CS 1110, LAB 2: WORKING WITH THE JAVA API

Name: ______________________________ **Net-ID**: _______________

There is an online version of these instructions at

http://www.cs.cornell.edu/courses/cs1110/2012sp/labs/lab02.php

You may wish to use that version of the instructions.

The purpose of this lab is to get you comfortable with using the classes and objects that are already built into Java. This also means familiarizing yourself with the Java API. Remember that you can access the Java API from the course website, or by going here:

http://docs.oracle.com/javase/6/docs/api/

Turning in the Lab. Labs are graded on effort, not correctness. When you are finished, you should show your written answers to this lab to your lab instructor, who will record that you did it. You do not actually need to turn in this piece of paper; it is yours to keep. If you do not finish during the lab, you have **until the beginning of lab next week to finish it**. However, you should always do your best to finish during lab hours.

Because the lab is graded on effort, you should focus all of your efforts on ***understanding*** and not just getting the answers correct. The primary purpose of thes labs is for you to have a hands-on session during which the consultants and instructors can help you out.

---

1. Variables, Declarations, and Assignment Statements

The format of this first part of the lab is very similar to the last one. For each expression or statement, make a *guess* at what you expect the result to be. If you have no idea, you should write "?". You should then enter the expression or statement into DrJava and compare the result. If the two results are different, you should try to figure out why DrJava gave the answer that it did. Come up with a reasonable explanation and put it in the final column.

It is important that you know the difference between a declaration and an assignment statement. A declaration like

boolean b;

simply says that in the rest of the "program", a variable named j may be used and its type is **boolean**. On the other hand an assignment like

$b = 3 < 5$;

is a command to do something; it evaluates the expression $3 < 5$ and stores its value in variable b. Moreover, because the assignment statement is not an expression, DrJava will not actually output a result when you type something in (provided you remembered the semicolon). It will just perform the command silently.

Enter the expressions or statements in the table below in **exactly the order that they are given**. Otherwise, you may get different results from everyone else.

Statement or Expression	**Expected Value**	**Calculated Value**	**Reason for Calculated Value**
int i;	None		
i			
i = 2;	None		
i			
int j;	None		
j = j + 9;	None		
j			
k = 5;	None		
j + k			
double w = i + j;	None		
w			

2. Class Math

As promised, here is another lab topic covering stuff we have not (yet) talked about in class. We saw a little bit of the class Math in the last lab, and we are going to revisit it here. Math is special because it has *static methods*. A static method is one that you call on the class, not the object (e.g. the file drawer, not the folder). So you never use new to create a Math object, you just call the methods on the class directly. For example, we already saw the expression Math.ceil(25.6) in the previous lab.

Static methods are used for functions where we do not need any information from a specific manilla folder to get the result. As it does not make sense to have different math manilla folders that have different math functions, these functions are made static.

in addition, Math has *static fields*; these are fields you can call directly without getter methods. For example, Math.PI is actually a field, and does not use parentheses.

2.1. **Finding the API for Math.** The class Math has a lot of methods. Instead of listing them all here, it is best that you discover them for yourself in the Java API specification. Find the root of the Java API from the website for CS1110. Open the API specifications for class Math in package java.lang. Alternatiely, simply copy and paste this URL into your browser:

`http://docs.oracle.com/javase/6/docs/api/java/lang/Math.html`

Read a bit to familiarize yourself with the page. Look at the methods in the table below and see if you can find them on this page. You might find it easiest to take advantage of the search capabilities in your browser.

2.2. **Evaluating Math Expressions.** Fill out the table below, using the same approach that you took in the previous part of the lab.

Expression	**Expected Value**	**Calculated Value**	**Reason for Calculated Value**
Math.min(-7, 4)			
Math.min(Math.min(3,4),5)			
Math.sqrt(5)			
Math.sqrt(-5)			
Math.floor(-3.7)			
Math.ceil(3.7)			
Math.ceil(-3.7)			
Math.abs(3.7)			
Math.abs(-3.7)			
Math.abs(-3)			
Math.PI			

3. Experimenting with JFrame

In this next part of the lab, you will play around with JFrame, one of the classes found in the Java API. JFrame is used for making GUI-based programs. While we have not covered anywhere near enough to do these types of applications, we can still play around with JFrame like a toy.

In the DrJava Interactions pane, create a JFrame object, storing its name in variable jf1 (which you may need to declare), and show it. For the creation and storage part, either use the assignment statement

```
javax.swing.JFrame jf1 = new javax.swing.JFrame();
```

or use the two-statement sequence below, which employs an import statement (see also Sec. 1.3.2 of the text):

```
import javax.swing.*;
JFrame jf1= new JFrame();
```

In addition, you will need to call the method `show()` to get it to appear on the screen. Do that now.

Once you have done that create and show a second JFrame object using a different variable jf2. Check that you can drag both windows around to different places.

¡hr width="50

3.1. **JFrame Methods.**

Execute calls for the eight methods shown in the table below. The three dots ("...") denote an argument that you have to fill in when you type the call into the Interactions pane. For each method, below denote whether it is a function or a procedure. Recall that a function returns a value, while a procedure instructs Java to do something.

Once you have done that, briefly write the results of the method call. If it is a function, write the value that is returned. If it is a procedure, write what you think Java did when you called the procedure.

Method	**Procedure or Function**	**Result When Called**
show()		
getWidth()		
getHeight()		
setSize(. . . , . . .)		
setLocation(. . . , . . .)		
getX()		
getY()		
setTitle(. . .)		

3.2. **Positioning JFrames.** Reset the Interactions pane with the "reset" button. Create two new JFrame objects, assigning their names to variables. Use method calls to make the first window 120 x 220 pixels and the second window 220 x 120 pixels. Which coordinate comes first, the horizontal or the vertical one?

Check your answer by calling methods getWidth and getHeight of one of the JFrames.

Next, drag the first window so that it is on the left of the screen and halfway down. Use the method calls to determine its x-coordinate and y-coordinate, and write the answer here:

Is there something unusual about how screen coordinates work? What do you notice about the difference in coordinates betwee the two windows?

3.3. **Resizing a JFrame.** Create a new JFrame, storing its name in a variable jf, and show it. Try resizing the JFrame with your mouse to make it bigger. Call function isResizable() of jf to see whether JFrame jf is resizable. What is the answer?

Now execute the procedure call jf.setResizable(false); Try resizing jf with your mouse. Is it resizeable?

Finally, call the function isResizable() in jf and tell us the results: