Tutorial - Week 02 5COSC019W – Object Oriented Programming – Java

Implement classes and objects, instance variables and instance methods, UML notation for classes

02/10/2023 – 06/10/2023

Note: The answers to the following questions are provided in RED. ***Try first to reply by yourself*** *and only after reading the correct answer.*

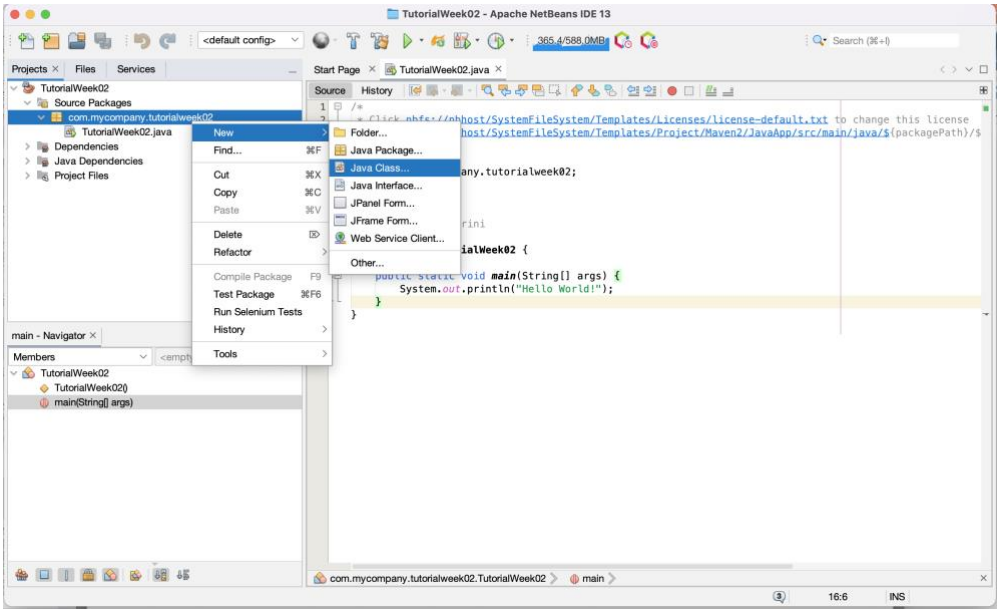
The solution code will be provided in BB after one week, so you will have the time to try or investigate your solution.

How to create a Class

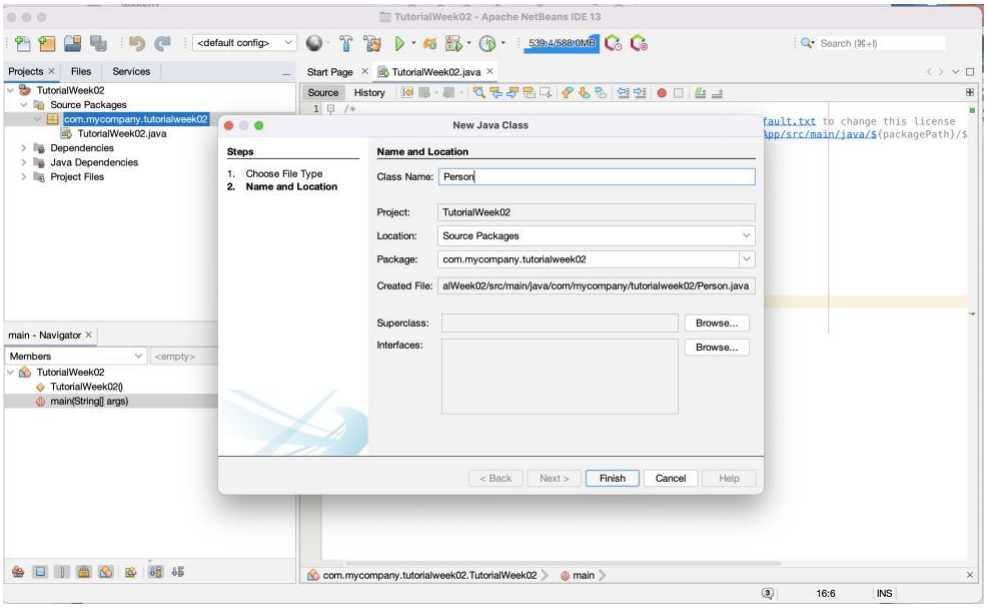
1) Create a new project in NetBeans called *TutorialWeek02*.
2) Let's create a simple new java class called **Person** that defines three instance variables: **name**, **surname** and **age**.

