Programming Assignment 10 (Pair Programming)

Total Points (30 pts) - Due Wednesday, December 9 at 11:00 AM

Interactive Drawing Application (30 points)

This assignment is designed to test your continued understanding of writing graphical applications in Java. You will implement a GUI application that uses the *MyShape class* to create an interactive drawing application. You'll create two classes for the GUI and provide a test class that launches the application. The classes of the *MyShape* hierarchy require no additional changes.

Components and Event-handling:

The first class to create is a subclass of JPanel called *DrawPanel*, which represents the area on which the user draws the shapes.

Class *DrawPanel* should have the following instance variables:

- An array shapes of type *MyShape* that will store all the shapes the user draws.
- An integer *shapeCount* that counts the number of shapes in the array.
- An integer *shapeType* that determines the type of shape to draw.
- A MyShape *currentShape* that represents the current shape the user is drawing.
- A Color *currentColor* that represents the current drawing color.
- A JLabel *statusLabel* that represents the status bar. The status bar will display the coordinates of the current mouse position.

Class *DrawPanel* should also declare the following methods:

- Overridden method *paintComponent* that draws the shapes in the array. Use instance variable *shapeCount* to determine how many shapes to draw. Method *paintComponent* should also call currentShape's draw method, provided that currentShape is not null.
- Set methods for the *shapeType* and *currentColor*.
- Method *clearLastShape* should clear the last shape drawn by decrementing instance variable *shapeCount*. Ensure that shapeCount is never less than zero.
- Method *clearDrawing* should remove all the shapes in the current drawing by setting shapeCount to zero.
- Methods *clearLastShape* and *clearDrawing* should call repaint (inherited from JPanel) to refresh the drawing on the DrawPanel by indicating that the system should call method *paintComponent*.

Class *DrawPanel* should also provide event handling to enable the user to draw with the mouse. Create a single inner class that both and implements *MouseMotionListener* and *MouseListener* to handle all mouse events in one class.

In the inner class:

- override method *mousePressed* so that it assigns *currentShape* a new shape of the type specified by shapeType and initializes both points to the mouse position.

- Next, override method *mouseReleased* to finish drawing the current shape and place it in the array. Set the second point of currentShape to the current mouse position and add currentShape to the array. Instance variable *shapeCount* determines the insertion index. Set *currentShape* to null and call method repaint to update the drawing with the new shape.
- Override method *mouseMoved* to set the text of the *statusLabel* so that it displays the mouse coordinates—this will update the label with the coordinates every time the user moves (but does not drag) the mouse within the DrawPanel.
- Next, override method *mouseDragged* so that it sets the second point of the currentShape to the current mouse position and calls method repaint. This will allow the user to see the shape while dragging the mouse.
- Also, update the JLabel in mouseDragged with the current position of the mouse.

Create a constructor for *DrawPanel* that has

- A single JLabel parameter. In the constructor, initialize statusLabel with the value passed to the parameter.
- Also initialize array shapes with 100 entries, shapeCount to 0, shapeType to the value that represents a line, currentShape to null and currentColor to Color.BLACK.
- The constructor should then set the background color of the DrawPanel to Color.WHITE and register the MouseListener and MouseMotionListener so the JPanel properly handles mouse events.

Next, create a JFrame subclass called *DrawFrame* that provides a GUI that enables the user to control various aspects of drawing.

