Homework Assignment 6

This homework assignment will give you practice with:

1. Java's layout managers
2. the MVC model
3. custom drawing and event handling in Java

---

1. Modify the BoxLayoutDemo found in /home/bvanderz/cs365/hw/hw6 in the following manner:
    a. The layout should be horizontal
    b. The buttons should be divided into two groups with group 1 consisting of buttons 1, 2, and 3 and group 2 consisting of buttons long-named button 4 and 5. If the window is made bigger, group 1 should cling to the left side of the window and group 2 should cling to the right side of the window. You do not need to use two layout managers to accomplish this grouping--you put "something" between buttons 3 and 4.
    c. Within each group there should be 20 pixels of space between each pair of buttons.

    If you have questions about how your layout should look or behave, you can run my executable:

    ```
    java -jar BoxLayoutDemo.jar
    ```

2. Write a Java Swing application that displays three views of a numeric value that may be adjusted: 1) a slider view that contains a slider and two buttons, an increment button and a decrement button, 2) a text view that contains a label that represents the value of the numeric value, and 3) a gauge view that contains a dial that allows you to manipulate the numeric value. The program should take four integers as command line arguments:

    a. minValue: The minimum value of the slider
    b. maxValue: The maximum value of the slider
    c. startValue: The starting value of the slider
    d. increment: The amount by which the buttons increment or decrement the slider's value

    These 4 values are the model's state information and you will need to write methods that allow the views to query these values and that allow the controllers to change the slider's current value. Technically startValue is not part of the model's state information but instead is used to initialize the model's current value, which is part of the model's state information.

    Here are additional program requirements:

    a. The increment and decrement buttons should increment and decrement the value of the slider by the increment value. The program should use the Integer class's parseInt method to convert the command line arguments to integers and it should use a try/catch block to catch a NumberFormatException if any of the four arguments cannot be converted to an integer. Your program should print an appropriate error to the console and then exit.

    b. You should position the three views using a BorderLayout manager. Place the text view in the north (top) region, the slider view in the center region, and the gauge view in the bottom region. The slider and gauge views will have to be placed in their own JPanels. I want the elements of the slider view to be laid out horizontally, in the order decrement button, slider, and increment button. There are at least three layout managers that I can think of which will position these three

elements correctly, so I will leave it up to you to choose one.

c. If the user presses the increment button and the value is already at the maximum allowable value for the slider then the application should pop up a message dialog indicating that the value cannot be further incremented (use the JOptionPane's showMessageDialog method). A similar message should be displayed if the user attempts to decrement the value when it is already at its minimum value. If the user attempts to increment the value and the increment would move the value past its maximum permissable value, then increase the value to the maximum value, but do not print an error message. Do the same thing for the decrement button.

d. The gauge view should be implemented using a class named `GaugeView` and it will draw a custom picture of a gauge with a needle. The view should allow the user to drag the needle to change the value of the model. The gauge view has the following requirements:
    a. it should consist of an arc that extends from 3:00 to 9:00 in a counter clockwise direction and a line (i.e., the needle) that extends from the center of the arc to the boundary of the arc. The arc can be drawn using the Graphic object's `fillArc` command.
    b. the arc must be orange and the line must be blue.
    c. the user can drag the needle around the arc by pressing on the mouse button while the mouse cursor is closer than 10 pixels from the endpoint of the needle and dragging the mouse cursor. The mouse cursor may move anywhere while the cursor is being dragged--it does not need to remain within 10 pixels of the endpoint of the needle. The needle should follow the mouse cursor around the arc and the needle's endpoint should stay attached to the boundary of the arc.
    d. Use a MouseListener to detect the MousePressed and MouseReleased events and the MouseMotionListener to detect the MouseDragged event. The coordinates of the mouse cursor can be obtained from the MouseEvent that is passed to your event handling methods. The distance of the cursor from the endpoint of a line is given by the equation:

