Mouse Input in Java Swing

SI@UCF Java / Java GUI (Swing) Recitation

Mouse input is one of the most vital aspects of GUI programs in Java. It is hard to make many useful programs without being able to get input from the user, and that is exactly what we are going to learn how to do.

Documentation:

- https://docs.oracle.com/javase/8/docs/api/java/awt/event/MouseListener.html
- https://docs.oracle.com/javase/tutorial/uiswing/events/mouselistener.html
- https://docs.oracle.com/javase/tutorial/uiswing/events/mousemotionlistener.html

MouseListener and MouseMotionListener:

Java provides two interfaces for processing mouse events:

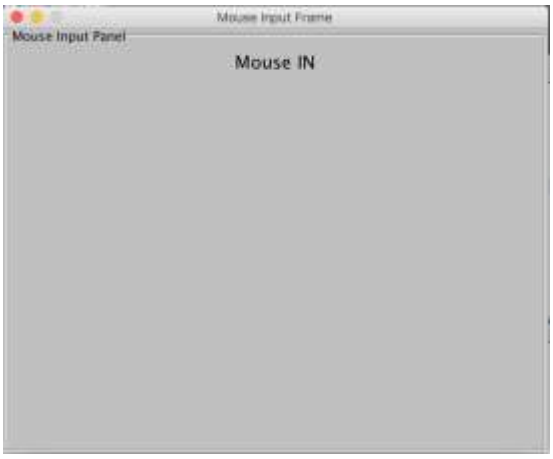
- `MouseListener` receives mouse events on a component. Defined mouse events are **press**, **release**, **click**, **enter**, and **exit**. A mouse event is generated when the mouse is pressed and released clicked (pressed and released), or when the mouse cursor enters or leaves a component.
- `MouseMotionListener` tracks mouse moves and mouse drags.

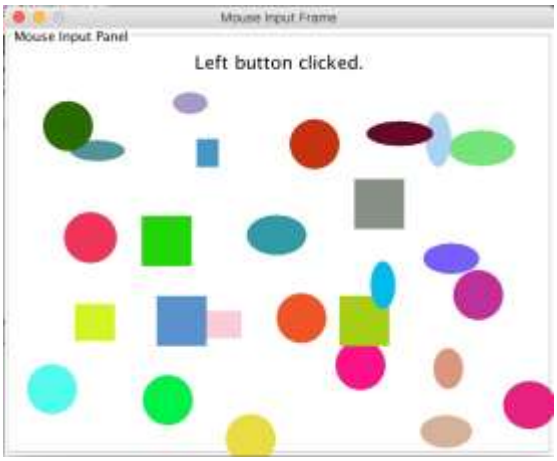
Methods:

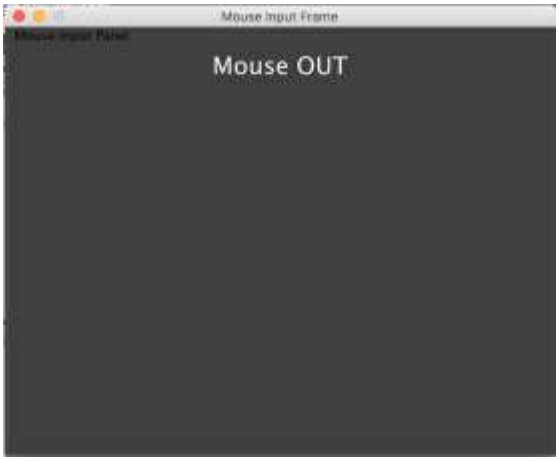
- `void mouseClicked(MouseEvent e)`
  Invoked when the mouse button has been clicked (pressed and released) on a component.
- `void mousePressed(MouseEvent e)`
  Invoked when a mouse button has been pressed on a component.
- `void mouseReleased(MouseEvent e)`
  Invoked when a mouse button has been released on a component.
- `void mouseEntered(MouseEvent e)`
  Invoked when the mouse enters a component.
- `void mouseExited(MouseEvent e)`
  Invoked when the mouse exits a component.

Example:

Let's review the different sections of a simple **MouseInput** NetBeans project that allow us to understand how mouse input events work. The following are some screenshots that illustrate some of the states of the program:

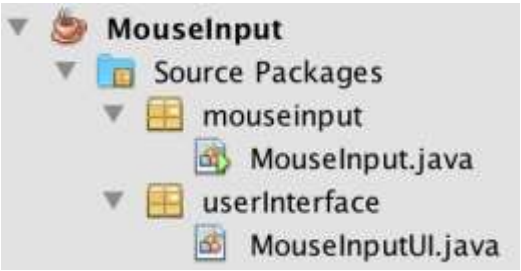






The project has two java files included in two packages:

- **MouseInput.java** in **mouseInput** (main package) and
- **MouseInpuitUI.java** in **userInterface**.



MouseInput.java has the `main()` method, which simply calls the program UI, as follows:

```
package mouseinput;
import userInterface.MouseInputUI;

public class MouseInput
{
    public static void main(String[] args)
    {
        MouseInputUI mouseInputUI = new MouseInputUI();
    }
}
```

MouseInpuitUI.java builds and displays the interactive user interface for the program. Let's analyze its different sections.

