INTRODUCTION

The declaration for a method or a constructor declares the number and the type of the arguments for that method or constructor.
For example, the ***Circle*** class (Listing 1) has *one constructor with 4 parameters* and *one method with 1 parameter* (**marked in red**) .

Listing 1.

```
public class Circle{
       //private fields (marked in blue):
       private double radius = 1;
       private double x = 0; //(x,y) coordinates of the circle center
       private double y = 0;
       private String name = "Circle";

       //public constructors:
       public Circle(){}
       public Circle(double radius, double x, double y, String name)
       {    this.radius = radius;
            this.x = x;
            this.y = y;
            this.name = name;
       }

        //public methods getXxxx/setXxx:
        public double getRadius() {
           return radius;
        }
        public void setRadius(double radius) {
            this.radius = radius;
        }

   } //end of Circle class
```

One of the constructors has four parameters: the ***radius***, the ***x*** and ***y*** coordinates and the ***name*** of a circle. The first three are double-precision floating point numbers, and the fourth is an String. The parameters are used in the method body and at runtime will take on the values of the arguments that are passed in.

Note: ***Parameters*** refers to the list of variables in a method declaration. ***Arguments*** are the actual values that are passed in when the method is invoked. When you invoke a method, *the arguments used must match the declaration's parameters in type and order*.

To create the circle object using the constructor with parameters you have to use 4 arguments, for example:

```
      Circle c1=new Circle(10.5, 0.5, 1.5, "My first circle");
```

To change the radius for the circle ***c1*** you can use the ***setRadius*** method with one argument:

```
      c1.setRadius(125);
```

You can use any data type for a parameter of a method or a constructor. This includes primitive data types, such as doubles, floats, and integers, and reference data types, such as objects and arrays.

Ex. 1. Other methods in the Circle class

a) Create the new **Lab3** project (***Java application***) and add new file to the project (***File/New file -> Java Class***) entitled **Circle**.
b) Copy the above definitions of **Circle** class (**Listing 1**) to the newly created file
c) Add other methods ***set/get*** to this class. Netbeans will help you with this - right click, select ***Insert Code*** and ***Add Getter and Setter***
d) Check what methods were created, their names and what are the parameters
e) Navigate to the file with the application (***Lab3.java***) and in the ***main*** method, create two objects of ***Circle*** class and then try to modify their radii, names, coordinates of the center. These Setter methods allow you to modify only the values of individual fields.
f) Go to the class definition file (***Circle.java***) and add the other methods (consider which method and what kind of parameters needs):
- `showCircleInfo` - method displays all information about a circle
- `moveCenter` - method modifies the coordinates of the center

g) Test the functionality of new methods

Ex. 2. The Fraction class - operations with fractions (Listing 2)

1. Define the **Fraction** class with:
   - **two private** <u>integer</u> fields: *numerator*, *denumerator* ,
   - **two public** constructor (first with two parameters, second – without parameters setting nominator and denominator =1), if the denominator equals 0 – change his value into 1 and print message about change.
   - public method *show* to show a fraction like: 12/13.
2. four **public** methods: *add*, *subtract*, *multiply* and *divide* –the method returns the fraction as the result of the addition (Listing 2), subtraction, multiplication and division on two fractions: eg.
   $\frac{2}{3}+\frac{3}{4}=\frac{17}{12}$,
   $\frac{2}{3}-\frac{3}{4}=\frac{-1}{12}$,
   $\frac{2}{3}\cdot\frac{3}{4}=\frac{6}{12}$,
   $\frac{2}{3}:\frac{3}{4}=\frac{8}{9}$.
   .Test your **Fraction** class in the main method (Listing 3):
3. Add the next public methods to the **Fraction** class:
   - *getGreater* – choose and **return** the grater of two fraction,
   - *getSmaller*- choose **and return** the smaller of two fraction,
   - *isSmaller* – check if first fraction is smaller than second one and **return** true or false
4. Test your methods for three fractions: 2/3,3/4 and 5/6:
   - find the values of:
     $\left(\frac{2}{3}+\frac{3}{4}\right):\frac{5}{6}$,
     $\frac{2}{3}\cdot\frac{3}{4}+\frac{5}{6}$.
   - choose the max and minimum of all tested fractions,
   - reduce the max fraction (define additional methods in Fraction class – you may use the algorithm described on the website (but it needs a little modification in our Fraction class): *https://www.geeksforgeeks.org/reduce-the-fraction-to-its-lowest-form/*

Listing 2. Fraction.java

```
public class Fraction {
    private int numerator=1;
    private int denumerator=1;

    //add constructors
    // add getter/setter methods
    // add show method

    //method to add two Fraction objects (this and f1),
    //returns also Fraction object:
    public Fraction add(Fraction f1){
      Fraction f=new Fraction();
      f.numerator=this.numerator*f1.denumerator +
                  f1.numerator*this.denumerator;
      f.denumerator=f1.denumerator*this.denumerator;
      return f;
    }

    //add other methods: subtract, multiply and divide
    //add other methods

}
```

Listing 3. Lab3.java – test your Fraction class

```
public class Lab3 {
    public static void main(String[] args) {
        Fraction f1=new Fraction(2,3);
        Fraction f2=new Fraction(3,4);
        Fraction f3=f1.add(f2);
        System.out.println(" Calculation f1+f2=f3");
        System.out.println( f1.show()+ "+" + f2.show() + "=" + f3.show() );
        //test other methods . . .
    }

}
```