ICOM 4015-Advanced Programming
Spring 2014
Instructor: Dr. Amir H. Chinaei
TAs: Hector Franqui, Jose Garcia, and Antonio Tapia

Reference: **Big Java**
By Hortsmann, Ed 4

Lab 2
Introduction to Applets, Testing, & Debugging as well as More Drawing

Department
of
Electrical and Computer Engineering
University of Puerto Rico at Mayagüez

Before the Lab:

1- Review Java Applets.

2- Review `HTML` files. In addition to study basic `html` tags, study what you need to do to make the size of an applet pane dynamic and resizable respective to the size of the web browser's window when it's resized.

3- Make sure you know how to draw a circle with radius ***r*** **in the center** of a screen that its width is ***w*** and its height is ***h***? How about if you want to draw another object (different shape) in the center of the screen?

4- Print the *evaluation* sheet (the last page) before you go to the lab. You may want to print all pages as your reference during the lab.

During the Lab:

0- Login to the computer (1 minute)

1- A `Java Applet` Application (20 minutes)

- Open a new Java Project.
  - Name it: `My First Applet`

- Add a new class named `CirclesApplet`
  - Recall name of a class should start with a capital letter
  - Because this is just a class that we are defining and it is not going to run by itself, you do NOT need to create a *`static main`* method in it
  - By now, Eclipse is going to create a file named `CirclesApplet` and open it such that you can continue working on it

- In class `CirclesApplet`, you are going to use the following classes:
  - `JApplet` which is in the `swing` package
  - `Graphics` and `Graphics2D`, that are in the `awt` package
  - and `Ellipse2D.Double` that is in the `awt.geom` package

- Hence, you need to add 4 `import` statements to the beginning of your `CirclesApplet`
  - Here, it's one of them

    ```
    import java.awt.Graphics;
    ```
  - Guess and add the other 3 `import` statements.
  - If you get some errors, go to Java API Documentation and try to figure out what your mistake is.
    - In other words, you are given enough time to fix your errors by yourself without looking up the answer from other codes or from the lab instructors.

- In fact, you should always have the API Documentation open in your web browsers, and try to troubleshoot your syntax errors and other issues there first.

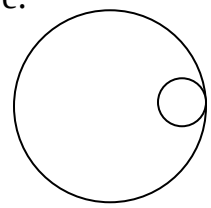
- Recall that `CirclesApplet` need to extend the `JApplet` class.

- Hence, add
  `extends JApplet`
  to the header of your class `CirclesApplet` just before `{`

- You also need to override method `paint` of the `JApplet` Class. Hence, add the following code to the body of your `CirclesApplet` Class; and complete the empty `(…)` spaces.

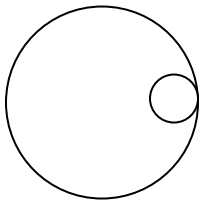
```
public void paint(Graphics g)
    {
       //Prepare for extended graphics
       Graphics2D g2 = (Graphics2D) g;

       //Construct a circle with center 200,150 & radius 50

            …
       //Construct a circle with c 237.5,150 & r 12.5

            …

       g2.draw(…);
       g2.draw(…);

}
```

- To see your applet from a web browser, you should create an `html` file.
- To do so, open an editor such the `notepad`.
- Add
  `<applet code="CirclesApplet.class" width="400" height="400">`
  `</applet>`
- Save your `html` file in the same folder that your `CirclesApplet.class` is saved.
- Now, open your `html` file with a web browser. You should see a picture similar to the following there.



Show your work to the lab instructor (20 Points).

- Modify your `CirclesApplet.java` and your `html` files such that the big circle is always in the center of the web browser window. (The radiuses of both circles always remain 50 and 12.5, respectively.) When you resize the web browser and refresh it, the big circle should be in the center of the window and you still should see the same picture



Show your work to the lab instructor (20 Points).

2- Unit Testing (15 minutes)

- Every time you define a class `A`, you can create a `Tester` class to test the logical functionality of your class `A`. This is called unit testing. Unit testing may identify only some logical errors meaning that if the `Tester` class passes all test cases, you still **cannot** be 100% sure that the unit does not have any logical error.

