CoSc 10403

Lab # 7 (Full MVC using a Model)

Due Date:

Part I, Experiment – classtime, Tuesday, April 28th 2020.

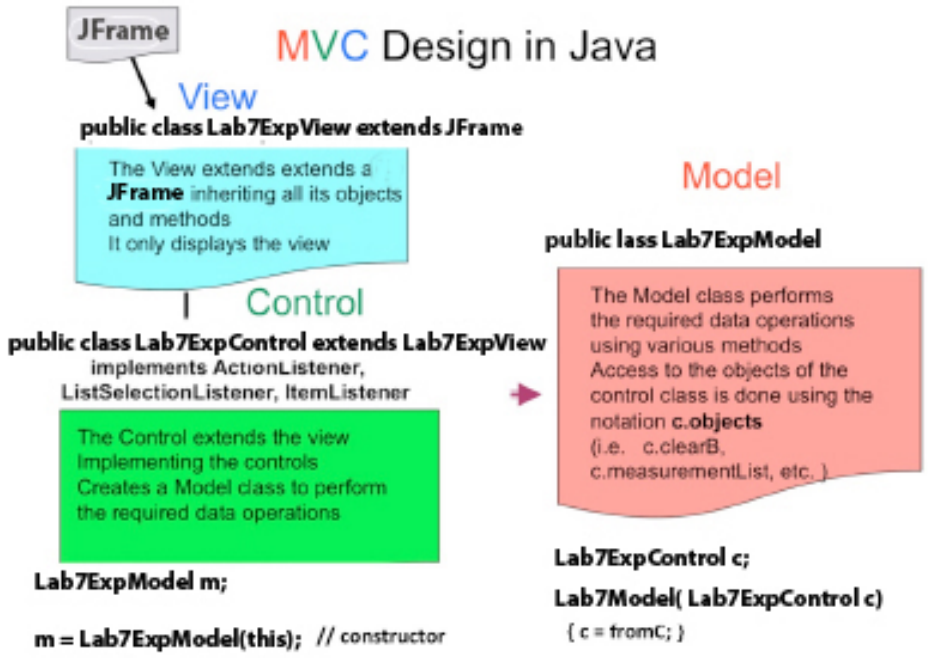
Part II, Program - by midnight, Wednesday, May 6th 2020.

Again you will be required to "zip" together the three .java files (Lab3.java, Lab7.java, Lab7Model.java) into one archive file (Lab7.zip) and submit only the resulting Lab7.zip file. Submit the Lab7`.zip` file for grading using D2L. This procedure is the same as was required for your Lab5. Of course, you should also ALWAYS verify that your lab can be executed correctly.

Submit Lab7 as done for previous labs. Again, Part I is the Experiment component and will not be accepted late. *At class time on the due date, you will be required to turn in a copy of the typed experiment questions and your typed answers.*

You should be aware that there is a Java reference readily available on the web; as you know 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.

The purpose of this lab is to familiarize you with the complete **`MVC model,`** take a look at the following diagram to see how you can set up three files you will need (**`Lab7ExpView.java`**, **`Lab7ExpControl.java`** and **`Lab7ExpModel.java`**, we will discuss in the classroom the architecture of the **`MVC`** model). Beware, that this an initial approach to this design concept, there are many ways to implement it.



Part I. Experiment. (30%).