The first section lists several required import statements.

```
package userInterface;

// IMPORTS
import javax.swing.JPanel;            // Creates panel to draw figures.
import javax.swing.JFrame;            // Creates window to place panels.
import javax.swing.JLabel;            // Creates mouse-action labels.
import javax.swing.JComponent;        // Creates objects of JComponent type.
import javax.swing.BorderFactory;     // Adds borders with titles to panels.
import java.awt.BorderLayout;         // Sets position of panels in frames.
import java.awt.Color;                // Paints panels and figures.
import java.awt.Dimension;            // Sets object dimensions.
import java.awt.Font;                 // Sets properties for fonts.
import java.awt.Graphics;             // Creates figures.
import java.awt.event.MouseListener;  // Detects mouse actions performed.
import java.awt.event.MouseEvent;     // Indicates which actions to perform.
import java.util.Random;              // Generates objects of Random type.
```

The second section is the **MouseInputUI** class. The class contains member variables, one class constructor, several methods, and an inner class to create a mouse action listener.

```
public class MouseInputUI
{
    // Member variables
    // Constructor
    // Methods
    // Inner class
}
```

Member variables, also called instance variables, are variables that are common to the class.

```
/**
 * Mouse Input Java Swing User Interface.
 */
public class MouseInputUI
{
    // ================
    // MEMBER VARIABLES
    // ================

    // Define objects:
    // - Frame object for main window.
    // - Panel object for mouse input.
    // - Label object to indicate actions performed by the mouse
    //   (i.e., mouse inside/outside panel and mouse button presses).
    private JFrame mouseInputFrame;
    private JPanel mouseInputPanel;
    private JLabel mouseInputPanelLabel;

    // Define listeners.
    private final MouseInputListener mouseInputListener;
```

The constructor is called whenever the class is instantiated.

```
    // ===========
    // CONSTRUCTOR
    // ===========
    public MouseInputUI()
    {
        mouseInputListener = new MouseInputListener();
        initComponents();
    }
```

The class implements several methods. The first one initializes all the components for the UI.

```
    // =======
    // METHODS
    // =======

    /**
     * Initializes all the components for the UI.
     */
    private void initComponents()
    {
        initMouseInputPanel();
        initMouseInputFrame();
        setDimensions();
    }
```

The next method initializes the mouse input panel and a panel label. Several properties of this panel will change depending on varying mouse actions on it. Notice that a mouse listener is implemented on this panel.

```
    /**
     * Initializes the mouse input panel.
     */
    private void initMouseInputPanel()
    {
        // Create panel and label objects.
        mouseInputPanel = new JPanel();
        mouseInputPanelLabel = new JLabel();

        // Add titled border, label, and mouse listener to panel.
        String title = "Mouse Input Panel";
        mouseInputPanel.setBorder(BorderFactory.createTitledBorder(title));
        mouseInputPanel.add(mouseInputPanelLabel);
        mouseInputPanel.addMouseListener(mouseInputListener);
    }
```

The next method builds the frame that contains the panel. A frame can have several panels, but in this program, we only use one panel, which is centered in the frame. We think of a panel as an object defined on a layer on top of a frame.

```
    /**
     * Initializes the mouse input frame holding the panel.
     */
    private void initMouseInputFrame()
    {
        // Create frame and set some properties.
        mouseInputFrame = new JFrame("Mouse Input Frame");
        mouseInputFrame.setLocation(370, 150);
        mouseInputFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mouseInputFrame.setResizable(false);

        // Add panel to the center of the frame and make the frame visible.
        mouseInputFrame.add(mouseInputPanel, BorderLayout.CENTER);
        mouseInputFrame.setVisible(true);
    }
```

The next three methods are used to define dimension properties of the frame, the panel, and any other related components.

```
    /**
     * Sets the size of the given JFrame object.
     * @param object
     * @param width
     * @param height
```

```
     */
    private static void setJFrameSize(JFrame object, int width, int height)
    {
        Dimension dim = new Dimension(width, height);
        object.setSize(dim);
    }
```

```
    /**
     * Sets the size of the given JComponent object.
     * @param object
     * @param width
     * @param height
     */
    private static void setJDimensions(JComponent object, int width, int height)
    {
        Dimension dim = new Dimension(width, height);
        object.setPreferredSize(dim);
        object.setMinimumSize(dim);
        object.setMaximumSize(dim);
    }
```

```
    /**
     * Sets dimensions for all components.
     */
    private void setDimensions()
    {
        setJFrameSize(mouseInputFrame,  550, 450);
        setJDimensions(mouseInputPanel, 550, 450);
    }
```

