CoSc 10403
Lab # 5 (The Controller)

Due Date:
Part I, Experiment – classtime, Tuesday, March 24th, 2020

Part II, Program - by midnight, Tuesday, March 24th, 2020

Part I MUST be typed and submitted to your instructor at the BEGINNING of class on the due date. IT WILL NOT BE ACCEPTED LATE. *Failure to print, left it in your dorm room, etc. are unacceptable excuses.*
Part II is the programming component. It should be submitted as a zipped file with two classes in the zip file: **Lab3.java** and **Lab5.java** using **D2L.** Submit the **Lab5.zip** file for grading.
You should be aware that there is a Java reference readily available on the web (https://docs.oracle.com/javase/7/docs/api/) that can be used to find information on all Java packages, classes, methods, etc. Some questions in Part I are designed to help familiarize you with this web site.

Part I. Experiment. (20%).
Use the Java code provided in the files "Lab5ExperimentControl.java" and "Lab5ExperimentView.java" file. Put both files in your project's **src** folder to run the experiment. (Follow the same procedure that you used to create your past labs). Modify the width and height parameters as required in order to see all components (I recommend width=500 height=200)
You will experiment with this code in answering some of the questions in this section. Testing this code will allow you to become familiar with the use of the **extends** clause in java by extending a ***view*** class into a ***control*** class. Also you will get practice using *accessor methods* to obtain information from **JComboBox**, **JLists**, **JRadioButton**s and **JCheckBox**s along with the use of **ActionListener**s. Finally you will learn to add and play audioClips in java. For this you will need to add a subfolder **sounds** within your **project folder** as you did with **images** and put in it an audio clip that we are providing (i.e. **oh.wav**).

You will experiment with this code in answering some of the questions in this section.
Testing these code will allow to get familiar with the use of JFrame a container not discussed in your textbook, that allows you to create a pop up window. Also you will get a practice on the getMethdods to obtain information from JComboBox, JLists,, JRadioButton and JCheckBox. Finally you will be able to se how an ActionListeners works.

REMEMBER - ALL EXPERIMENTS MUST BE TYPED - NOT HANDWRITTEN!!!!
Navigate to the Oracle Java site whose URL is shown above. In the upper left-hand pane are listed the classes that are included in the standard Java Platform. Scroll down to find the **javax.swing** class. In the javax.swing classes that are now presented to you in the lower left-hand pane, scroll to find the **JComboBox, JList** and **JCheckBox** classes and click on it.
Answer the next questions using information derived from this page.
1. By referencing the javasoft web site, determine how you get the value of a given selection in the **JComboBox**?

2. By referencing the javasoft web site, determine how you get the value of a given selection in the **JList**?
3. How do you check if the **JCheckBox** has been selected?
4. In examining the methods defined in the **JButton** class the method **addActionListener()** is not present, yet have the ability to use it that with method calls. How?
For questions 5-9 perform each of the tasks described. (You should modify the Lab5Experiment code and run the program to verify your answers; after each question restore the code to its original form to continue)
5. What happens in the *Lab5ExperimentCode* if you remove the line *import java.awt.event.*;* from the initial imports
6. What happens in the *Lab5ExperimentCode* if you remove the clause *implements ActionListener* from the initial line of the class
6. What happens in the *Lab5ExperimentCode* in the *init()* method you remove the lines *cancel.addActionListener(this);* and *display.addActionListener(this);*

7. In the actionPerformed(ActionEvent e) method change the lines

```
String whichButton = e.getActionCommand();
if (whichButton.equals("DISPLAY")) displayData();
if (whichButton.equals("CANCEL")) clearAll();
```

For the lines

```
Object source = e.getSource(); // another way to select button
if (source.equals(display)) displayData();
if (source.equals(cancel)) clearAll();
```

What is the performance of the program?

8. What happens in the *Lab5ExperimentControl* in the **constructor** method if you remove the lines:
cancel.addActionListener(this); and **display.addActionListener(this);**

9. What happens in the *Lab5ExperimentControl* in the method **displayMessage()** if you change the line: **ohSound.play( );** to **ohSound.loop( );**

10. In the **actionPerformed(ActionEvent e)** method exchange the lines:

```
String whichButton = e.getActionCommand();

if (whichButton.equals("Display")) displayMessage();
if (whichButton.equals("Cancel"))  clearMessage();
```

For the lines:

```
Object source = e.getSource();

 if (source.equals(display)) displayMessage();
```

if (source.equals(cancel)) clearMessage();