Right-click on *TutorialWeek02* package and select New -> Java Class.

Remember that **any class must be entirely defined in a single source file**.



Click on Next, name the class "Person", and leave all the other fields as they are. Click on the "Finish" button when you are done.



3) Let's make the Person class do something: create a *constructor* that takes a String and sets the name to be that String. Also, write a method called *displayName*(), which will print the name on the screen.

```
public class Person {
    private String name;
    private String surname;
    private int age;

    /* Constructor - Create a new instance of Person */
    public Person(String n){
        name = n;
    }

    // display the name
    public void displayName(){
        System.out.println(name);
    }
}
```

4) Let's go back to our *TutorialWeek02* class. Since we now have the Person class, let's make some use out of it. In the "main" method of our *TutorialWeek02* class, create a Person named "Ben".

```
public class TutorialWeek02 {

    public static void main(String[] args) {
        System.out.println("Hello World!");

        Person p = new Person("Ben");
        p.displayName();
    }
}
```

5) Write for the class Person other two methods that set the surname and the age:
- void setSurname(String s);
- void setAge(int num);

6) Implement also accessory methods to return the surname and the age.
- String getSurname();
- int getAge();

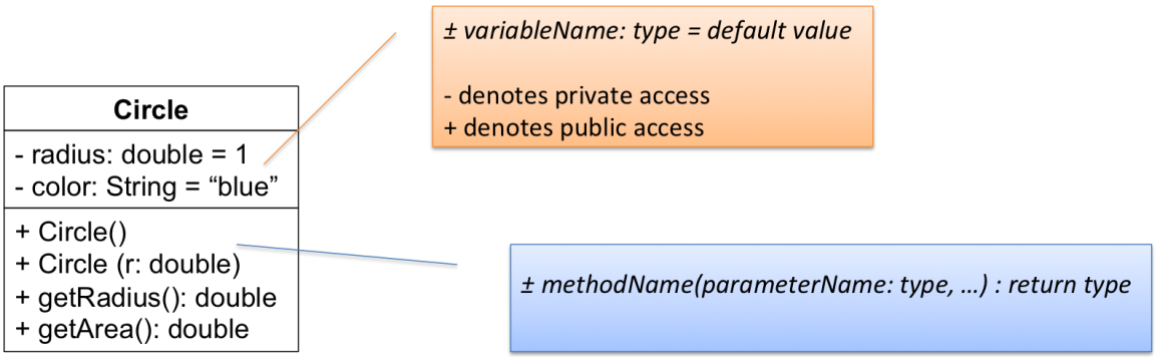
7) In the main() on *TutorialWeek02* class create other three persons and for all of them print on the screen their name , surname and age.

Classes and Instances

8) In this tutorial we want to implement a class called *Circle*. Let's create our project in NetBeans:

- Start NetBeans IDE. Choose File -> NewProject.
- As explained in the previous exercise, in the New Project wizard, expand the Java category and select Java Application. Then click Next.
- Give a name to your project, for example *TutorialCircle*.
- Add the class *Circle*: Right click on the package icon under Source Packages, (on the left side of your IDE), and select New -> Java Class.
- Name the class *Circle* and leave all the other fields. Click on the "Finish" button when you are done.

9) The class Circle is designed as shown in the following UML class diagram:



As you can see from the UML class diagram, the class Circle contains:

- Two private **instance variables**: radius of type double and colour of type String (with a default value of 1 and "blue" respectively).
- Two overloaded **constructors**: a default constructor with no argument, and another that takes an argument of type double for the radius
- Two public **methods** that return the radius and the area respectively, for this instance.

Implement the class Circle based on the class diagram above. **Try to implement by yourself** and then check with the code provided below:

```
public class Circle {
    // private instance variables, not accessible from outside the class
    private String colour;
    private double radius;

    // default constructor with no arguments
    public Circle(){
        radius = 1;
        colour = "Blue";
    }

    // second constructor takes as argument the radius but has default colour
    public Circle(double r){

       radius = r;
        colour = "Blue";
    }

    // public method to retrieve the radius
    public double getRadius(){
        return radius;
    }
```

```
    // public method to compute and return the area of circle
    public double getArea(){
        return radius*radius*Math.PI;
    }
}
```

10) Compile the class Circle. Can you run the Circle class?
Note: Circle does not have a main() but is a building block that you can use in another program!