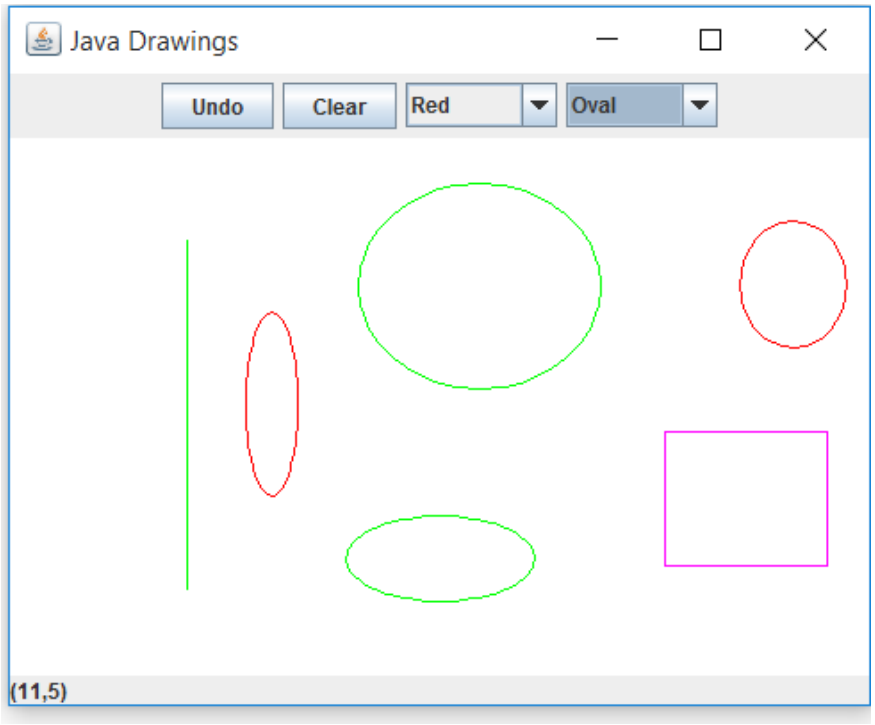
For the layout of the DrawFrame, we recommend a BorderLayout, with the components in the NORTH region, the main drawing panel in the CENTER region, and a status bar in the SOUTH region. In the top panel, create the components listed below. Each component's event handler should call the appropriate method in class DrawPanel.

- A button to undo the last shape drawn.
- A button to clear all shapes from the drawing.
- A combo box for selecting the color from the 13 predefined colors.
- A combo box for selecting the shape to draw.

Declare and create the interface components in DrawFrame's constructor. You'll need to create the status bar JLabel before you create the DrawPanel, so you can pass the JLabel as an argument to DrawPanel's constructor.
Finally, create a test class *TestDraw* that initializes and displays the *DrawFrame* to execute the application.

Sample Run:

I have provided an executable jar sample file.



Extra Credit: Palindrome (5 points)

A **palindrome** is a string that reads the same forward and backward, such as "radar". Write a static recursive method that has one parameter of type String and returns true if the argument is a palindrome and false otherwise.

Disregard spaces and punctuation marks in the string, and consider upper- and lowercase versions of the same letter to be equal. For example, the following strings should be considered palindromes by your method:
"Straw? No, too stupid a fad, I put soot on warts."
"xyzcZYx?"

Here are some well-known palindromes:

Able was I, ere I saw Elba

Kayak

Desserts, I stressed

Demonstrate the method in a program that accepts a string as input from the user, then returns whether or not the string is a palindrome.

Name this file *Palindrome.java*

Notes:

- The algorithm for this Part is a bit tricky. The recursive algorithm leads to some inefficiency. For example, the problem statement asks for a method that takes a string with spaces and punctuation, but only looks at the letters and returns a Boolean value. So the method must parse the input string and eliminate everything but letters.
- Although the string needs to be parsed just once, a recursive algorithm must be identical each iteration, so the parsing occurs each iteration.
- The base case either returns TRUE if the string has zero or one characters, or, if the string has two or more characters, it checks the first and last characters and has a more complex algorithm to determine whether to return TRUE or FALSE.
- If the two letters (the first and last) are different it returns FALSE. But if the two are the same, it returns the result of a recursive call with a smaller string, with the first and last letters removed. So each iteration reduces the string by a pair of letters until the string is down to zero or one character (even or odd number of letters, respectively, in the original string).

An alternative approach would be to write a second recursive method that looks at each character in the string recursively and appends it to the result if it is not space or punctuation.

Sample Run:

```
Enter a string
Able was I, ere I saw Elba

The sentence is a palindrome

Enter another sentence? (y/n)
y
Enter a string
Java Software Solutions

It is NOT a palindrome

Enter another sentence? (y/n)
y
Enter a string
Desserts, I stressed

The sentence is a palindrome

Enter another sentence? (y/n)
n
```

Submitting Your Assignment

Each assignment requires two submissions: an electronic component submitted through Insight, and a hardcopy component submitted in class.

- The hardcopy submissions must include the following:
  1. A copy of your program. This copy needs to include all the code that you have written for the assignment.
  2. One or more sample runs of your program. In addition to the program copy, submit examples showing the output of your program.
- Additionally, your program must be uploaded to Insight. To upload:
  1. Create a new directory (Assignment10) on your computer, and put all files:
     (TestDraw.java, DrawPanel.java, and DrawFrame.java) inside that directory. Then, compress, or zip up, the directory.
  2. Upload the file Assignment1.zip to Insight by the due date/time.
- The program uploaded to Insight must be the exact same one submitted in class.