BSM 261: Object Oriented Programming
LAB 2: Calling Java Methods (Functions)
Creating Java Objects

October 14, 2022

Date Performed:	October 14, 2022
Instructor:	Orhan özgüner
Teaching Assistant:	Yunus Emre Avci

1 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 the DrJava 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. Open your terminal and use the following commend to start DrJava

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

2 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, you will receive 0.**

3 Lab Instructions and Questions

You will be writing a lab report in this lab. Follow the following procedure through the lab. At the end of lab, you will show your report to the TA to receive full credit.

3.1 Playing with JFrames

We will begin the lab with the pre-defined class JFrame. The JFrame class represents a window on the screen. To use a pre-defined class, we must first load it. The JFrame class is in the javax.swing package, and so the the Java statement to load a class is

```
import javax.swing.JFrame;
```

Type that into the DrJava interactions pane now. Now that we have the JFrame class loaded, we can use it.

Question 1: Enter the following lines, and in your lab report, describe what happens after each statement.

```
import javax.swing.JFrame;
JFrame j1 = new JFrame();
j1.setVisible(true);
j1.setSize(200,400);
j1.setLocation(500,500);
```

Now try creating a few more JFrame variables. You can call the variables (almost) anything you wish. Try calling the following methods:

setVisible	getWidth
setSize	getHeight
setLocation	getX
	getY

setVisible expects a single input of type boolean,
setSize and setLocation each expect two values of type int (separate the values by a comma), and getWidth, getHeight, getX and getY do not expect any input.

Also try calling each method with a semicolon after them and without a semicolon after them. If you place a semicolon at the end of a line, it is a Java statement. A statement does not have a value. If you do not place a semicolon at the end, it is a Java expression, and expressions may return values. By omitting the semicolon, you will see any values that an expression may return.

Question 2: Answer the following questions in your lab report:
A) Consider the methods listed in Question 1. What are the ***syntactic*** differences between the methods in the left column and the methods in the right column? Syntax has to do with the actual structure of the line of code: how does the line appear when you type it in? Make a brief list of the syntactic differences you observe.

B) Consider the methods listed in Question 1. What are the ***semantic*** differences between the methods in the left column and the methods in the right column? Semantics has to do with the meaning of the code: what do the methods do to a JFrame? Make a brief list of the semantic differences you observe.

3.2 Creating, Resizing and Moving JFrames

Now let us see how much you remember from above. First, reset the Interactions pane. To reset, you need to right click on the interactions pane, from the menu, click Reset Interactions option. This erases everything you have done including all variables.

Question 3: Enter the statements you used to do the below operations in your lab report.

- Create and initialize two JFrame variables, and show (i.e. setVisible) them.
- Use the setSize method to make window 1 300x200.
- Use getWidth and getHeight methods to get the width and height of window 1 and store the results into two variables called w and h. (What type should the variables be?)
- Use setSize to make window 2 the same dimensions as window 1. *Do this without explicitly typing in the numbers 300 and 200. Instead use the variables you created above.*

Question 4: Consider the window 1 and window 2 from the question 3. Enter the two setLocation statements you used to do the below operations in your lab report.

- use the setLocation method to move the window 1 is flush against the top edge of the screen, halfway across
- use the setLocation method to move the window 2 is flush against the bottom edge of the screen, halfway across.

3.3 Calling Functions From the Math Class

The Math class is a pre-defined class that contains common mathematical functions. The Math class is in the java.lang package. All classes in java.lang are automatically loaded in every Java program. This means that we do not need to import the Math class in order to use it.

The methods of the Math class have another interesting property. We do not have to create an instance of the Math class to use the methods. This makes sense because, while the height of a JFrame "belongs" to a specific JFrame, the square root function is universal. Such methods that do not belong to an instance of a class are called **static** or **class methods**.
So, to compute 4 raised to the 5th power, we type in

Math.pow(4.0, 5.0)

Try that now. Notice that we still need to use the dot, but instead of prefixing the method name with an object reference, we prefix it with the class name. Try typing in JFrame.getWidth() and see what happens.

Another method in Math is the square root method, **sqrt** that takes a single double as input. For example, **Math.sqrt(2.0)** returns the square root of 2. Note that we can nest method calls such as **Math.sqrt(Math.pow(4.0, 2.0))** first computes the square of 4, and then takes the square root of the result, hopefully returning 4.0. You can nest method calls all you want. All that matters is that the type of the value returned by the inside method matches that type expected as input from the outside method. **Always keep track of your types!**

Question 5: Answer the following questions and type your answers in the lab report.
A) Try computing the square root of 2^2, and then computing the square of the square root of 2. Mathematically these are the same, but does Java give you the same answer? Explain why or why not.
B) Using the fact that $sin^2(x) + cos^2(x) = 1$, and using the pow and sqrt methods of Math, create an expression that computes that cosine of an angle that has the sine 0.5. (Do not use variables. Do it in one large expression of nested methods.) When you have an expression that works, add it to your lab report.