Asistent: Jurica Đurić; jurica.duric@vsite.hr

## Zadatak 1: Razlika između klase i strukture (referentni, odn. vrijednosni tip)

Definirati klasu i strukturu koje će svaka za sebe predstavljati kompleksni broj:

```
class KompleksniBrojKlasa {
    public double RealniDio;
    public double ImaginarniDio
}

struct KompleksniBrojStruktura {
    public double RealniDio;
    public double ImaginarniDio
}
```

Dodati konstruktore za oba tipa, koji će kao argumente primati dva `double` broja: realni i imaginarni dio novog kompleksnog broja.

Također, za oba tipa pregaziti (*override*) metodu `ToString` koja će omogućiti formatirani ispis kompleksnog broja:

```
public override string ToString() {
    return string.Format("{0} + {1}i", RealniDio, ImaginarniDio);
}
```

Inicijalizirati po jedan kompleksni broj referentnog, odnosno vrijednosnog tipa:

```
KompleksniBrojKlasa kbk = new KomplesniBrojKlasa(1, 2);
KompleksniBrojStruktura kbs = new KompleksniBrojStruktura(3, 4);
```

te ispisati njihove realne i imaginarne dijelove.

Definirati nove instance oba tipa i pridružiti im vrijednosti postojećih:

```
KompleksniBrojKlasa kbk2 = kbk;
KompleksniBrojStruktura kbs2 = kbs;
```

te ispisati njihove realne i imaginarne dijelove.

Promijeniti vrijednosti članova novih instanci, npr.:

```
kbk2.RealniDio = 5;
kbs2.RealniDio = 6;
```

te ispisati realne (i imaginarne) dijelove svih instanci: `kbk`, `kbs`, `kbk2` i `kbs2`. Što možete zaključiti na osnovu ispisanih vrijednosti?

Asistent: Jurica Đurić; jurica.duric@vsite.hr

## Zadatak 2: Klasa za rješavanje kvadratne jednadžbe

Napisati klasu za rad s kvadratnom jednadžbom: zadavanje koeficijenata (*a*, *b* i *c*), računanje vrijednosti *y = f(x)* te računanje realnih korijena.

1. Napisati konzolnu aplikaciju `QuadraticEquation` za rješavanje kvadratne jednadžbe:

   $$y = a \cdot x^2 + b \cdot x + c$$

   Klasa `QuadraticEquation` neka sadrži privatna polja

   ```
   private double m_a;
   private double m_b;
   private double m_c;
   ```

   koja predstavljaju koeficijente kvadratne jednadžbe, te dva konstruktora:

   ```
   public QuadraticEquation();
   public QuadraticEquation(double a, double b, double c);
   ```

   Napisati implementacije oba konstruktora.

2. Implementirati svojstva (*property*) `A`, `B` i `C` za dohvaćanje i postavljanje koeficijenata kvadratne jednadžbe (`get` i `set`).

3. Napisati metodu:

   ```
   public double Y(double x);
   ```

   koja će vraćati vrijednost kvadratne jednadžbe za zadani *x*.

4. Napisati svojstvo:

   ```
   public double Discriminant { get; }
   ```

   koja će računati i vraćati vrijednost diskriminante kvadratne jednadžbe:

   $$D = b^2 - 4 \cdot a \cdot c.$$

5. Napisati svojstvo:

   ```
   public double[] Roots { get; }
   ```

   koja će računati i vraćati korijene kvadratne jednadžbe:

   $$\frac{-b + \sqrt{b^2 - 4ac}}{2a} = \frac{-b + \sqrt{D}}{2a}$$
   $$\frac{-b - \sqrt{b^2 - 4ac}}{2a} = \frac{-b - \sqrt{D}}{2a}$$

   Za računanje kvadratnog korijena koristiti statičku metodu `Sqrt()` iz klase `System.Math`. Testirati klasu za sljedeće vrijednosti:

| *a* | *b* | *c* | 1. korijen | 2. korijen |
|---|---|---|---|---|
| 1 | 2 | –3 | 1 | –3 |
| –1 | 2 | –1 | 1 | 1 |

6. U računanje korijena ubaciti provjeru da li je diskriminanta *D* veća ili jednaka 0 (tj. da li kvadratna jednadžba ima realne korijene). Ako je diskriminanta manja od 0, baciti iznimku

tipa `NotRealRootsException`. Napisati implementaciju tog tipa tako da bude izveden iz tipa `System.ArithmeticException`:

```
public class NotRealRootsException : ArithmeticException {
  public NotRealRootsException()
  {
  }

  public NotRealRootsException(string message) : base(message)
  {
  }
}
```

Testirati (hvatanjem iznimke) ispravnost za $a = 1$, $b = 0$, $c = 1$.

## Zadatak 3: Struktura za računanje s kompleksnim brojevima

Proširiti funkcionalnost programa za rješavanje kvadratne jednadžbe tako da može računati i kompleksne korijene.

1. Implementirati strukturu `Complex` za rad s kompleksnim brojevima. Struktura neka sadrži dva polja:

   ```
   private double m_real;
   private double m_imaginary;
   ```

   koji će sadržavati realni, odnosno imaginarni dio kompleksnog broja.

2. Implementirati konstruktore:

   ```
   public Complex(double real, double imaginary);
   public Complex(double real);
   ```

3. Implementirati svojstva (*property*):

   ```
   public double Re { get; set; }
   public double Im { get; set; }
   ```

4. Implementirati operatore zbrajanja, oduzimanja, množenja i dijeljenja kompleksnih brojeva:

   ```
   public static Complex operator+(Complex c1, Complex c2) { ... }
   public static Complex operator-(Complex c1, Complex c2) { ... }
   public static Complex operator*(Complex c1, Complex c2) { ... }
   public static Complex operator/(Complex c1, Complex c2) { ... }
   ```

   Množenje i dijeljenje definirani su sljedećim formulama:

   $$(a + b\,i) \cdot (c + d\,i) = (ac - bd) + (ad - bc)\,i$$
   $$\frac{a + b\,i}{c + d\,i} = \frac{(ac + bd) + (bc - ad)\,i}{c^2 + d^2}$$

   Testirati operatore za sljedeće izraze:

   $$(1 + 2\,i) + (3 + 4\,i) = 4 + 6\,i$$
   $$(3 + 4\,i) - (2 + 1\,i) = 1 + 3\,i$$
   $$(1 + 2\,i) \cdot (3 + 4\,i) = -5 - 2\,i$$

$$\frac{5+5i}{1+2i} = 3 - i$$

5. Implementirati operator implicitne pretvorbe iz tipa `double`:

   ```
   public static implicit operator Complex(double number) { ... }
   ```

   Testirati operator sljedećom naredbom:

   ```
   Complex compl = 5;
   ```

6. Implementirati statičku metodu koja će računati općenite korijene broja tipa `double`:

   ```
   public static Complex[] Sqrt(double number) { ... }
   ```

   Za pozitivni argument metoda mora vratiti:

   $\pm\sqrt{x}$,

   a za negativni argument mora vratiti:

   $\pm i\sqrt{x}$

   Testirati metodu za pozitivni i negativni argument (npr. +4 i –4).

7. Implementirati metodu `ToString` metodu koja će ispisati vrijednost kompleksnog broja u obliku:

   "*realni_dio* + *imaginarni_dio* i"

8. U klasi `QuadraticEquation` izmijeniti metodu `Roots` tako da vraća korijene kao kompleksne brojeve:

   ```
   public Complex[] Roots { get; }
   ```