- Create a new project, and a new class named `MyInteger` in it.

```
public class MyInteger {
        public MyInteger(){
                value=0;
        }
        public MyInteger(int i){
                setValue(i);
        }
        public void setValue (int i) {
                value=i;
        }
        public int getValue (){
                return value;
        }
        public int binValue() {
                int numBin=0;
                int numInt=value;
                int remainder;

                while (numInt!=0) {
                        remainder=numInt % 2;
                        numBin=numBin*10+remainder;
                        numInt=numInt/2;
                }
                return numBin;
        }
        private int value;

}
```

- Now, in this project, add a new class called `Tester`
- And copy following contents in it.

```
public class Tester {

        public static void main(String[] args) {

                // Test Case 1
                 MyInteger i1=new MyInteger(1);
                 System.out.print("The int number is ");
                 System.out.println(i1.getValue());
                 System.out.print("Its binary rep is ");
                 System.out.print(i1.binValue());
                 System.out.print(" ; expected value is ");
                 System.out.println("1");

                // Test Case 2
                 MyInteger i2=new MyInteger(3);
                 System.out.print("The int number is ");
                 System.out.println(i2.getValue());
                 System.out.print("Its binary rep is ");
                 System.out.print(i2.binValue());
                 System. out.print(" ; expected value is ");
                 System.out.println("11");

        }
}
```

- What has the above `Tester` class tested?
- Can we now say that the `MyInteger` class has no logical errors?
- Now, by copying the above test cases and changing them slightly, add 5 more test cases to the `Tester` class.
- What is your observation?

Show your work to the lab instructor (20 Points).

3- Debugging (20 minutes)

- Follow the instructions of the lab instructors to run your above `Tester` project **line by line** and in particular **to monitor the values** of all local variables, instance fields, and parameters in the *Debug* perspective.
- The instructions should help you to figure out where exactly the problem is from as well as guidelines to fix it;

4- Unit Testing using JUnit (15 minutes)

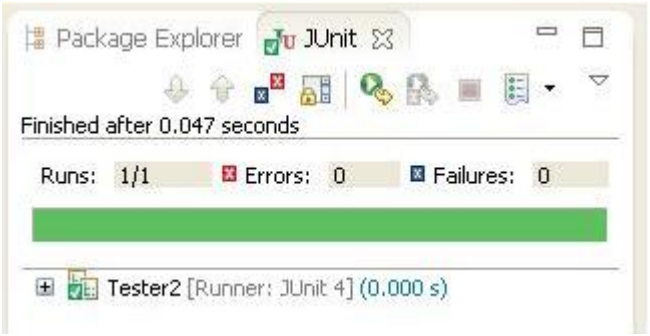
- To create a JUnit class, click on `File>New>JUnit Test Case` while you are still in the project above
- Name the new class: `Tester2`
- You should initially see the following code in your `Tester`

```
import static org.junit.Assert.*;
import org.junit.Test;


public class Tester2 {

        @Test
        public void test() {
                fail("Not yet implemented");
        }
}
```

- Add the following code to the `test()` method above

```
MyInteger i1=new MyInteger(13);
assertEquals("Binary representation of 13 must be 1101",1101,
                                                    i1.binValue());
```

- Remove the `fail` statement from the `test()` method
- Run `Tester2`
- You should see the following, with 0 errors and 0 test failures



- Go back to `Tester2` and add 4 more assertions, run it

Show your work to the lab instructor (20 Points).

5- More about JUnit (15 minutes)

- The lab instructors are going to explain more features of the JUnit, such as annotations `@Test`, `@Before`, `@Ignore`, and two more assertion test methods.

6- Start a new *Applet* project (20 minutes)

Reuse the project that you drew for Step 6 of Lab 1 (or start from scratch if you do not have it) to draw a colorful car, motorcycle, human, or pet of your interest; however, this time your drawing must 1) be shown in an *applet* 2) always in the center of the web browser. **When the lab instructor resizes the web browser and refreshes it, your drawing should still be in the center of the window.**

Show your work to the lab instructor (20 Points).

Moreover email a good screenshot of your drawing to your instructor lab and CC the professor.

After the Lab:

1- Investigate more features of test strategies.
2- Investigate more features of debugging.

Evaluating Lab 2

Write Your Section# here: ……..

Please evaluate the quality of the lab and performance of the instructors by filling up the following table and give it to your lab representative. (Choose 5 as the highest and 1 as the lowest grade).

Items	5	4	3	2	1
The lab started **on time SHARP**.					
The instructor was able to quickly answer your questions during the Development of **Applet Application** (Item 1 above), **Unit Testing** (Item 2 above), **Using JUnit** (Item 4 above) and **Drawing Application** (Item 6 above).					
The instructor gave a good workshop on **Debugging** and how to find and fix the error of the above project (Item 3 above) and managed his time accordingly.					
The instructor gave a good workshop on **other features of JUnit** (Item 5 above) and managed his time accordingly.					
The instructor followed the specified **timeline** for each step					
You found the lab today **overall** Great (helpful, fruitful, interesting, etc.).					