Object-Oriented Design and Implementation
Assignment 1

Important *This is an individual assignment. No group work is allowed.*
Due Date 11:59 PM on March 14, 2016
Goals

- Create a Java application using GUI, serialization, and collection classes
- Use figure drawing operations

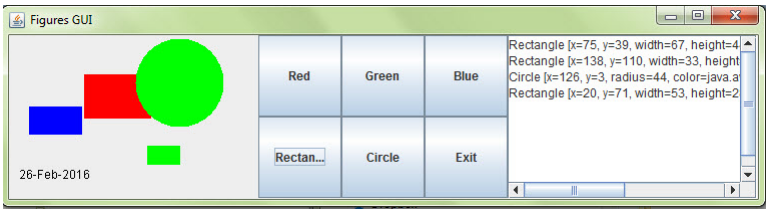
Problem Create, test, and document a Java program with the following classes.

Figure This abstract class (given to you) represents a figure that can be drawn on the GUI.

Rectangle This extends Figure and implements the functionality needed to draw filled rectangles with a given color.

Circle This extends Figure and implements the functionality needed to draw filled circles with a given color.

The GUI should appear as below. (Minor deviations are fine.)



Watch the video file `PA1.mp4` on the Moodle site to see the program in action.

The current date should appear in the bottom left corner of the drawing. All figures appearing in the GUI should be listed with their String representations in the JTextArea appearing on the right side.

As in the video file, to draw the rectangle, the user first clicks the top left corner and then the bottom right corner. To draw a circle, the user clicks the center and a point that is to the right and bottom of the first click; the second point is on the circumference.

When the exit button is clicked, serialize the figures and the content of the text area to disk in a file named "figures". When the GUI is started, check for the existence of a file named "figures" and if it exists, open it to populate the figures and the corresponding list in the text area. You must not assume any specific directory structure. For example, do not write to a file named `C:\Temp\figures`. Instead, use "figures" to ensure that it gets stored in or retrieved from the current directory.

You may wish to use the file `PA2 Starter.zip` available under Assignments.

Lay out your code properly.

Program Submission

Zip the source files and submit to the dropbox. Gross violations such as missing some files or not submitting a zip file will incur penalties. If the program has syntax errors, the grade will be 0: no exceptions.

Program Grading Criteria

Correctness of the Java code: 40 points

- GUI layout: 10
- Can create rectangles as specified: 6
- Can create circles as specified: 6
- Can use all three colors properly: 4

- Lists all figures correctly: 4
- Save works correctly: 4
- Exit works correctly: 2
- Retrieves saved data: 4

Program structure: 30 points

- Proper access for methods and fields: 4
- Properly coded logic: 9
- Proper organization of methods: 9
- Proper organization of actionPerformed: 8

Coding standards: 10 points

- Proper indentation and line breaks, etc.: 6
- Properly named identifiers: 4

Coding Standards It is important that you properly format your program. There are several reasons why it is important to follow coding conventions. There is the high cost of software maintenance. It is rare that the original author would maintain a piece of software for its entire life. It is quite difficult for most people to remember their code if they haven't seen it for a long period of time. Programmers should create code that they are proud of.

In your programs, you must obey the following subset of conventions. As we write more complicated programs, I may ask you to follow more requirements.

- Do not type long lines. Try to limit them to 80 characters
- If an expression does not fit on a single line, break it according to these general principles:
  - Break after a comma or before other operators.
  - Prefer higher-level breaks to lower-level breaks: For example, if the following expression

    ```
    (a + b * (c + d)) * (e + f)
    ```

    runs into multiple lines, you should try to break it as

    ```
    (a + b * (c + d))
    * (e + f)
    ```

    as opposed to

    ```
    (a + b *
    (c + d)) * (e + f)
    ```

  - Attempt to align the new line with the beginning of the expression at the same level on the previous line.
  - If the application of the above rules results in ugly-looking code, just indent with enough tabs that make the code look reasonable.
- Variable Names: Variable names must begin with a lower-case letter and be in full words. Words after the one must have their first letter capitalized.

  Here are examples of acceptable variable names:

  ```
  count costOfItem gallonsPumped testSucceeded
  ```

Here are examples of unacceptable variable names:

```
Count itmCst gallonspumped tstSucceeded
```

- Indentation and Alignment: If you are using Eclipse, use `Window->Preferences, Java, Editor, Save Actions` and check the `Format source code` option. Eclipse will format your source code when you save the file.
  - Indent code within a class by one tab.
  - Indent code within a method by one tab.
  - Align code as below.

    ```
    int thatValue;
    int thisValue = 10;
    int count;
    thatValue = 0;
    count = Math.min(thatValue, thisValue);
    ```

- There are more requirements: like putting a space before and after a binary operator other than comma, separating tokens properly with a space, writing version number, your name, etc. at the beginning, writing information about methods, etc. As we write more involved programs, we will learn more and more standards.

Refer to the code below for an example of a program with acceptable formatting.

```
import java.util.LinkedList;
public class LinkedSet implements Cloneable, Set {
    LinkedList list = new LinkedList();
    public boolean insert(int value) {
        if (isMember(value)) {
           return false;
        }
        return list.add(new Integer(value));
    }

    public boolean remove(int value) {
        return list.remove(new Integer(value));
    }

    public Object clone() {
       LinkedSet copy;
       try {
          copy = (LinkedSet) super.clone();
       } catch (CloneNotSupportedException cnse) {
         cnse.printStackTrace();
         return null;
       }
       copy.list = new LinkedList();
       for (int i = 0; i < list.size(); i++) {
           copy.list.add(new Integer(((Integer) list.get(i)).intValue()));
       }
       return copy;
    }
}
```