Task 1. - Complex numbers

Complex numbers are an extension to the concept of real numbers. Originally, complex numbers have been introduced for the sake of defining a closed formulation of the problem to find the roots of algebraic equations. In structural engineering, complex numbers play a fundamental role in the theory of vibrations.

The set

$$\mathbb{C} = \{z \mid z = \alpha + i\beta;\ \alpha, \beta \in \mathbb{R}\}$$

is called the set of complex numbers in which i denotes the imaginary unit defined by $i^2 = -1$. For the complex number $z = \alpha + i\beta$ we call α the real part and β the imaginary part.

1. Define new complex numbers with given real and imaginary parts. For example, the two complex numbers z_1 and z_2 can be defined as

$$z_1 = \alpha_1 + i\beta_1, \quad z_2 = \alpha_2 + i\beta_2$$

2. Given a complex number z_1, we can add a second complex number z_2 to it, giving

$$z_1 + z_2 = (\alpha_1 + \alpha_2) + i(\beta_1 + \beta_2).$$

Similarly, we can subtract z_2 from z_1, giving

$$z_1 - z_2 = (\alpha_1 - \alpha_2) + i(\beta_1 - \beta_2).$$

3. The absolute value of $z = \alpha + i\beta$ is

$$|z| = \sqrt{\alpha^2 + \beta^2}.$$

4. Finally, printing a complex number such as $3.11 - 7.34i$ to the console should look something like

```
3.11-7.34i
```

Želimo deklarirati klasu kompleksnih brojeva koja pohranjuje realni i imaginarni dio i obezbjeđuje metode za istovremeno izvođenje gore navedenih operacija.

To design a class in Java, the first and a very important step (!!!) is to write an UML diagram.

- UML diagram opisuje nerelevantne aspekte programa bez bavljenja svim detaljom.

- UML je jezik za crtanje softverskih modela.

- Ima 12 dijagrama ⇒ MI KORISTIMO CLASS DIAGRAM.

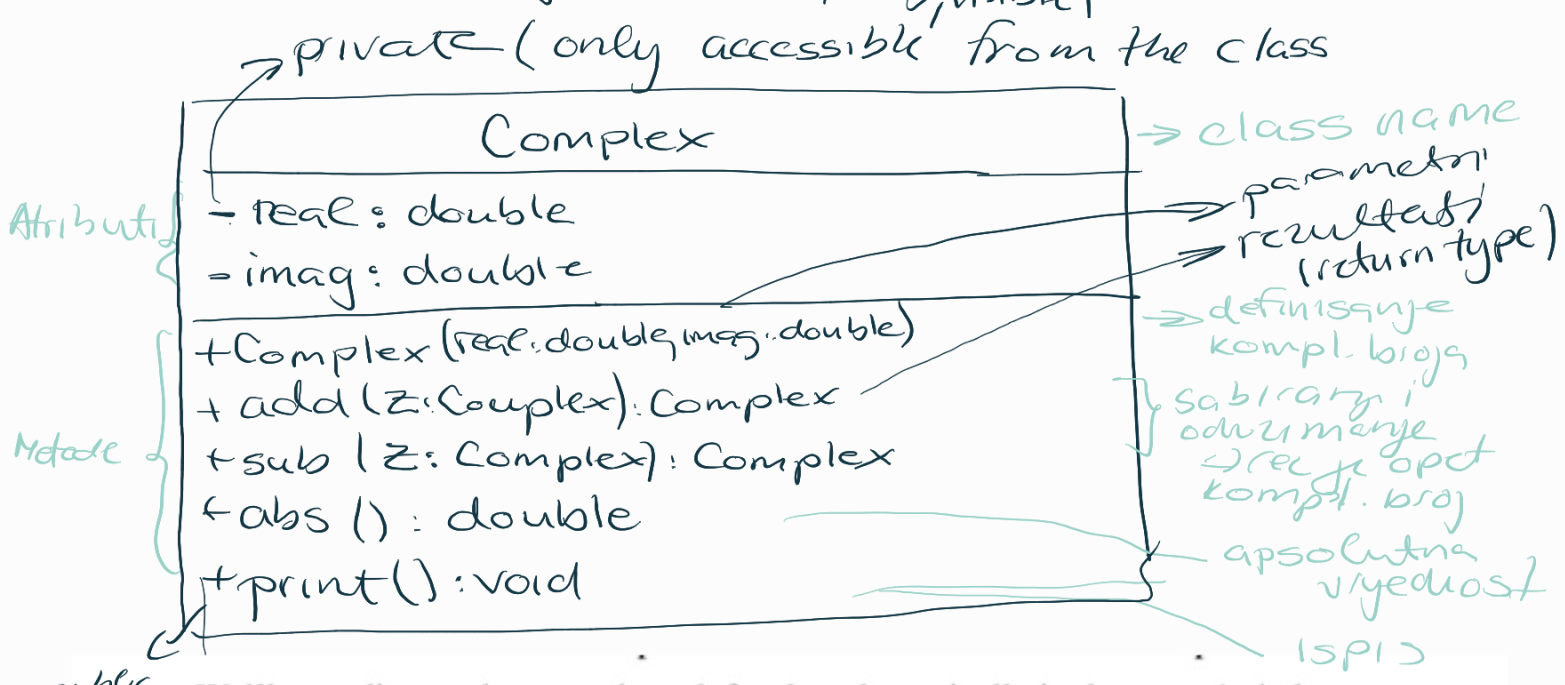
- CLASS DIAGRAM ima 3 sekcije:

