказа.

```
class Program {
     public static void Main(string[] args){
            Object a1 = new KlasaA();
            Object a2 = new KlasaA();
            Console.WriteLine(a1.Equals(a2));
     }
}
class KlasaA {
     int x;
     public bool Equals(KlasaA a){
            return this.x == a.x;
     }
}
```

2

1. Програм има грешку, јер се изразом a1.Equals(a2) проверава једнакост објеката a1 и a2 различитог типа од Object.
2. Програм има грешку, јер се једнакост објеката a1 и a2 типа KlasaA проверава изразом a1 == a2.
3. Програм се извршава без грешке и приказује се true на екрану.
4. Програм се извршава без грешке и приказује се false на екрану.

91. Дат је код програма у програмском језику C# у ком су дефинисане три класе: class Program која садржи Main(string[] args) методу, class A и class B. Анализирати дати код и одредити да ли је код исправно написан.
Заокружити број испред исказа који даје информацију о тачности кода.

```
class Program {
  public static void Main(string[] args) {
    B b = new B();
    b.Metod(5);
    Console.WriteLine("b.i je " + b.CitajI());
  }
}
class A {
   int i;
   public int CitajI(){return i;}
   public void Metod(int i) { this.i = i; }
}
class B : A {
   public void Metod(string s){
   Console.WriteLine(s);
  }
}
```

2

1. Програм има грешку, јер је метод **Metod(int i)** надјачан (предефинисан) са различитим потписом у класи B.
2. Програм има грешку, јер се **b.Метод(5)** не може позвати пошто је метод **Metod(int i)** заклоњен у класи B.
3. Програм има грешку због **b.i**, јер је поље **i** неприступачно из класе B.
4. Програм нема грешке, јер наслеђени метод класе A, **Metod(int i)** није надјачан у класи B, већ је дефинисан преоптерећен метод **Metod(string s).**

92. Дат је део кода који је написан на C# програмском језику. Анализирати код и одредити шта ће се приказати на излазу.

```
try
{
    int x = 0;
    int y = 5 / x;
}
catch (Exception e)
{
    Console.WriteLine("Exception");
}
catch (ArithmeticException ae)
{
    Console.WriteLine(" Arithmetic Exception");
}
Console.WriteLine("finished");
```

Заокружи број испред тачног одговора:

1. Приказује се текст: finished
2. Приказује се текст: Exception
3. Ништа. Дешава се грешка приликом компајлирања
4. Приказује се текст: Arithmetic Exception

2

93. Заокружити број испред исказа који представља исправан наставак дате реченице:

Ако try-catch наредба има више catch блокова у којима "хватамо" изузетак основне ***Exception*** класе, заједно са изузецима других класа изведених из класе ***Exceptions***...

1. онда се изузетак основне Exception класе може „хватати" у било ком catch блоку (редослед није битан, битно је да се наведу све могуће грешке)
2. онда се изузетак основне Exception класе мора „хватати" у последњем catch блоку
3. онда се изузетак основне Exception класе мора „хватати" у првом catch блоку
4. основна Exception класа се не комбинује у истој наредби са класама изведеним из ње јер их оснавна класа „маскира"

2

У следећим задацима заокружите бројеве испред тражених одговора

94. Дати су изкази који се односе на правила писања try-catch-finally блокова за руковање изузетима. Заокружити бројеве испред исказа који су тачни:

1. Блок try мора имати бар један catch или један finally блок
2. Блок try може имати више catch блокова
3. Ако блок try има више catch блокова, изузетак основне Exception класе мора се хватати у првом catch блоку
4. Ако блок try има више catch блокова, битан је редослед њиховог писања
5. Блок try мора имати бар један finally блок
6. Блок try не сме да има више catch блокова

1,5

95. Да би наслеђени метод могао да се **редефинише** и тиме измени његова функционалност у класама наследницама, у родитељској класи испред ознаке повратног типа метода наводи се нека од понуђених кључних речи.

Заокружити бројеве испред кључних речи које омогућавају редефинисање дефинисаног метода кроз ланац наслеђивања:

1. new
2. virtual
3. sealed
4. override
5. abstract
6. base
7. довољно је да буде public или protected

1,5

96. Дата је наредба кода у програмском језику C# која представља декларацију низа**.** Проценити које од доле наведених декларација су тачне.

