CPSC 1181
Lab 2: Introduction to Objects

Objectives:

1) To become familiar with the concept of an object as well as how to manipulate some of the objects that are available within the java libraries.

2) To learn the basics of designing your own class and practice some of the thought patterns necessary to design objects.

Introduction:

In the past, you have worked with the java libraries but you may have not been completely familiar with what you were interacting with. This lab re-introduces the String as a class, breaking it down into its components. The more complicated Rectangle class is studied in the same fashion. Finally, you will have to apply your analysis techniques to design two simple objects. Reference information for this lab can be found in *Big Java* Chapter 2: *Using Objects*, and *Chapter 3: Implementing Classes*.

Procedure:

(Part 1 - String objects)

One of the methods provided by the String class is the length method. This method counts the number of characters in a string. For example, let's use a sentence that is too long to count. If you are unfamiliar with String object from an earlier class, the class reference can be found at http://docs.oracle.com/javase/1.5.0/docs/api/java/lang/String.html

```
String intro = "When Gregor Samsa woke one morning from uneasy dreams he
found himself transformed in his bed into a monstrous verminous insect";
int n = intro.length();
```

All objects can be defined by the combination of a set of properties and a set of operations. Because intro is a String, one of its properties is its length. Strings have a large set of properties and operations that can be found in the java documentation. An example of an operation that can be performed on an object is the toUpperCase method. Add the following to your code, and print it to the screen:

```
String capsIntro = intro.toUpperCase();
```

Notice that the program applied the toUpperCase method to the original string. The result of which was set to capsIntro. Try and become a bit more familiar with String objects.

Examine the following complicated print statement:

```
System.out.println("The sentence \""+intro+ "\" is "+ intro.length()+"
characters long");
```

This statement uses the escape character \" as well as the + operator to concatenate string objects.

1.1) Write a program that changes all of the characters in **intro** into lower case. Save your code for submission.

1.2) As mentioned previously, all of the classes in the Java library are fully documented. Go to http://java.sun.com/javase/7/docs/api/index.html and find out how the **concat** method of the String class behaves.

2) Complete the following program so that it prints the message, "the quick brown fox jumps over the lazy dog", and save your code for submission.

```
public class ConcatDemo
{
   public static void main(String[] args)
   {
      String animal1 = "quick brown fox";
      String animal2 = "lazy dog";
      String article = "the";
      String action = "jumps over";
      /* Your work goes here */
      System.out.println(message);
   }
}
```

As seen in the intro measurement print statement the concat method can be replaced with the + operator. For example, the statements

```
String catTest = "beginning" + "end";
System.out.println(catTest);
```

Will print "beginningend" to the console.

3) Refer back to the documentation and complete the following program so that it prints the message "abcdefghi", Do this in only one line of code, then save your code for submission.

```
public class TrimIt
{
   public static void main(String[] args)
   {
      String sentence1 = "      abc   ";
      String sentence2 = "   def";
      String sentence3 = "ghi         ";
      /* Your work goes here */
      System.out.println(message);
   }
}
```

(Part 2 – Geometric Objects)

In part one you saw that a string object has both a set of properties (in this case, the set of unicode characters in the string) and a set of operations (toLowerCase, concat, trim, etc.). In this section we are going to apply that same abstraction to a Rectangle. A Rectangle has 4 corners, whose locations are its properties. Possible operations that can be performed on a Rectangle are 2D transformations, and calculating area and perimeter.

4.1) Copy the following program then add code to the program to create a second rectangle (r2) with the same properties as the first. Then, apply the grow method "grow(10,20)" to the second rectangle. Next, print both rectangles. To better understand the grow method, research it in the java API. Take note of other methods while you are reading. Save your code for submission.

```
import java.awt.Rectangle;

public class RectanglePrinter
{
   public static void main(String[] args)
   {
      Rectangle r1 = new Rectangle(0, 0, 100, 50);
      /* Your code goes here */
      System.out.println(r1);
      /* and here */
   }
}
```

Q4.2) Modify your program so that the lines creating the second rectangle change from

```
Rectangle r2 = new Rectangle(0, 0, 100, 50);
```

to

```
Rectangle r2 = r1;
```

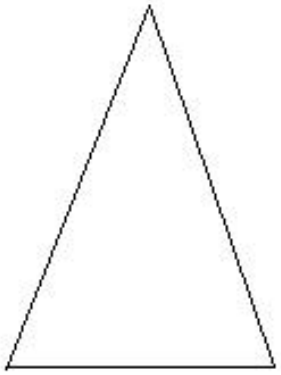
What is the output now? How can you explain this? Save your answer in a text file.