1. Class name
2. Attributes
3. Methods

UML diagram za klasu Complex

- Class name: Complex (dobar naziv - kratko opisuje konceptualnu ideju klase) | Complex Numbers je pretjerano, CN prekriptično, tj. nedovoljno jasno, ComplexNumbers je množina ⇒ nije dobro jer svaka klasa odnosi samo jedan objekt

- Atributi: kompleksni broj je definisan svojim realnim i imaginarnim dijelom. "double" se koristi za realne brojeve.
- Metode: definišu kako objekat klase interagira sa ostalim dijelovima programa (visible)



We'll now discuss the operations defined mathematically in the steps 1–4 above.

- In step 1, the complex numbers z_1 and z_2 are initialized. In Java the methods used to create and initialize an object are called constructors. Constructors have the same name as the class, can take any parameter list and have no return type (in that sense constructors are special). For our purpose, it is convenient to define a constructor taking the two parts of a complex number as arguments.
- In step 2, we take the given complex number z_1 and add a second complex number z_2 to it, resulting in a third complex number, namely the sum of z_1 and z_2. Thus we equip the `Complex` class with a method `add` taking another `Complex` object as an argument and returning the sum as a further `Complex` object. Similarly, we also define a `sub` method.
- In order to calculate the absolute value of a given complex number z as defined in step 3, no further information is needed. Therefore, the `abs` method needs no parameters and simply returns a `double` value, namely $|z|$.
- Finally, we will introduce the `print` method to display the contents of a complex number object to the screen (in the console window). It does not compute anything and therefore the return type is `void` (which means nothing, see also Table 1.1 on page 13 for further information).

Java code (moja verzija):

```
public class Compex {                    → glavna klasa
    private double real;
    private double real;                 } Definisemo atribute za klasu → koriste se za sve metode
```

// sada definišemo pojedinačne metode, a zatim konačan program koji će konkretno da se primjenjuje tzv. TEST CASE kojim isprobavamo metode

// Prvo definišemo "class metodu" Complex za generisanje KOMPLEKSNOG BROJA

javna → public | naziv → Complex | ulazi → dva realna broja, od kojih jedan čini realan a drugi imag. dio kompleksnog br.

```
(double real, double imag) {
    this.real = real;        ← adrese parametara
    this.imag = imag;        ← dodjeljujemo atributima
}                            → nema izlaza
```