Заокружити бројеве испред очекиваних одговора:

1. int niz = new int(30);
2. double[ ] niz = new double[30];
3. int[ ] niz = { 3, 4, 3, 2 };
4. char[ ] niz = new char[ ];
5. char[ ] niz = new char { 'a', 'b', 'c', 'd' };
6. char[ ] niz = new char[ ] { 'a', 'b' };

1,5

97. Дат је код на C#-у којим су креиране три класе у ланцу наслеђивања. Имајући у виду класификаторе приступа пољима класа, заокружити бројеве испред поља која ће бити видљива унутар класе Sin:

```
public class Deda {
   private double penzija;
   protected string adresa;
   public string ime;
}
public class Otac: Deda {
   private double plata;
   protected string struka;
}
public class Sin: Deda {
   public int razred;
}
```

1. penzija
2. adresa
3. ime
4. plata
5. struka
6. razred

1,5

98. Дат је код на C#-у којим су креиране три класе у ланцу наслеђивања. Унутар сваке класе декларисан је по један private, public и protected атрибут. У методи ***Main()*** класе ***Program*** креиран је објекат s класе ***Sin*** (`Sin s = new Sin();`) Заокружити бројеве испред поља која ће бити видљива у креираном објекту s класе Sin:

```
public class Deda {
   private double penzija;
   protected string adresa;
   public string ime;
}
public class Otac: Deda {
   private double plata;
    protected string  firma;
    public string struka;
}
public class Sin: Otac {
    private double prosek;
    protected int razred;
    public string skola;
}
```

1. penzija
2. adresa
3. ime
4. plata
5. struka
6. firma
7. prosek
8. razred
9. skola

1,5

99. Дата је дефиниција класе у програмском језику C# и састоји се од два конструктора, методе и поља x и y. У петом реду дефинисан је конструктор са параметрима који формира тачку са координатама x и y. Заокружити наредбе којима се може допунити дефиниција конструктора:

```
1. public class Point {
2.     private double x, y;
3.     public Point() { x = 0; y = 0; }
4.     public void set(double xx, double yy) { x = xx; y = yy; }
5.     public Point(double x, double y) { _____________; }
6. }
```

Заокружити бројеве испред тачних одговора:

1. this.x=x; this.y=y;
2. x=x; y=y;
3. set(x,y);
4. set(this.x,this.y);
5. x=this.x; y=this.y;

2

100 Дати су делови кода у програмском језику C# који треба да рачунају збир елемената матрице **a,** декларисане на следећи начин: `int[,] a = new int[10, 10];`. Анализирати дате кодове и проценити који од предлога је тачан.

Заокружити бројеве испред очекиваних одговора:

1.
```
int sum = 0;
for (int i = 0; i < b.Length; i++)
                for (int j = 0; j < b[i].Length; j++)
                    sum3 += b[i][j];
```

2.
```
int sum = 0;
foreach (int x in a) sum1 += x;
```

3.
```
int sum = 0;
for (int i = 0; i < a.GetLength(0); i++)
                for(int j=0; j<a.GetLength(1); j++)
                    sum2 += a[i,j];
```

4.
```
int sum = 0;
foreach (int[] vrsta in b)
                foreach (int el in vrsta)
                    sum4 += el;
```

2

101 Дате су наредбе у програмском језику C# које дефинишу заглавље методе Print() са променљивим бројем параметара. Одредити који од понуђених одговора су исправни.

Заокружити бројеве испред очекиваних одговора:

1. `public void Print(params string[] niska, params double[] broj)`
2. `public void Print(params double[] broj, string niska)`
3. `public void params Print(double d1, double d2)`
4. `public void Print(params double[] broj)`
5. `public void Print(int n, params double[] broj)`

2

102 Дата је дефиниција класе у програмском језику C# и састоји се од два конструктора, једне методе и поља x. У дефиницији се користи службена реч **this**. Анализирати дати код и проценити тачност следећих исказа. Заокружити бројеве испред тачних исказа:

```
class TestPrimer {
        public double x;
        public TestPrimer(double x){
            this.fun();
            this.x = x;
        }
        public TestPrimer(){
            Console.WriteLine("Podrazumevani konstruktor");
            this(23);
        }
        public void Fun(){
            Console.WriteLine("Poziv metoda fun()");
        }
    }
```

1. this.Fun() у конструктору TestPrimer(double x) може се поједноставити и заменити само са Fun().
2. this.x у конструктору TestPrimer(double x) може се поједноставити и заменити само са x.
3. позив конструктора this(23) унутар другог конструктора TestPrimer() је прво шта се извршава и мора се писати одмах после декларације `public TestPrimer():this(23)`
4. this(23) у конструктору Test() мора се заменити са прецизнијим изразом this(23.0).

