Assignment7 of Xiaowei Liu

```
>>>>>>>>>>>>>>>>>>>>>
Largest Shape info: Circle{radius=5.0, pi=3.141592653589793, area=78.53981633974483}
>>>>>>>>>>>>>>>>>>>>>
Circle{radius=2.0, pi=3.141592653589793, area=12.566370614359172}
Circle{radius=5.0, pi=3.141592653589793, area=78.53981633974483}
Circle{radius=3.0, pi=3.141592653589793, area=28.274333882308138}
Rectangle{length=3.0, width=3.0, area=9.0}
Rectangle{length=6.0, width=3.0, area=18.0}
Serialize Done!
>>>>>>>>>>>>>>>>>>>>>
Deserialize Done!
Circle{radius=2.0, pi=3.141592653589793, area=12.566370614359172}
Circle{radius=5.0, pi=3.141592653589793, area=78.53981633974483}
Circle{radius=3.0, pi=3.141592653589793, area=28.274333882308138}
Rectangle{length=3.0, width=3.0, area=9.0}
Rectangle{length=6.0, width=3.0, area=18.0}
```

Driver.java

```
// Importing necessary libraries for input-output operations and working with lists
import java.io.*;
import java.util.LinkedList;
import java.util.List;

// The main class of your program
public class Driver {

    // The main method where the program execution begins
    public static void main(String[] args) throws IOException, ClassNotFoundException {

        // Creating a list to store different shapes
        List<Shapes> list = new LinkedList<Shapes>();

        // Adding circles and rectangles to the list
        list.add(new Circle(2));
        list.add(new Circle(5));
        list.add(new Circle(3));
        list.add(new Rectangle(3,3));
        list.add(new Rectangle(6,3));

        // Finding the shape with the largest area
```

```
        Shapes maxArea = largestShape(list);

        // Printing information about the largest shape
        System.out.println(">>>>>>>>>>>>>>>>>>>>>>");
        System.out.println("Largest Shape info: " + maxArea.toString());
        System.out.println(">>>>>>>>>>>>>>>>>>>>>>");

        // Serializing the list of shapes
        Serialize(list);
        System.out.println(">>>>>>>>>>>>>>>>>>>>>>");

        // Deserializing and printing the shapes from the file
        Deserialize();
    }

    // Method to find the shape with the largest area in the list
    public static Shapes largestShape(List<Shapes> list){
        int index = 0;
        double maxValue = 0;

        // Loop through the list and calculate the area of each shape
        for(int i = 0; i < list.size(); i++){
            list.get(i).calculateArea();

            // Compare the area with the current maximum value
            if(list.get(i).area >= maxValue){
                index = i;
                maxValue = list.get(i).area;
            } else {
                continue;
            }
        }

        // Return the shape with the largest area
        return list.get(index);
    }

    // Method to deserialize the list of shapes from a file
    public static void Deserialize() throws IOException, ClassNotFoundException {

        // Initialize an ObjectInputStream to read from the file
        ObjectInputStream in = new ObjectInputStream(new FileInputStream("list.txt"));

        // Read the object from the file
        Object obj = in.readObject();
        System.out.println("Deserialize Done!");

        // Print information about each shape in the deserialized list
        for(Shapes shapes: (List<Shapes>)obj){
            System.out.println(shapes.toString());
        }
    }
```

```
    // Method to serialize and write the list of shapes to a file
    public static void Serialize(List<Shapes> list) throws IOException {

        // Print information about each shape in the list
        for(Shapes shapes: list){
            System.out.println(shapes.toString());
        }

        // Initialize an ObjectOutputStream to write to the file
        ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("list.txt"));

        // Write the list of shapes to the file
        out.writeObject(list);
        System.out.println("Serialize Done!");
    }
}
```