Create a **Lab7Project** project using the **Lab7ExpView.java**, **Lab7ExpControl.java**, and **Lab7ExpModel.java** files provided along with the **sounds folder**. Run the **Lab6ExpControl.java** and play with it. Observe its operation and note that the **Lab7ExpControl** actually extends the **Lab7ExpView** not the **JFrame** and so it must be present in the same folder; also note that the **Lab7ExpControl** creates a new class m from the **Lab7ExpModel.java** class also present in this same folder. To do this it constructs the class by the statement:

```
public Lab7ExpModel m = new Lab7ExpModel(this);
```

Beware, that this an initial approach to this design concept, there are many ways to implement it. Also note that it is in the model class, that all the required methods to perform the activities of the experiment are available. Once you have been familiarized with the workings of these three classes we want you to try to make typing errors and then catch them using the **try/catch** statements in java. After you have a good understanding of it answer the following questions:

1. Modify **Lab7ExpView.java** by changing the line

   ```
   public JButton bCompute = new JButton("SQRT");
   ```

   to **private JButton bCompute = new JButton("SQRT");**

   now check the code of **Lab7ExpControl.java.** What do you get from eclipse? Why?

2. Make the **JButton public** again and now modify in **Lab6ExpControl.java** the line

   ```
   else if (e.getSource() == bStop)  m.pStop();
   ```

   to **else if (e.getSource() == bStop) pStop();**

   now check the code. What do you get from eclipse? Why?

3. Restore the code to its initial state. What is the purpose of the line

   ```
   public Lab7ExpModel m = new Lab7ExpModel(this);  ?
   ```

4. Look at **Lab7ExpModel.java** and explain the purpose of the instruction in the constructor: **c = fromC; ?**

5. In the Model file look at the method **pCompute**. What is the class type of its input parameter of the What is the class type of its return value?

6. When **Lab7ExpControl** calls the **pCompute** method. Where is the result stored and What is the purpose of using the **decimal.format()** method?

7. Back in the **Lab7ExpModel.java** what does the method **c.sn.loop()** do? And where is the sound object located in the model or in the control? What does the notation **c.sn** tell you?

8. In the **Lab7ExpModel.java** file what is the purpose of the **try/catch** statements?

9. How does the **Lab7ExpControl.java** file report an error to the user?

Part II. Programming. (70%)

Note. Using the MVC approach for this lab you will need to submit three classes: a *view*, a *controller* and a *model*. Specifically you will use **Lab3** as your *view*. You will use a portion of your **Lab5** code as your *control* but you will have to add new listeners to complete the **Lab7** requirement. *You can **copy-and-paste** that portion of you **Lab5** that was created to handle the user events (i.e., the **actionPerformed** method and any of the user-defined methods that are associated with your event handling).* For **Lab7** you will have to write a **Lab7.java** program that will use at least three different type of listeners: **JListListener** when selecting a planet, **ItemListener** when selecting the **JComboBox** or and **JCheckBoxes** to show moons or the English unit display then you will include the **ActionListener** for the **JButtons** to compute the weight or to reset the whole applet (***note** – this is the code you are obtaining from your Lab5).*

The important part here is that your **Lab7** *(the controller)* will extend the **Lab3** *(the view)* and well create a new object as an instance of the class **Lab7MD** *(the model)* that will perform the various actions required in **Lab7**. Remember **Lab7** extends **Lab3** which is the *view* and so it has access to all the methods and objects in Lab3.

The desired interaction of the lab is as described below.

1) By selecting a planet from the JList you change the image generating a sound *(here use an JListListener).*
2) By selecting a fact from the JComboBox the data for the selecting plant should display with the proper unit in the JTextField and JLabels available for that *(here use the ItemListener)* also generating a sound.
3) If the Show Moons JCheckBox is selected then the image displayed should be the one of the moons and not the planet *(use again the ItemListener).*
4) If the English Units JCheckBox is selected the data should be converted to the appropriate values accordingly *(use again the ItemListener).*
5) Finally you enter you weight and when pressing the Select Button the calculation of you weight in the selected planet should be provided *(use the ActionListener).*

You can use the NASA site to gather important fact about the various planets used here, note that in the case of the Asteroid Belt we have used Ceres to provide you with its various facts. The website of NASA is located at **http://solarsystem.nasa.gov/planets/solarsystem**

Here we provide you with the data we used in our implementation:

Planet	Discovered	Orbit	Orbit Velocity	Density	Surface Area Km^2	Gravity	Effective Temp	Elements	Relative Gravity to Ea
mercury	Known by Ancients	57,909,227.00	170,503.00	5.43	74,797,000.00	3.70	-173/427	N,H,Na	0.38
venus	Known by Ancients	108,209,475.00	126,074.00	5.24	460,234,317.00	8.87	462.00	N,CO2	0.91
earth	Known by Ancients	149,598,262.00	107,218.00	5.51	510,064,472.00	9.80	-88/58	C,H,O,N	1
mars	Known by Ancients	227,943,824.00	86,677.00	3.93	144,371,391.00	3.71	-153/20	N,CO2,Ar	0.38
asteroids	January 1,1801	413,690,250.00	64,360.00	2.09	2,849,631.00	0.27	-227/70	H2O,O,N	0.376
jupiter	Known by Ancients	778,340,821.00	47,002.00	1.33	61,418,738,571.00	24.79	-148	H, Helium	2.53
saturn	Known by Ancients	1,426,666,422.00	34,701.00	0.69	42,612,133,285.00	10.40	-178	H,Helium	1.07
uranus	March 1,1781	2,870,658,186.00	24,477.00	1.27	8,083,079,690.00	8.87	-216	H, He, Methane	0.91
neptune	September23, 1846	4,498,396,441.00	19,566.00	1.64	7,618,272,763.00	11.15	-214	H, He, Helium	1.14
pluto	February 18,1930	5,906,440,628.00	16,809.00	2.05	94,361,927.32	0.66	-233/-369	N,CO2,Argon	0.185
Units	Year	Km	Km/hour	g/cm^3	Km^2	m/s^2	°Centrigrade		

Source: http://solarsystem.nasa.gov/planets/solarsystem

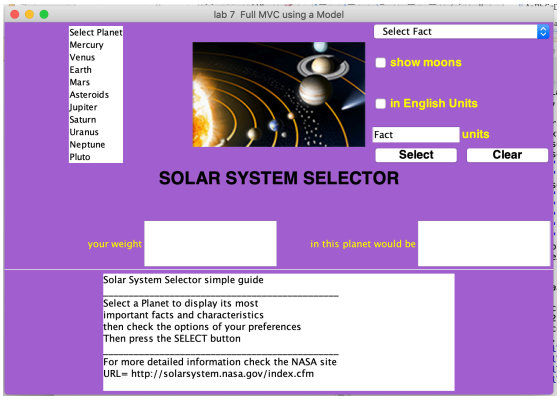
As in Lab5 note that the use of double and long might be required for some of the facts, you can also use the decimalformat Error validation must be provided so that only numeric values appear in the JTextFields, this is done using exception handlers as described in the classroom. The Clear Button should reset everything.

For the system to work properly **`Lab7.class`** (the controller) is called by the browser; the other two classes (**`Lab3.class`** and **`Lab7MD.class`**) as well as the images and sounds folders must located in the **`bin`** folder. PLEASE NOTE that this time using **`Turnin`** you will submit a zip file containing the (*View*) **`Lab3.java`**, (*Control*) **`Lab7.java`**, and the (*Model*) **`Lab7MD.java`** files. 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.

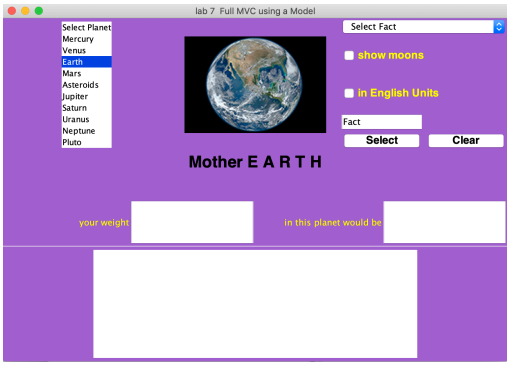
When you create the Lab7MD.java file you will need to include a constructor method with the same name and having as parameter a link to the controller class, i.e. Lab7 just as shown in the experiment code for this lab.