2

103 Дати су искази који у програмском језику C# дефинишу конструктор.

Заокружити бројеве испред очекиваних одговора:

1. Подразумевани конструктор без параметара се увек аутоматски додаје класи.
2. Подразумевани конструктор без параметара се класи аутоматски додаје уколико у њој није експлицитно дефинисан ниједан конструктор.
3. У класи се мора експлицитно дефинисати бар један конструктор.
4. Конструктори немају тип резултата, чак ни **void.**

2

104 Заокружити бројеве испред наведених чланова класе који се ни под којим условима **НЕ** наслеђују са родитељске класе на изведену класу:

1. Readonly својства
2. Заштићени чланови класа
3. Својства (property)
4. Приватни чланови класа
5. Конструктор класе

2

105 Дат је код програма у програмском језику C#. Код садржи објекте две класе у којима је дефинисан метод ***ToString().*** Анализирати код датог програма и одредити који од датих исказа су тачни.

```
namespace TestPrimer {
    class Program {
        static void Main(string[] args) {
            Object a = new Klasa();
            Object obj = new Object();
            Console.WriteLine(a);
            Console.WriteLine(obj);
        }
    }
}
class Klasa{
   int x;
   public override string ToString() {return "x u A je " + x;}
}
```

2

Заокружити бројеве испред очекиваних одговора:

1. Програм има грешку, јер наредбу ***Console.WriteLine(a)*** треба заменити наредбом ***Console.WriteLine(a.ToString()).***
2. Приликом извршавања наредбе ***Console.WriteLine(a)***, програм позива се метод ToString() наслеђен из класе Object.
3. Приликом извршавања наредбе ***Console.WriteLine(a)***, програм позива метод ToString() из класе Klasa.
4. Приликом извршавања наредбе ***Console.WriteLine(obj)***, програм позива метод ToString() из класе Object.

106 У програмском језику C# дата је декларација две класе: ***KlasaA*** и ***KlasaB*** која наслеђује класу ***KlasaA***. Анализирати дате класе и проценити који од понуђених исказа су тачни.

```
namespace TestPrimer {
    class Program {
        static void Main(string[] args) {
            KlasaB b = new KlasaB();
            b.Print();
        }
    }
    class KlasaA {
        string s;
        public KlasaA(string s) { this.s = s; }
        public void Print() { Console.WriteLine(this.s); }
    }
    class KlasaB :KlasaA{ }
}
```

2

Заокружити бројеве испред очекиваних одговора:

1. Програм има грешку, јер ***KlasaB*** нема подразумевани конструктор ***KlasaB***().
2. Програм има грешку јер ***KlasaB*** има подразумевани конструктор, док родитељска ***KlasaA*** нема такав конструктор. Програм би радио без грешке уколико би се уклонио конструктор са параметрима из ***KlasaA***.
3. Програм има грешку која се може отклонити уколико би се у ***KlasaA*** експлицитно додао конструктор без параметара ***KlasaA***().
4. Програм нема грешку, извршава се, али се на конзоли ништа не исписује јер је поље ***s*** добило подразумевану вредност ***String.Empty***

107 У класи **Figura** дат је подразумевани (deafult) конструктор и конструктор са 4 параметра:

```
public Figura() {...}
public Figura(string ime, string boja, int pozX, int pozY) {...}
```

Заокружити бројеве испред исправно написаних наредби креирања објеката класе Figura:

1. `Figura f = Figura("lovac", "beli", 7, 3);`
2. `Figura f = new Figura("beli", "lovac", 7, 3);`
3. `Figura f = new Figura();`
4. `Figura f = new Figura("lovac", 3, 7, "beli");`
5. `Figura f = new Figura("lovac", "beli", 3, 7);`
6. `Figura f = new Figura("lovac", "beli", 3);`

3

Допуните следеће реченице и табеле

108. Дате су започете изјаве које се односе на делове кода за обраду изузетака. Довршити започете реченице:

Наредбе које се извршавају у случају настанка грешке стављају се унутар блока ___catch___

Наредбе које се извршавају и ако се деси и ако се не деси грешка стављају се унутар блока ___finally___

Наредбе које могу узазвати грешку стављају се унутар блока ___try___

1,5

109. На програмском језику C# дефинисане су две класе:

```
public class Racun {
   public virtual int Uvecaj() { return 10; }
}
public class Dinarski: Racun {
   public override int Uvecaj() { return 20 * base.Uvecaj(); }
}
public class Devizni : Racun {
   public override int Uvecaj() { return 50 + base.Uvecaj(); }
}
```

