BSM 261: Object Oriented Programming
LAB 3: Creating Classes

October 21, 2022

Date Performed:	October 21, 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, or did not finish all of the questions in class time you will receive 0.**

3 Lab Instructions and Questions

In this lab you will do your first Java programming by creating an new class with four methods. This lab is different from the previous two in that you will not be writing a report. Instead, you will design a Java class and show your design at the end of lab to the Teaching Assistant to receive full credit.

3.1 Creating a Class That is a Subclass of JFrame

Create a subclass of **JFrame** called **MaxWindow**. If you remember how to do this, create the class now and skip these steps. If not, keep reading for the necessary steps.

- **Step 1:** You will type the code into the upper right window of DrJava (the text editor). The left hand window lists all the files you have open and highlights the file you are currently editing. In Java, every public class is placed in its own file so be certain to start typing your code into a new file and not into a file that already contains other data.

- **Step 2:** The first step to create the class is to place a comment at the top of the file that contains your name. There are two types of comments. Single line comments begin with `//`:
  ```
  // This is a comment.  The entire comment must be on this line.
  ```

  Multiline comments are bracketed by /* and */:
  ```
  /* This is a multiline comment.
  The comment continues for as many lines as you want
  until the closing star-slash is encountered.
  */
  ```
  Write a multiline comment with your name at the top of the file.

- **Step 3:** To create the subclass, the code for your class should go below the comment, and it will start with
  ```
  public class MaxWindow extends JFrame {
  ```

  Be certain to add the line: `import javax.swing.JFrame;` to the top of your file, above the class definition so you can use the JFrame class, but below the comment with your name.

- **Step 4:** Add a closing curly brace } to the end of your file, and save the file as **MaxWindow.java**.

  Important: In Java, every public class must go into a file with the same name as the class and ending with .java. You should also start your class name with a capital letter. This is not required in Java, but it is the style that has become the industry standard.

- **Step 5:** Now compile your code by selecting Compile. If the code has no mistakes, move on to the next step. If it does, fix the errors, save, and recompile before continuing.

3.2 Creating Two Methods to Move the Window

Create the following two methods and place a comment above each method.

Method 1: Write a void method (with a comment above it) called **snapToTop** that moves the window to the top of the screen, but does not change it's horizontal location. If you remember how to do this, do so and move to the next method, if you do not, here are further instructions.
The method should begin:

```
public void snapToTop() {
```

Each method definition in a class goes between the opening curly brace and the close curly brace of the class definition. *The code should be indented. The code you write inside should also be indented*

and aligned so that each statement inside a compound statement starts on the same column.

You will need to call some methods that your class inherits from JFrame (setLocation, getX, getY). You used these methods in the last lab. You can look at that lab for a hint. **Hint: Recall that the syntax for a method call is**

```
object-reference.methodname(argument list)
```

Your snapToTop method will be run for a specific window (object), and you need to be able to call that window's methods from inside snapToTop. Java provides a reference called **this** that is used inside a method to get the reference of the object running that method. So, to get the current horizontal location of your window, from inside **snapToTop**, use **this.getX()**. Once you complete the method, save your code and compile it. If it has no errors, test your code:

Create a new **MaxWindow** in the Interactions pane, move that window to somewhere other than the top, and call the **snapToTop** method. If correct, the window should now jump to the top of the screen without changing in size or moving left or right.

Place a comment above your method explaining what the method does. You should indent your comment so that it starts in the same column as the method.

Method 2: Create a method **snapToLeft** that moves the window to the left edge of the screen without changing its vertical location or its size, and put a comment above it. The technique is almost exactly the same as the **snapToTop** method.

3.3 A Short Pause in Coding: Getting The Screen Size

In last week's lab, you had to move a window to the edge of the screen. This was challenging because you probably did not know the screen size of your monitor. Java has pre-defined classes that can provide you with the screen size. You will use them in this lab to help you with your tasks.

Toolkit is a pre-defined class that is used to manipulate the computer's desktop. One of the useful methods of Toolkit is **getScreenSize** which returns an object of type **Dimension**. Dimension represents a 2-dimensional coordinate.

Before you code with these classes, you should practice using them. Type the following code into the Interactions pane (not in the editor). Of course, before you can use the classes, you have to import them!

```
import java.awt.Dimension;
import java.awt.Toolkit;
```

Now you can use these classes:

```
Toolkit kit = Toolkit.getDefaultToolkit();
Dimension d = kit.getScreenSize();
```

The first line calls a method of the **Toolkit** class that creates and returns a Toolkit object of the proper type for your system, and this object (or really its location in memory) is stored in the variable called kit. The second line calls the **getScreenSize** method of Toolkit on the Toolkit object and stores the reference to the returned Dimension in a variable called **d**.

Now you can access the values of the Dimension through either its methods or its fields. Type the following into the interactions pane. Remember to omit the semicolons so that you can see the values

of these expressions.

```
d.getWidth()
d.getHeight()
d.width
d.height
```

So, you can use either the methods or the fields of Dimension to get the screen size. Note that the methods have a return type of double while the type of the fields is int. It does not matter which you use, but depending on what you decide you may need to use a typecast.

3.4 Creating Methods that Use the Screen Size

Method 3: Add a void method to the **MaxWindow** class called **maximizeHeight**. The method will move the window to the top of the screen and make it as tall as the screen.
The method should begin:

```
public void maximizeHeight() {
```

Your method will need to use the Toolkit and Dimension classes described above. First, you need to import the classes

```
import java.awt.Dimension;
import java.awt.Toolkit;
```

Now, you will add code from above to your method to get the screen height and width. Note, you do not add all the code you typed into the Interactions pane, just what is needed to get the required screen dimension.

Save your code, compile it, and fix any errors. Test your code in the interactions pane. Create a MaxWindow instance and move it to somewhere other than the upper left corner. Call the **maximizeHeight** method, and if correct, the window should now stretch from the top to the bottom of the screen without changing in width or moving left or right.

Place a comment above your method explaining what the method does. You should indent your comment so that it starts in the same column as the method.

Method 4: Add the void method **maximizeWidth** to the class that moves the window to the left of the screen and makes it as wide as the screen. Save, compile, and test your class.

Demonstrate your class to your lab instructor to receive full credit