    $$\text{sqrt}((\text{cursor.x} - \text{line.x})^2 + (\text{cursor.y} - \text{line.y})^2)$$

    e. Your mouse point will probably not lie directly on the circle's boundary so you will have to obtain the closest point on the circle to the mouse point. This point is obtained by drawing a line from the center of the circle to the mouse point. You can use atan2 (arctan) to obtain the angle and then the polar coordinate equations for a circle will give you the (x, y) points. For those of you not familiar with polar coordinates, the equations are:

    $$x = \text{radius} * \cos(\Theta)$$
    $$y = \text{radius} * \sin(\Theta)$$

    where Θ is your angle.
    f. 3:00 on the arc represents the minimum value of the gauge. Values increase in a counter-clockwise direction to 9:00 on the arc. You can use the angle of the needle to determine the value of the gauge. For example, if the minimum value of the gauge is 0, the maximum value is 100, and the angle is 90 degrees (i.e., 12:00), then the value of the gauge is 50.
    g. the preferred width and height of your gauge view should be 200x100.
    h. the arc should be horizontally centered within its JPanel and vertically attached to the bottom of its JPanel.
    i. The diameter of the arc must be the minimum of the width or 2 times the height of its JPanel. You can get the width and height of a JPanel using the getWidth and getHeight methods. As an example, if the layout manager assigns the JPanel a width of 450 and a height of 100, then the arc's diameter must be 200 pixels, which is 2 times the height.

Designing Your Implementation

Your implementation must fit the MVC model. If you are unsure of how to split up the code, look at my sample MVC application that I covered in the videotape lecture. Put all your files in a package named SliderApp, including the "Glue" file that will contain your main method.

Layout Hints

1. You will need to use the setHorizontalAlignment method for a JLabel since BorderLayout ignores the setAlignmentX method.
2. Remember that you can get gaps in a BorderLayout by using the hgap and vgap properties.
3. If you extract the content pane from a JFrame using getContentPane, make sure you downcast the content pane to a JPanel, because the return type from getContentPane is a Container object.
4. If you extract the layout manager from the JFrame's content pane, make sure you downcast the layout manager to a BorderLayout, because the return type from getLayout is LayoutManager.

My Executable

If you have questions about how your application should look or behave, then execute the SliderApp.jar file in /home/bvanderz/cs365/hw/hw6 using the command:

```
java -jar SliderApp.jar minValue maxValue startValue incrementValue
```

A sample invocation might be:

```
java -jar SliderApp.jar 50 100 80 5
```

You do not have to get the spacing between elements exactly like mine, but I do want the general positions to be the same, and I want the text view centered.

Please make sure that when the window is re-sized and made bigger that everything stays laid out in the manner that I specified above (centered text view, diameter of the arc must be the minimum of the width or 2 times the height of its JPanel, something reasonable happens with the slider view). In particular it is not okay for the buttons in the slider view to stay in the same place, something needs to get resized to accommodate the extra space. Since you may use a different layout manager than I used, the manner in which the slider view is updated may be different than mine but I want something reasonable to happen.

More Implementation Hints/Notes

Here are a few things to which you should pay careful attention:

1. The y-axis of a window goes from the top to the bottom of the window. Hence in a window that is 200 pixels high, a y-value of 0 occurs at the top of the window and a y-value of 200 occurs at the bottom of the window.
2. When trying to obtain an angle using the arctan function, you should use Java's Math.atan2 function rather than its Math.atan function. The difference is that atan2 handles the case where x is 0 (occurs at $\pi/2$ radians) whereas atan will not work if x is 0 (y/x will yield NaN and atan will return NaN rather than $\pi/2$ radians).
3. Java's fillArc command uses degrees but its Math trigonometric functions use radians. You can convert back and forth using `Math.toDegrees` and `Math.toRadians`.

What To Submit

Submit two jar files named BoxDemoLayout.jar and SliderApp.jar. Both should be executable and contain the source files for their problems.