BSM 261: Object Oriented Programming
LAB 5: Classes and Constructors

November 3/4, 2022

Date Performed:	November 3/4, 2022
Instructor:	Orhan Özgüner
Teaching Assistant:	Yunus Emre Avci

1 Class Material, Lab&Quiz Scores

You can access to the class material from the following link:
https://drive.google.com/drive/folders/1yo4pvaTngQ1L2TkHuWPCzbfapLDyvwUt?usp=sharing

You can check your lab attendance and quiz scores from the following link:
https://docs.google.com/spreadsheets/d/1xKQEqMSGlj3RBo9kLh97UzbTO1fyCvaT/edit?usp=sharing&ouid=104851561367296854968&rtpof=true&sd=true

2 Getting Started

Before you perform the lab please check the following information:
• Make sure that the lab computer you are using has DrJava downloaded into the desktop. Here is the jar file to check:

drjava-beta-20190813-220051.jar

• If you can not locate DrJava, please download it from the following link and place it to your desktop.

https://sourceforge.net/projects/drjava/files/latest/download

• If you have the DrJava jar file in your desktop, you are ready to perform your lab. In order to use jar files in linux, you need to run it from terminal. Run the following commend from the terminal to start DrJava

```
java -jar  /Desktop/drjava-beta-20190813-220051.jar
```

3 Lab Rules

The purpose of the lab is to get you used to using DrJava, to start experimenting with Java, and to digest the topics you learn over the week.
Here are the lab rules:

- The teaching assistant will make a short recap of the week. Then you will perform the lab in the recitation hour.
- Your grade will be determined by your participation and finishing the assigned work during the recitation. When you get stuck on a question, please ask for help.
- Once you finish all of the tasks, show your work to the TA to receive full credit.
- **If you did not join the lab, or joined more than 10 minutes late or did not show your work to the Teaching Assistant, or did not finish all of the questions in class time you will receive 0.**

4 Lab Instructions and Questions

In this lab, you will create a couple of classes and practice: constructors, method override and method overload. Make sure to follow object oriented coding and commenting rules.

Here are some of the lecture material that you should know for the lab.

Method overloading: We can create multiple methods of the same name (including constructor methods) as long as their "parameter signature" differs. The "parameter signature" is the type, number, and order of the input values. Here are two constructors which an example for the method overloading.

```
public class Die(){
    numberOfSides = 6;
}
public class Die(int numberOfSides){
     this.numberOfSides = numberOfSides;
}
```

Method Overriding: In order to change the behavior of the inherited methods, we override them. IDE can help us determine any mistake on the override using an annotation. This particular annotation is @Override, and if we place it before a method that is overriding an inherited method, the compiler, upon seeing the annotation, will verify that we really are overriding. If we are not, it will give a compiler error instead of allowing the overload.

Constructor:
1) The first line of a constructor must be a call to another constructor. (This is also the only place in the code were we can have a constructor call.)

These are the two possibilities:
super() ← possibly with input in the parentheses. This calls the constructor of the parent class.
this() ← possibly with input in the parentheses. This calls a constructor of the same class.

If you do not explicitly have a constructor call as the first line of your constructor body, Java automatically places super(), with no input, there.

2) The constructors do the following when they are run:
a) The first line of the constructor that calls another constructor is executed. (Recall that Java adds super() if you omit this line.)
b) All fields of the instance are initialized.
c) The rest of the constructor body is executed.
Note point (b) above. Java basically takes any assignment statements on your fields and places that code after the constructor call that is the first line of your constructor and before the rest of the constructor code. This is important to remember for the situations where you care about the order that things are being done in your program.

4.1 Writing a Polygon class

Write a class called **Polygon**. This class represents the polygon shape in geometry. This class will have a single constructor which takes an **int** number of sides as input. *This being said you need to create a field for the number of sides.* In the construct,you need to assign the input value to the field. Before assigning, you need to make sure that the given number of sides needs to be checked before assigning it to the field. Assing if the number of sides is more than 2 nothing otherwise.

Polygon class will have necessary methods to retrieve and update the number of sides. **Hope you remember what this means!**

4.2 Writing a Rectangle class

From the geometry, we know that every rectangle is a polygon. You can also say that a rectangle is a four-sided polygon. Now, you will create a **Rectangle** class which extends **Polygon** class.

The rectangle class will have two constructors.
• First constructor does not take a value and sets the number of sides to 4.
• Second constructor takes two double values namely **length** and **width**. This constructor sets number of sides to 4 and assigns width and length to the fields defined in the class.

Note that 4 is not a magic number because it is always 4.

Rectangle class will have necessary methods to retrieve and update the number of sides, length, and width. **Hope you remember what this means!**

Hint: Remember, if a field already exists in the parent class, then you do not need to create a field in the class. You can simply use the fields and methods from the parent class. You only need to create the fields and methods that does not exists in the parent class.

In addition, the rectangle class has a method called **area**. This method does not take any input and returns the area of the rectangle.

4.3 Writing a Square class

Now, you will create a **Square** class. From the geometry, we know that Square is a Rectangle. Therefore, the Square extends the Rectangle class.

First of all, you do not need to create a field called side because it is either width or length. All you need to do is to make sure that the square class uses width or length when side is needed.

Now, we need to make a design choice since length and width means the same in square. Let us set the side as width and make the length always same as the width for the Square class. This means that we will need to override some of the inherited methods such as setLength() and setWidth().

The square class will have two constructors.
• First constructor does not take a value and sets the number of sides to 4.
• Second constructor takes a single input called **side**. This constructor sets number of sides to 4 and assigns width and length to side. Remember, a square is a rectangle.

Square class will have necessary methods to retrieve and update the number of sides and size. **Hope you remember what this means!**

Now, the Square class has methods getLength(), getWidth(), setLength(), and setWidth() (all inherited from Rectangle class). However, the square class has side length instead of length and width. From our design choice, we need to ensure that side is the same as length and width.

Since we have a constructor that takes a single value size, then we know that getLength() and getWidth() will return the same value. Then, all we need to do is to override the setWidth() and setLength() methods

setWidth() : From out design choice, we need to make sure that given side is set to both length and width. We can achieve this by using the setWidth() and setLength() methods from the Rectangle class. Call both setWidth() and setLength() methods with the input. This will make sure that any time width is changed, we also change the length.

setLength() : From our design choice, we know that the side is equal to width and length must be equal to the width. This means that if the user would like to change the length, then we only need to update the width. You can achieve this by calling only setWidth() method from the Rectangle class.

getSideLength() : Now, we will finalize the class by creating a getter method for the side. Remember, we made a design choice, we used one of the fields from the Rectangle class to store the side length. Therefore, we need to call the appropriate method from the Rectangle class to return the side length of the Square class.

Demonstrate your class to your lab instructor to receive full credit