This assignment assumes you have studied *Tutorial: Time Of Day, Part 2, GUI Design in NetBeans* available as a pdf-file from the website [1].

**Assignment Rectangles**

The GUI application to be developed should produce a window that looks approximately like the screen shot below. The application maintains a list of axis parallel grid rectangles. The rectangles from the list are shown with a black border and a yellow interior all labeled with their index in the list. The so-called *bounding box* is shown using its red border. Lower numbered rectangles are drawn over higher numbered rectangles. With a click of the mouse a rectangle can be selected which will then be shown with a light blue (cyan) interior on top of all other rectangles. Its index and coordinates can then be seen in the text fields on the right. Text field *Count* shows the number of rectangles. Using the buttons the list can be emptied (*Clear*), extended with a random rectangle (*New Rectangle*), or completely replaced by a list of random rectangles (*New List*).
The application can be built in the following steps

1. Download the skeleton project Rectangles from the website [1]. This skeleton project contains RectanglesApp.java providing an empty JFrame form with some additional code added for this assignment and Rectangle.java providing the class Rectangle complete with its contract. Study this contract.

2. Lay out the visual components on the form as indicated in the picture below

- to the left is a large JPanel indicated by an orange boundary and to the right is a narrower but equally high JPanel having a black line border (property border)
- this last JPanel contains two smaller JPanels of which the top one contains all JButtons (vertically centered) and the bottom one contains six JLabels (right aligned with one another) and six JTextFields (left aligned with one another, all having the same width, all having the editable property disabled); the two subpanels have black line borders in order to be able to distinguish them in the picture, in the final applications they should have no border
- the standard texts on all JButtons and all JLabels has been adapted to the texts shown; the standard texts in all JTextFields have been deleted

When laying out components on the form they are automatically given standard names like JPanel1 and JButton2. Give the components that are added the more informative names from the following list
After you have completed the layout of the application set the Layout properties Horizontal Resizable and Vertical Resizable of displayPanel to enabled (tick marks). For panel controlInfoPanel set Layout property Horizontal Resizable to disabled (no tick mark) and Layout property Vertical Resizable to enabled (tick mark).

3. The form can be edited in two ways: through the Design View (graphical view for placing components, setting properties of components, etc.) and through the Source View (textual view allowing direct editing of the code). In the following chose the appropriate view for the editing you are asked to perform.

4. Add two private fields to the form at the place indicated in the code:

   - private ArrayList<Rectangle> rectangleList; which contains the current list of rectangles
   - private int selected; which contains the index of the selected rectangle in the list (−1 if no rectangle is selected)

Add the following initialization code for these fields to method initAddedComponents

```java
rectangleList = new ArrayList<Rectangle>();
selected = -1;
// invariant: -1 <= selected < rectangleList.length()
```

5. The last component to be added is a visual component that is added manually. The form contains the declaration of a private inner class DrawingPanel that is a descendent of class JPanel (extends JPanel) providing a panel that has access to private instance variables rectangleList and selected so that in its paintComponent method it can draw all rectangles, the selected rectangle, and the bounding box. Furthermore, the class implements the MouseListener interface (implements MouseListener) that will be used to process mouse events on its panel (in the constructor the panel is added as a mouse listener to itself: this.addMouseListener(this));.

Add a private DrawingPanel to the form at the place indicated in the code:
private DrawingPanel drawingPanel;

Add the following initialization code for this panel to method initAddedComponents before the initialization code for the list (assuming the large left hand panel on the form is named displayPanel)

drawingPanel = new DrawingPanel();
drawingPanel.setSize(100,100);
displayPanel.setLayout(new BorderLayout());
displayPanel.add(drawingPanel);

(Remark drawingPanel.setSize(100,100); is added as a temporary fix to assure that the drawing panel has positive width and height as these are to be used in the code for generating random rectangles fitting within this panel). After displayPanel has been given a BorderLayout manager, drawingPanel is added in the central position of displayPanel (default position in BorderLayout) and as there are no other components added to displayPanel drawingPanel covers displayPanel completely.

6. Whenever there is a (possible) change in rectangleList and/or selected the information in the text fields needs to be updated and the drawing panel needs to be redrawn. Since the need for such an update arises at various positions in the code a private method updateViews is added to the form for this task. The header of this method is already provided. Give a proper implementation of this method (the text fields should be empty in case selected equals −1, otherwise they should contain the coordinates of the selected rectangle and its index; the count text field should always contain the size of the list; redrawing of the drawing panel can be requested by invoking its repaint method). Carefully check whether calls to updateViews need to be inserted in your current code.