// Sada generišemo metodu za ispis

→ ne daje nikakav rezultat

```
void print() {
    System.out.println("The complex number is
    z" + real + "+" + imag + "i");
}
```

→ nema izlaza → samo se izvrši kod

// Sada generišemo metodu za sabiranje

javna ← tip izlaza; naziv; sabiramo realni i realni, imag. i imag. dio

```
public Complex add (Complex z) {
  double r = this.real + z.real;
  double i = this.imag + z.imag;
  Complex result = new Complex (r,i);
  return result;   }
```

→ lokalna varijabla koja sprema klasu

→ vraća nam taj rezultat dok dodjeljuje tu vrijednost nekoj varijabli kao i

// Metoda za oduzimanje → isti princip za sabiranje

```
public Complex sub (Complex z) {
  double r = this.real - z.real;
  double i = this.imag - z.imag;
  Complex result = new Complex (r,i);
  return result; }
```

// Metoda za određivanje apsolutne vrijednosti

← javna metoda → naziv; nema ulaza; nema ulaza

```
public abs () {
  double result;
  return result = Math.sqrt (Math.pow(this.real,2)
                  + Math.pow (this.imag,2);
}
```

// Metoda za množenje

```
public Complex multiply (Complex z) {
double r = this.reae * z.real - this.imag * z.imag;
double i = this.real * z.imag + this.imag * z.real;
 Complex result = new Complex (r,i);
 return result}
```

ulaz — tip izlaza; naziv

// Metoda za dijeljenje

```
public Complex division (Complex z) {
double r = ((this.reae * z.reae) + (this.imag * z.imag))
          / (Math.pow (z.real,2) + Math.pow(z.mag,2));

double i = ((z.real * this.imag - this.reae * z.imag))
          / (Math.pow (z.reol,2) + Math.pow(z.imag,2)

Complex result = new Complex (r,i);
return result; }
```

// Sada generisemo Test case - deo koda za kvrsavanje operacija. Program krece od ove tacke i prvi su sami "ALATI"

```
public static void main (String[] args) {
// Generisanje jednog kompleksnog broja
    Complex z1 = new Complex (3,4);
    z1.print ();
```

// Sabiranje

```
Complex z2 = new Complex (3,4);
Complex z3 = z1.add (z2);
z3.print ();
```

(z2) -> parametri

z1. -> this. se odnosi ovo

// Oduzimanje

```
Complex z4 = z1.sub (z2);
z4.print();
```

// Apsolutna vrednost

```
System.out.println ("Apsolutna vrednost je " +
                    z1.abs());
```

```
// Množenje
Complex z5 = z1.multiply(z2);
z5.print();

// Dijeljenje
Complex z6 = z1.division(z2);
z6.print();

// Sada ćemo ove metode provjeriti za
razlicite brojeve

// Izraz 1: (3+4i)^2
Complex z7 = new Complex(3,4);
Complex res1 = z7.multiply(z7);
System.out.println("Rezultat prvog izraza");
res5.print();

// Izraz 2: (3+4i)^5
Complex z8 = new Complex(3,4);
Complex res2 = z8.multiply(z8);
Complex res3 = res2.multiply(z8);
Complex res4 = res3.multiply(z8);
Complex res5 = res4.multiply(z8);

res5.print();
```

// Izraz 3 : 6 − (4 − i)

```
Complex z9 = new Complex(6, 0);
Complex z10 = new Complex(4, -1);

Complex res6 = z9.sub(z10);
res6.print();
```

// Izraz 4 : ((25 (3 + 5i) / (3 − 4i) + 11) / 27)²

```
Complex z11 = new Complex(25, 0);
Complex z12 = new Complex(3, 5);
Complex z13 = new Complex(3, -4);

Complex z14 = new Complex(11, 0);

Complex z15 = new Complex(27, 0);

Complex res7 = ((( z11.multiply(z12)).division(z13)).add(z14));
Complex res8 = res7.multiply(z15);
Complex res9 = res8.multiply(res8);
res.9();
```