We cannot run a class, but we can compile it. A class is a building block, Circle does not have main()! You will not see any output in the console if you run the file!

You can instantiate an object of the class in another program!

11) Implement a program to test the class Circle. In order to test the class we need to create some instances of this class. We create the instances in the main() method, which is the entry point of your program.
So, go to the file where there is the main(). If you follow the tutorial, it should be in TutorialCircle.java.

```
public class TutorialCircle {
    public static void main(String[] args) {
        // Declare an istance of Circle class called circle1
        // Invoke the default constructor
        Circle circle1 = new Circle();

        // invoke public methods
        System.out.println("The circle has radius of " + circle1.getRadius() + " and area of " +
       circle1.getArea());

        //declare an instance of Circle class called circle2
        // invoke the second constructor
        Circle circle2 = new Circle(8);

        // invoke public methods
        System.out.println("The circle has radius of " + circle2.getRadius() + " and area of " +
       circle2.getArea());

    }
}
```

12) Run the class *TutorialCircle* and using breakpoints and debugging the code, study how the program works and the results that you get.

Private and Public

13) In *TutorialCircle*, can you access the instance variable radius or assign a new value to it directly?
Try the following line of code and explain why you get an error message:

```
System.out.println(circle1.radius);

circle1.radius = 10;
```

***Radius* is private variable you cannot access directly from your main!! You need to use get and set methods! The right way to do the previous line of code is:**

```
System.out.println(circle1.getRadius());

circle1.setRadius(10);
```

Constructors

14) Add a third constructor in the class *Circle*. This new constructor will take two arguments:
- a double for the radius
- a String for the colour

```
public Circle(double r, String c){
      … // code for the new constructor
}
```

15) Modify the *TutorialCircle* code in order to instantiate a new object using the third constructor that you just implemented.

Get methods

16) In the class *Circle* add a method to retrieve the colour

```
public String getColor(){
      …// code for the method
}
```

17) Modify *TutorialCircle* in order to print on the screen the colour of the circles you instantiated.

Set methods

18) *Are you able to set a different value of the radius and colour once the object has been instantiated in TestCircle*?

As we said earlier, these variables are private, and you cannot modify them from outside the Circle class! For this reason, you can implement two public setter methods to change these values after the circle is constructed:

```
public void setRadius(double newRadius){
   radius = newRadius;
}

public void setColour(String newColour){
   … // code for the method
}
```

19) Modify *TutorialCircle* in order to use the methods you implemented in 11). After the circle is constructed you will change the color and the radius and you will print the new values on the console.

20) Can you run the following?

```
System.out.println(circle1.setRadius(20));
```

No, you can't. setRadius does not have any return type! (The return type is void). setRadius is used to modify the instance variable radius not to return it!

This

This is a keyword in Java. It can be used inside the Method or constructor of a Class. ***This* works as a reference to the current Object** whose method or constructor is being invoked. *This* keyword can be used to refer to any member of the current object from within an instance Method or a constructor.

For instance, when we construct a Circle we passed the variable r (r for radius) as argument. Although we prefer to call it radius (it would be more understandable) and we can use the keyword this to resolve conflicts. For example:

```
public Circle(double radius){
 this.radius = radius;
      …
    }
```

21) Modify all the constructors and the setter methods in the class Circle in order to use the keyword *this*.

toString()

It is good coding practice to add a public method to your class, which returns a String with a short description of the object that you instantiated. You can call this method toString(). In our example, it could return the radius and the colour.

22) Implement the method `public String toString()` for the class `Circle` and call this method from *TutorialCircle* in order to print the description on the console.

    Note: `toString()` is called automatically when you pass the instance of the object to `println()`

    Try both solutions:

    ```
    System.out.println(circle3.toSring());
    System.out.println(circle3);
    ```

    Is there any difference? **No, there isn't. With both solutions, we print on the screen a description of circle3 object.**

23) Modify the class diagram in 2) adding all the methods that are now in the class Circle