The next two methods draw and paint a rectangle and an oval of given width and height, at the position that was clicked with the mouse, and with a given color passed defined by separated RGB arguments.

```
    /**
     * Draws a rectangle of given width and height, at the position x,y
     * that was clicked, and painted with a given (RGB) color.
     * If width and height are equal, the figure is a square.
     */
    public void paintRectangle(Graphics g, int width, int height, int x, int y,
                               int R, int G, int B)
    {
        g.setColor(new Color(R,G,B));
        g.fillRect(x, y, width, height);
    }
```

```
    /**
     * Draws an oval of given width and height, at the position x,y
     * that was clicked, and painted with a given (RGB) color.
     * If width and height are equal, the figure is a circle.
     */
    public void paintOval(Graphics g, int width, int height, int x, int y,
                          int R, int G, int B)
    {
        g.setColor(new Color(R,G,B));
        g.fillOval(x, y, width, height);
    }
```

After the methods, we define an inner class to implement action listeners. In this project we only implement a mouse listener. The listener contains several methods that define actions to be taken when the left and right mouse buttons are pressed and clicked (i.e., pressed and released), or when the mouse enters or exits a specific region in the panel.

- Some of the panel properties and labels change when the mouse enters and exits the panel.
- An oval of random color and dimensions is drawn with a *left mouse click*. If the control key is pressed while left clicking the mouse, a circle (width == height == diameter) is drawn instead.
- A rectangle of random color and dimensions is drawn with a *right mouse click*. If the control key is pressed while right clicking the mouse, an square (width == height) is drawn instead.

```
    // ================================
    // INNER CLASSES TO CREATE LISTENERS
    // ================================

    /**
     * Creates mouse input listener for panel.
     */
    private class MouseInputListener implements MouseListener
    {
        @Override
        public void mouseClicked(MouseEvent e)
        {
            // If the shift key is pressed, reset the panel and its label.
            if (e.isShiftDown())
            {
                mouseInputPanelLabel.setText("");
                mouseInputPanel.repaint();
            }

            // Get the position (col,row) that has been clicked with the mouse.
            int x = e.getX();
```

```
            int y = e.getY();

            // Create graphics object to paint figures on it.
            Graphics g = mouseInputPanel.getGraphics();

            // Create random object for random number generation.
            Random r = new Random();
```

```
            // Set actions for right and left mouse clicks (press & release).
            // - If right click, draw rectangles/squares of random size/color.
            // - If left click, draw ovals/circles of random size/color.
            int width, height;
            if (e.isMetaDown())
            {
                mouseInputPanelLabel.setText("Right button clicked.");

                // If the control key is pressed, draw squares. Otherwise,
                // draw rectangles of random dimensions (20 <= w,h < 60).
                if (e.isControlDown())
                {
                    width  = 50;
                    height = 50;
                }
                else
                {
                    width  = r.nextInt(40) + 20;
                    height = r.nextInt(40) + 20;
                }
                paintRectangle(g, width, height, x, y,
                               r.nextInt(255), r.nextInt(255), r.nextInt(255));
            }
```

```
            else
            {
                mouseInputPanelLabel.setText("Left button clicked.");

                // If the control key is pressed, draw circles. Otherwise,
                // draw ovals of random dimensions (20 <= diameter < 70).
                if (e.isControlDown())
                {
                    width  = 50;
                    height = 50;
                }
                else
                {
                    width  = r.nextInt(50) + 20;
                    height = r.nextInt(40) + 20;
                }
                paintOval(g, width, height, x, y,
                          r.nextInt(255), r.nextInt(255), r.nextInt(255));
            }
```

```
            mouseInputPanelLabel.setFont(new Font(Font.SANS_SERIF,
                                                  Font.PLAIN, 18));
        }

        @Override
        public void mousePressed(MouseEvent e)
        {
            mouseInputPanel.setBackground(Color.WHITE);

            if (e.isMetaDown())
                mouseInputPanelLabel.setText("Right button pressed.");
            else
                mouseInputPanelLabel.setText("Left button pressed.");

            mouseInputPanelLabel.setFont(new Font(Font.SANS_SERIF,
                                                  Font.PLAIN, 18));
        }

        @Override
        public void mouseReleased(MouseEvent e)
        {
            mouseInputPanelLabel.setText("Mouse button released.");
            mouseInputPanelLabel.setFont(new Font(Font.SANS_SERIF,
                                                  Font.PLAIN, 18));
        }

        @Override
        public void mouseEntered(MouseEvent e)
        {
            // Paint the panel of the given color when mouse enters the panel.
            mouseInputPanel.setBackground(Color.LIGHT_GRAY);
            mouseInputPanelLabel.setText("Mouse IN");
            mouseInputPanelLabel.setFont(new Font(Font.SANS_SERIF,
                                                  Font.PLAIN, 18));
            mouseInputPanelLabel.setForeground(Color.BLACK);
        }

        @Override
        public void mouseExited(MouseEvent e)
        {
            // Paint the panel of the given color when mouse exits the panel.
            mouseInputPanel.setBackground(Color.DARK_GRAY);
            mouseInputPanelLabel.setText("Mouse OUT");
            mouseInputPanelLabel.setFont(new Font(Font.SANS_SERIF,
                                                  Font.PLAIN, 24));
            mouseInputPanelLabel.setForeground(Color.WHITE);
        }
    }
}
```