Унутар функције Main, креирана су три објекта ових класа на следећи начин:

```
Racun r = new Racun();
Racun rDin = new Dinarski();
Racun rDev = new Devizni();
```

Анализирати код и на предвиђене линије уписати шта метод Uvecaj() враћа при позиву из наведених објеката:

```
r.Uvecaj();      ___10___
rDin.Uvecaj();   ___200___
rDev.Uvecaj();   ___60___
```

3

110. На програмском језику C# дефинисане су две класе:

```
public class Racun {
   public virtual int Uvecaj() { return 10; }
}
public class Dinarski: Racun {
   public override int Uvecaj() { return 20 * base.Uvecaj(); }
}
public class Devizni : Dinarski {
   public override int Uvecaj() { return 50 + base.Uvecaj(); }
}
```

Унутар функције Main, креирана су три објекта ових класа на следећи начин:

```
Racun r = new Racun();
Racun rDin = new Dinarski();
Racun rDev = new Devizni();
```

Анализирати код и на предвиђене линије уписати шта метод Uvecaj() враћа при позиву из наведених објеката:

```
r.Uvecaj();
rDin.Uvecaj();
rDev.Uvecaj();
```

10
200
250

3

111. На програмском језику C# дефинисане су две класе:

```
public class KlasaA {
   public virtual int Metod() { return 10; } }
public class KlasaB : KlasaA {
   public override int Metod() { return 20; } }
public class KlasaC : KlasaB {
   public new int Metod() { return 30; } }
```

Креирани су објекти ових класа и из њих позвана метода ***Metod().*** На предвиђене линије уписати шта метод ***Metod()*** враћа при позиву из наведених објеката:

```
KlasaA a = new KlasaA(); a.Metod()
KlasaB b = new KlasaB(); b.Metod()
KlasaA bb = new KlasaB(); bb.Metod()
KlasaC c = new KlasaC(); c.Metod()
KlasaB cc = new KlasaC(); cc.Metod()
KlasaA ccc = new KlasaC(); ccc.Metod()
```

враћа вредност
враћа вредност
враћа вредност
враћа вредност
враћа вредност
враћа вредност

10
20
20
30
20
20

3

112. На програмском језику C# дефинисане су две класе:

```
public class Roditelj {
     public virtual void Poruka1(){ Console.WriteLine("R1"); }
     public void Poruka2(){  Console.WriteLine("R2");  }
}
public class Dete: Roditelj {
     public override void Poruka1(){ Console.WriteLine("D1"); }
     public new void Poruka2(){ Console.WriteLine("D2"); }
}
```

Унутар функције Main креирана су два објекта ових класа на следећи начин:

```
Dete x = new Dete();
Roditelj y = new Dete();
```

Проценити ефекат извршења наведених позива и на предвиђене линије уписати шта ће се видети на стандардном излазу извршењем позваних метода:

```
x.Poruka1();
x.Poruka2();
y.Poruka1();
y.Poruka2();
```

D1
D2
D1
R2

4

У следећим задацима уредите и повежите појмове према захтеву

113. Са леве стране дате су кључне речи које одређују типове класа, а са десне су описи класа. На линију испред описа уписати редни број под којим је наведен одговарајући тип класе:

1.	abstract	3	Класа која се простире у више фајлова
2.	sealed	4	Класа садржи само декларације метода, али не и дефиницију (тело) методе
3.	partial	1	Класа која се не може инстанцирати
4.	interface	2	Класа из које се не може наслеђивати

2

114. Са десне стране су наведене су области видљивости појединих елемената класе, а са леве стране класификатори приступа којима се врши контрола области видљивости. На линију испред описа области видљивости унети редни број под којим је наведен одговарајући класификатор приступа:

1.	private	3	видљив унутар класе у којој је дефинисан, као и унутар изведених класа
2.	public	1	видљив само унутар класе у којој је дефинисан
3.	protected	4	видљив унутар пројекта у коме је дефинисан
4.	internal	2	видљив и ван своје класе у којој је дефинисан

2

115. Са леве стране су наведени делови/елементи класе, а са десне стране улоге појединих класних елемената. На линију испред описа улоге унети редни број под којим је наведен одговарајући елеменат класе:

1.	поље (атрибут)	4	Опис функционалности објеката класе
2.	деструктор	5	Контрола приступа пољима класе
3.	конструктор	1	Опис особина објеката класе
4.	метод	3	Креирање објеката класе
5.	својство / property	2	Уништавање објеката класе

2,5