5) Now that you are more familiar with shapes, you will be introduced to the swing library which is a useful library used to graphically represent objects. To test your ability to read the java documentation, and study code, you now have to fill in the blanks in the following code. Do not worry about the 'your code goes here' comments yet, they are part of question 6.

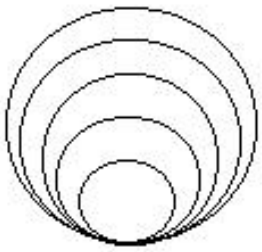
```
import java.awt.Graphics;
import java.awt.Graphics2D;
import javax.swing.JFrame;
import javax.swing.JComponent;

public class TestFrameViewer
{
   public static void draw(Graphics g)
   {
      Graphics2D g2 = (Graphics2D) g;
      //your code goes here
   }
   public static void main(String[] args)
   {
      ____ frame = new ____();

      final int FRAME_WIDTH = 250;
      final int FRAME_HEIGHT = 250;

      frame.____(FRAME_WIDTH, FRAME_HEIGHT);
      frame.setTitle("A Test Frame");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      JComponent component = new JComponent()
      {
         public void paintComponent(Graphics graph)
         {
            draw(graph);
         }
      };
      frame.add(component);
      frame.____(true);
   }

}
```

In this code, a frame is initialized and the painting component is added to the frame. In the draw method, you will use the graphics2D.draw(shape) method to create the next few images.

Q 6.1) Use Line2D.Double objects to drawn a triangle that looks like this. Add those components to your frame. Save your code for submission.



Q 6.2) Generate five circles with diameters 40, 80, 120 ,160 , and 200 all tangent to a common point then color all of those circles a different color. Investigate the class Ellipse2D.Double to draw the shape. When finished save your code for submission.



Challenge** write a method that generates a spiral using only 2D lines. This is difficult and will involve a fair amount of math as well as programming knowledge. Try to implement the method by passing the previous line in the spiral and having the method produce another line based on the size and angle of the line passed in the argument. This is not for marks, but will help you understand the basics of fractal imagery.

(Part 3 – Designing Objects)

In the first two portions of this lab you used objects that were already defined in the java library. In order to program with objects, you will need to construct your own. To help with

the idea of creating objects we will use the running analogy of a door. To begin, create a new class file and name it “Door”. This time, your class file will not contain a main(String[] args) method.

As mentioned before, an object is **a set of properties and a set of operations**. We need to first consider the properties that a door has. It may be useful to refer to a door by its name e.g. “Front” or “Side”. Therefore, one property of a door would be its name, which we can represent with a String. Furthermore, a property of an object can be its state such as “open” or “closed”.

Add the two instance variables **name** and **state** to the Door class. Make them **private** and make them **Strings**. The door will have two methods **open(),** and **close(),** declare them **public** and **void**.

With the definitions in place, you have successfully created a class. However, the default java constructor will initialize the name and state of your door to NULL. Create a constructor for the Door with two String parameters for the name and the state.

It is often convenient (and always secure) to have accessor methods that operate on a single instance variable. Here is an accessor method for the name variable:

```
public String getName()
{
   return name;
}
```

The word String in front of getName() indicates that the method returns a String when it is invoked. The body is simple and just returns the value of name.

Many instance variables in a class will have corresponding mutator methods that allow you to change the value of the variable. Here is a mutator method for the name variable:

```
public void setName(String newName)
{
   name = newName;
}
```

These two types of methods for accessing and mutating attributes are called accessor methods and mutator methods.