7. You should be able to run the project at this stage: only the count text field should show the size of the list (0), the other text fields should be empty, nothing is drawn on the drawing panel.

8. In order to generate rectangles and lists of rectangles two private methods are added to the form

private Rectangle randomRectangle() {...}
private ArrayList<Rectangle> randomRectangleList() {...}

The first should generate a random rectangle fitting within the drawing panel, the second should generate a list of random length (in between 1 and 20) containing random rectangles all fitting within the drawing panel. The form already contains the headers of these methods. Give proper implementations.

Change the initialization code for rectangleList and selected in method initAddedComponents so that the list is initialized to a random list and selected to an arbitrary index within this list.

9. The form contains the following method for drawing a rectangle:

public void drawRectangle(Graphics g, Rectangle r, Color bc, Color fc, String t) {
    Color c = g.getColor(); // save original color
    // draw border with color bc
    g.setColor(bc);
    g.drawRect(r.getXL(), r.getYL(), r.getWidth(), r.getHeight());
    // fill interior with color fc
    g.setColor(fc);
    // ...
Within the graphical context, the rectangle \( r \) is drawn with a border colored \( \textcolor{bc} \) and an interior filled with color \( \textcolor{fc} \) and labeled with text \( t \) starting at the upper left corner outside the rectangle.

10. Using method `drawRectangle`, fill in the parts of method `paintComponent` of inner class `DrawingPanel` that draw all rectangles in the list (border color `\textcolor{Color\_black}` , fill color `\textcolor{Color\_yellow}`) and the selected rectangle (border color `\textcolor{Color\_black}` , fill color `\textcolor{Color\_cyan}`). The rectangles in the list should be drawn in such an order that rectangles with a lower index are on top of rectangles with a higher index. The selected rectangle (if there is one!) should be on top of all rectangles.

11. Running the project now should show you a drawing of all rectangles in the randomly generated list with the randomly selected rectangle on top. Furthermore, the text fields should contain the coordinates and index of the selected rectangle, and the size of the list.

12. Add event handlers to each of the three buttons. To do this for one button change to Design View in the editor and double click the button. The editor changes to Source View showing the generated event handler method for the button. All you need to do is give a proper implementation of this method.

When creating a new random rectangle it should be placed at the end of the list and it should be selected. When replacing the current list with a new nonempty list of rectangles select the first rectangle in the list. After clearing the list there is no selected rectangle. In all three cases the list and the selected rectangle are (possibly) changed so method `updateViews` needs to be called.

Try out your program again. After clicking button `New Rectangle` the newly created rectangle should be on top in a different color.

13. In order to draw the bounding box of the rectangles in the list a private function `Rectangle boundingBox()` is added to the form. Provided the list is not empty it should return the smallest rectangle containing all rectangles from the list. The header of this method is already in the code. Give an implementation using method `hull` from class `Rectangle`.

Using this function implement the part of method `paintComponent` of inner class `DrawingPanel` that draws the bounding box in case the list is not empty (border color `\textcolor{Color\_red}` , fill color `\textcolor{Color\_white}` (the background color of the drawing panel), empty text label).

14. Finally selecting rectangles through mouse clicks on the drawing panel is to be added. A function `int findFirstWindow(int x, int y)` is added to the form that returns the index of the first rectangle in `rectangleList` that contains point \((x, y)\) or \(-1\) if there is no such rectangle in the list. The header of this function is already present, give the implementation.

Next implement method `void mouseClicked(MouseEvent e)` of inner class `DrawingPanel` using function `findFirstWindow`. It should assign to `selected` the index of the first rectangle in the list in which the mouse click occurred or \(-1\) if the mouse click occurred outside of all rectangles. The coordinates at which the mouse click occurred can be obtained by `e.getX()` and `e.getY()`.

15. Run your project. It should now have all the functionality described at the beginning of this assignment. Test whether this is indeed the case.
16. [extra, need not be done] Certain rectangles in the list may not be selectable using the mouse because they are completely covered by the rectangles that precede them in the list. In order to be able to select any rectangle add the possibility to select the rectangle before the selected rectangle in the list or the rectangle after it (provided they exist of course).

References