Remember -- submit ONLY the three *.java* files **(this time as a zip file)** code using D2L and **ALWAYS** verify that your Lab7 executes correctly!

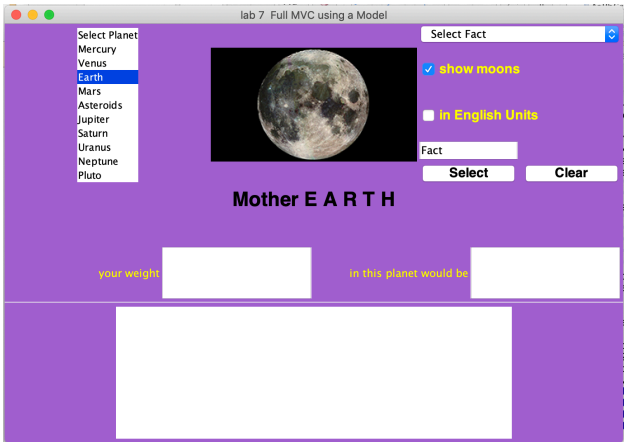
`At the beginning`



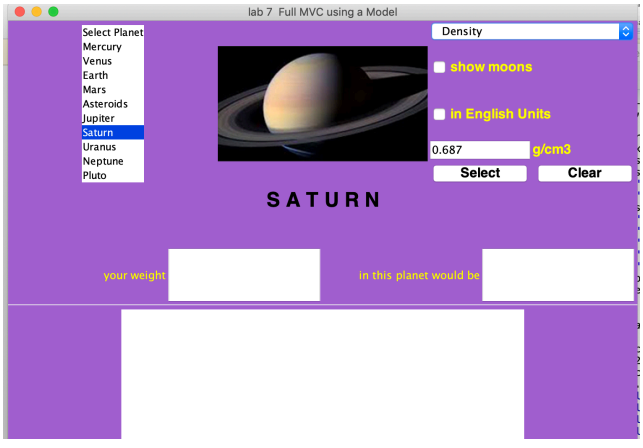
`When selecting a planet`



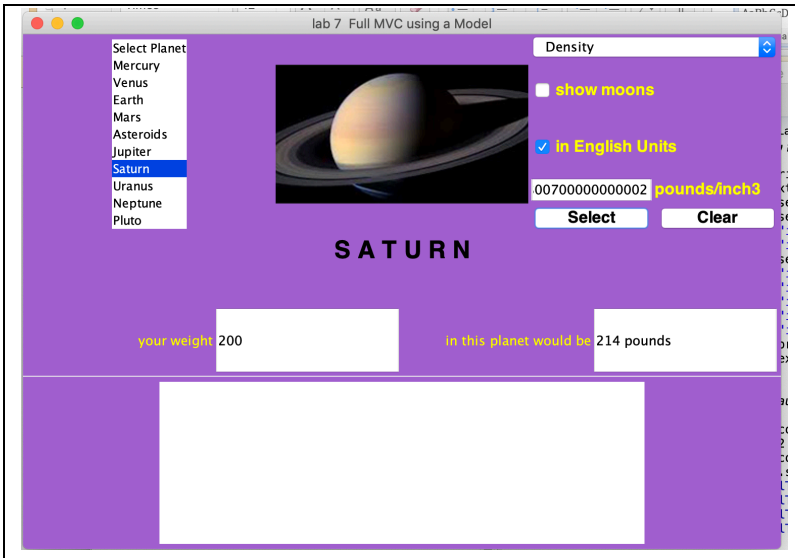
`When choosing the moons`



`When selecting a fact`



`When selecting English Unit with Button`



Note it is very important to realize that the **images folder** in this case has many **subfolders,** this because there are many images and they are properly organized, **do keep this organization** when you **read your images!**