The door can be opened or closed, changing its state. You should also write a method to get the state. Use the following code:

```
public void open(){
      state = "open";
```

```
}
public void close(){
      state = "close";
}
public String getState(){
      return state;
}
```

Now that you have created a class with properties and methods, you can create an object of it. Use the following code to examine your door object.

```
package question7;

public class DoorTester
{
   /**
      Tests the methods of the Door class
      @param args not used
   */
   public static void main(String[] args)
   {
    Door frontDoor = new Door("Front", "open");
    System.out.println("The front door is " + frontDoor.getState());
    System.out.println("Expected:  open");
    Door backDoor = new Door("Back", "closed");
    System.out.println("The back door is " + backDoor.getState());
    System.out.println("Expected:  closed");
    // Use the mutator to change the state variable
    backDoor.open();
    System.out.println("The back door is " + backDoor.getState());
    System.out.println("Expected:  open");
    // Add code to test the setName mutator here
   }
}
```

7.1) implement a boolean method called **isOpen()** which returns true if the door is open and returns false otherwise. What does the code for your door look like? Save this code for submission, but keep it handy because you will need to reuse it for question 9.

7.2) If the door should only be open or closed, how can an invalid state occur in this object? How would you fix it? What other properties can a door have that could have been included in this object? What functions would be used to manipulate those properties?

Now you will use the same object oriented thought pattern to model a postcard. Imagine you are on vacation and want to send postcard to your friends. A typical postcard might look like this:

```
Dear Richard: I am having a great time on the
Island of Java. The weather is great. Wish you were here!
From,
Linus
```

Suppose you decide to write a computer program that sends postcards to various friends, each of them with the same message, except that the first name is substituted to match each recipient.

Q8.1) The following class implements a Postcard. Notice that we do not set the recipient in the constructor because we want to be able to change the recipient, and keep the same message and sender. What method would you add to support this functionality? Implement the method.

```
public class Postcard
{
   private String message;
   private String sender;
   private String recipient;
   public Postcard(String aSender, String aMessage)
   {
      message = aMessage;
      sender = aSender;
      recipient = "";
   }
   // Your method here
}
```

Add a print method to the Postcard class to display the contents of the postcard on the screen. Use the instance variables message, sender, and recipient defined in the last step when you implement the method. Remember that you can concatenate strings with the "+" operator and a newline can be denoted with the "\n" escape character.

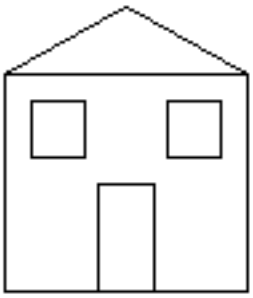
To ensure that your code is functional, use the following main method to test it:

```
public class PostcardPrinter
{
   public static void main(String[] args)
   {
      String text = "I am having a great time on\nthe island of Java.
Weather\nis great. Wish you were here!";
      Postcard postcard = new Postcard("Janice", text);
```

```
      postcard.setRecipient("Sue");
      postcard.print();
      postcard.setRecipient("Tim");
      postcard.print();
   }
}
```

Submit your finalized postcard class.

9) Finally you must bring two of your pieces of code together. First create a new file and mimic the drawing frame code, seen earlier for drawing the triangle and the embedded circles, to draw a house out of 4 rectangles and a triangle like so:



Once you have drawn the house, consider the fact that it has a door. We are now going to merge the two objects with the use of inheritance, at the top of your door file, add “extends Rectangle” to the class definition as so.

```
import java.awt.Rectangle;

public class Door extends Rectangle
```

Now in your draw method where you drew the other rectangles and lines for the house, add an instance of your door.

```
Door frontDoor = new Door("front","open");
```

You should now notice that you have functionality with the front door the same as with a rectangle. Therefore you can use functions like frontDoor.setBounds() to give the door some dimensions. You will also be able to draw your door just like any of the lines or rectangles.

The final task for this lab is that you should write an if statement which checks whether or not the door is open, and if it is open the shape of the door should be filled in rather than just outlined. Submit both the code for your drawing class, and the newly modified Door class.