11. Answer the following questions regarding the programming design of this code:
a) What does the various calls to the method **System.out.println()** do?
b) What do you think is the purpose of adding that statement?
c) In the method **displayMessage()**, in the line
course = (String)myList.getSelectedValue(); what does the clause **(String)** do?
d) In the method **clearMessage()** what is the purpose of the line
myList.setSelectedIndex(0);

Part II. Programming. (80%)

In this next lab you will give some interactivity to the Solar System Selector. For this you will extend the View given by lab 3 and provide the Control part in this lab. Use only ActionListeners along with a getSelectedIndex() method from the list to select the planetin the JList. When you use the Select JButton the image should appear also the p,ant surface fact should be selected in the fact JComboBox and providing the corresponding surface for each planet. We took the data from the NASA site and it is the following:

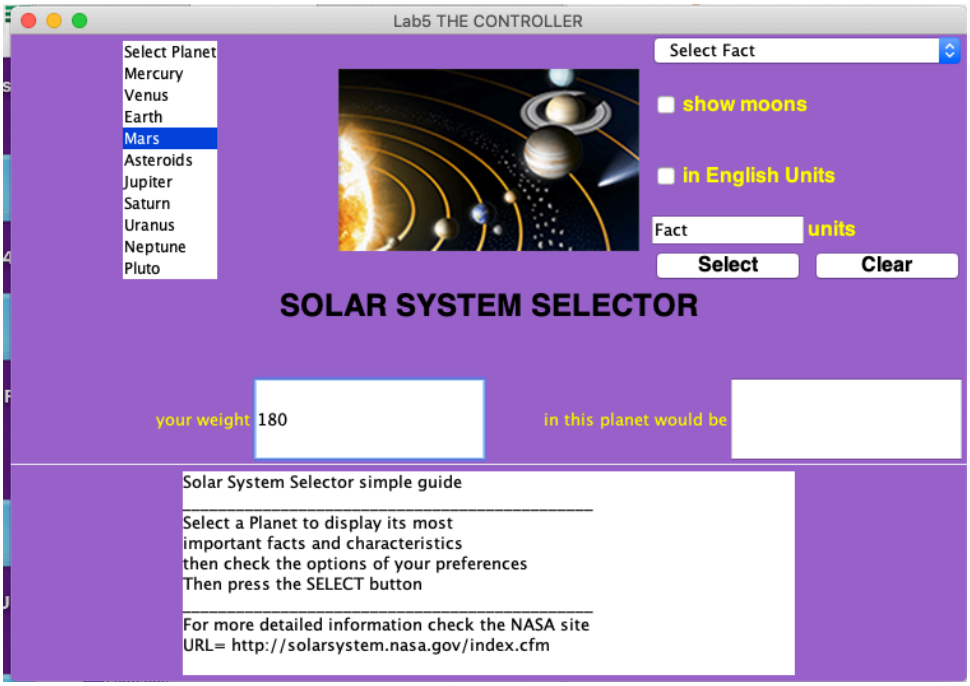
Mercury	Gravity 0.38 Area 74,797,000.00 $km^{2.}$
Venus	Gravity 0.91 Area 460,234,317.00 km^2
Earth	Gravity 1.00 Area 510,064,472.99 km^2
Mars	Gravity 0.38 Area 144,371,391.00 km^2
Asteroid Ring (consider asteroid Ceres)	Gravity 0.03 Area 2,849,631.00 km^2
Jupiter	Gravity 2.53 Area 61,418,738,571.00 km^2
Saturn	Gravity 1.07 Area 42,612,133,285.00 km^2
Uranus	Gravity 0.91 Area 808,3079,690.00 km^2
Neptune	Gravity 1.14 Area 7,618,272,763.00 km^2
Pluto	Gravity 0.07 Area 16,647,940.00 km^2

Note that this numbers must be enter as double since the Integer may be limited

Finally use the earth JTextFields to enter your weight and use this number to compute the corresponding weight in the selected planet. For the time being do not be concerned with selecting moon view or the unit display that will be left for the next lab.

Your should illustrate the use of **ALL** common widgets discussed in class as well as your skills to show images within the widgets. You will be graded on its appearance, the appropriate use of widgets and its correct functionality. Please feel free to be creative, here we provide our applet demo so you can get the look and feel we expect from your own implementation.

It looks like, BEFORE the Select Button is pressed but the planet in the JList has been selected and the weight is entered



It looks like, AFTER the Select Button is pressed