Lab 3 — Observer Pattern

CS 205

Due Oct. 15 at 11:55 pm

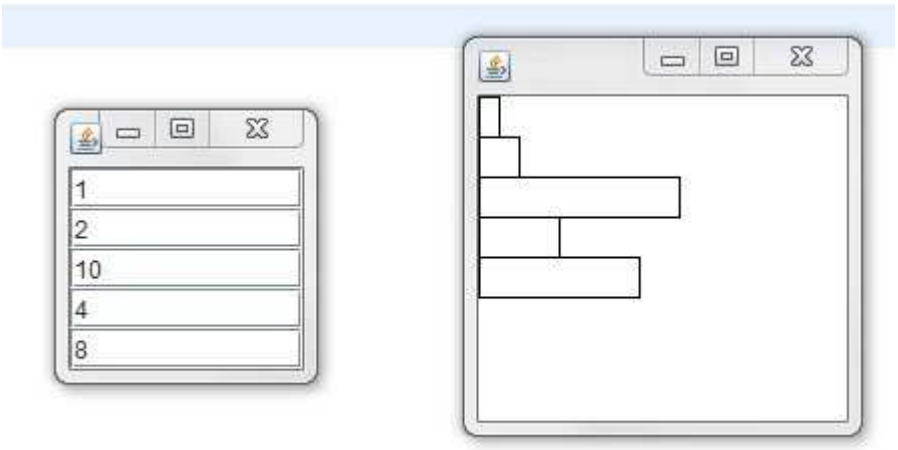
Lab objectives:

- Implement the observer pattern within a Model/View/Controller architecture.

Pairs for this lab:

- Nico and Conor
- Justin and Sam
- Jay, Weston, and Alex
- Anthony and Jordan
- Alison and Drew
- John and Surbhi

You will implement the classical example of the observer pattern — a data set and two views, one text view and one graphics view.



The data model consists of an array of integers. For simplicity, you may assume that the integers are in a range that is suitable for plotting in the graphical view. The text view is a collection of `JTextField` components. Editing a text field updates the model and hence the other view. The graphical view is a simple bar chart. Clicking anywhere next to a bar moves the bar to the mouse position.

You *must* use the Observer interface and the Observable class as follows:

```
public class DataModel extends Observable
{
    . . .
}

public class TextView extends Box implements Observer
{
    public TextView(DataModel aModel)
    {
        super(BoxLayout.Y_AXIS);
        . . .
    } // TextView constructor
    . . .
} // TextView class

public class GraphView extends JPanel implements Observer, MouseListener
{
    public void paintComponent(Graphics g)
    {
        super.setBackground(Color.white);
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D)g;
        . . .
    }
    . . .
}
```

Hints:

1. Your main method will create an array of integers which can be either read-in or hard-coded. You will create `DataModel`, `TextView`, and `GraphView` objects and register listeners and observers. Make two `JFrame` objects. In the first frame, set its content pane to the text view and pack it. In the second frame, set the content pane to the graph view. Use `setBounds` to move and set the graph frame to an appropriate size.

2. The `Box` class takes care of packing the `JTextField` components vertically. Just use the `add` method.

3. Each `JTextField` object should have an `actionListener` which will take the appropriate action when the text value is changed.

4. The graph view should have a `MouseListener` which takes the appropriate action when the mouse is clicked. You can have empty methods for the other mouse actions.

5. The `DataModel` class will need to use the `setChanged` and `notifyObservers` methods inherited from the `Observable` class when a data value is changed.

6. Give each bar in the graph view a fixed height. I used

   ```
   private static final int HEIGHT = 20
   ```

   Then when a user clicks on the point `(x,y)`, the graphical view can tell the model to set the data point with index `y/HEIGHT` to `x`. You can find out which point was clicked from the `MouseEvent`.

Add Javadoc comments to record the members of your team in the Class file containing your `main()` method and also to document the code that you write for the lab. Export your lab into a ZIP archive and submit it in GoucherLearn.