**Further Object Oriented Programming**

U08026

David Sutton

Department of Computing and Communication Technologies

# Java GUI Programming

# Introduction

After you have completed this session, you will understand how to create applications that have simple graphical user interfaces.

**You should submit your solution to Exercise 9.**

# Exercise 1: The Hello World Application in Swing

Create a new NetBeans project. Add the Hello World Main class from the lecture slides (set out in ). Run the project and check that it works

Table Hello World GUI

|  |
| --- |
| import javax.swing.\*;  public class Main {  public static void main(String[] args) {  JFrame frame = new JFrame("HelloWorldSwing");  frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  JLabel label = new JLabel("Hello World");  frame.getContentPane().add(label);  frame.pack();  frame.setVisible(true);  }  } |

# Exercise 2: Layout of the Time Application

Create a new project. Write code that uses a horizontal BoxLayout to display a JButton and a JLabel next to each other. Try to work out how do this by reading the lecture slides, but the code is set out in Table 2 if you get stuck.

Run the project. You will see an application that displays the button and label. However the button does not do anything yet. N.B. the GUI maybe quite small.

Table Laying out Components

|  |
| --- |
| import java.awt.Container;  import javax.swing.BoxLayout;  import javax.swing.JButton;  import javax.swing.JFrame;  import javax.swing.JLabel;  public class TimeApp extends JFrame {  JButton button = new JButton("Get Time");  JLabel label = new JLabel(" Button last pressed at ..................");  public static void main(String[] args) {  new TimeApp();  }  public TimeApp() {  super("Time");  setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  Container contentPane = getContentPane();  contentPane.setLayout(new BoxLayout(contentPane, BoxLayout.X\_AXIS));  contentPane.add(button);  contentPane.add(label);  pack();  setVisible(true);  }  } |

# Exercise 3: Event Handling in the Time Application

Modify your TimeApp class so that it implements the ActionListener interface, and listens to events emanating from the JButton. The code should look like that set out in Table 3. The modified code is shown in bold.

Table TimeApp with Event Handling

|  |
| --- |
| import java.awt.Container;  **import java.awt.event.ActionEvent;**  **import java.awt.event.ActionListener;**  **import java.text.DateFormat;**  **import java.util.Date;**  import javax.swing.BoxLayout;  import javax.swing.JButton;  import javax.swing.JFrame;  import javax.swing.JLabel;  public class TimeApp extends JFrame **implements ActionListener** {  JButton button = new JButton("Get Time");  JLabel label = new JLabel(" Button last pressed at ..................");  public static void main(String[] args) {  new TimeApp();  }  public TimeApp() {  super("Time");  setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  **button.addActionListener(this);**  Container contentPane = getContentPane();  contentPane.setLayout(new BoxLayout(contentPane, BoxLayout.X\_AXIS));  contentPane.add(button);  contentPane.add(label);  pack();  setVisible(true);  }  **public void actionPerformed(ActionEvent ae) {**  **DateFormat df = DateFormat.getTimeInstance();**  **label.setText(" Button last pressed at " + df.format(new Date()));**  **}**  } |

# Exercise 4: Layout of the Stopwatch Application

Create a new project. Add a new class that displays the user interface for the stopwatch application set out in the lecture slides, but doesn’t handle events yet. The required code is set out in Table 4. Run it and check that what you see makes sense.

Table Layout for Stopwatch Application

|  |
| --- |
| import java.awt.Container;  import javax.swing.BoxLayout;  import javax.swing.JButton;  import javax.swing.JFrame;  import javax.swing.JPanel;  import javax.swing.JTextField;  public class StopWatch {  private JFrame frame =  new JFrame("Digital Stopwatch");  private JPanel buttonPanel = new JPanel();  private JButton startButton = new JButton("Start");  private JButton stopButton = new JButton("Stop");  private JTextField timeField = new JTextField("");  private long startTime;  public StopWatch() {  buttonPanel.setLayout(new BoxLayout(buttonPanel, BoxLayout.X\_AXIS));  buttonPanel.add(startButton);  buttonPanel.add(stopButton);  stopButton.setEnabled(false);  Container contentPane = frame.getContentPane();  contentPane.setLayout(new BoxLayout(contentPane, BoxLayout.Y\_AXIS));  timeField.setEditable(false);  contentPane.add(timeField);  contentPane.add(buttonPanel);  frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  frame.pack();  frame.setVisible(true);  }  public static void main(String[] args) {  new StopWatch();  }  } |

# Exercise 5: Event Handling in the Stopwatch Application

Now modify your stopwatch application so that it creates a Timer, and handles ActionEvents generated by both the timer and the two buttons. The modifications necessary are marked in bold in Table 5. Run the application and check that it behaves correctly.

Table Stopwatch with Event Handling Code

|  |
| --- |
| import java.awt.Container;  **import java.awt.event.ActionEvent;**  **import java.awt.event.ActionListener;**  **import java.text.NumberFormat;**  import javax.swing.BoxLayout;  import javax.swing.JButton;  import javax.swing.JFrame;  import javax.swing.JPanel;  import javax.swing.JTextField;  **import javax.swing.Timer;**  public class StopWatch **implements ActionListener** {  private JFrame frame =  new JFrame("Digital Stopwatch");  private JPanel buttonPanel = new JPanel();  private JButton startButton = new JButton("Start");  private JButton stopButton = new JButton("Stop");  private JTextField timeField = new JTextField("");  **private Timer timer;**  **private long startTime;**  public StopWatch() {  buttonPanel.setLayout(new BoxLayout(buttonPanel, BoxLayout.X\_AXIS));  buttonPanel.add(startButton);  buttonPanel.add(stopButton);  stopButton.setEnabled(false);  **startButton.addActionListener(this);**  **stopButton.addActionListener(this);**  Container contentPane = frame.getContentPane();  contentPane.setLayout(new BoxLayout(contentPane, BoxLayout.Y\_AXIS));  timeField.setEditable(false);  contentPane.add(timeField);  contentPane.add(buttonPanel);  **timer = new Timer(100, this);**  frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  frame.pack();  frame.setVisible(true);  }  **public void actionPerformed(ActionEvent e) {**  **if (e.getSource() == startButton) {**  **startTime = e.getWhen();**  **timer.start();**  **stopButton.setEnabled(true);**  **startButton.setEnabled(false);**  **}**  **if (e.getSource() == timer) {**  **setTimeField(e.getWhen());**  **}**  **if (e.getSource() == stopButton) {**  **setTimeField(e.getWhen());**  **timer.stop();**  **stopButton.setEnabled(false);**  **startButton.setEnabled(true);**  **}**  **}**  **private void setTimeField(long time) {**  **long elapsed = time - startTime;**  **long centisecs = elapsed / 10;**  **long seconds = centisecs / 100;**  **long minutes = seconds / 60;**  **long hours = minutes / 60;**  **NumberFormat nf = NumberFormat.getNumberInstance();**  **nf.setMinimumIntegerDigits(2);**  **String timeStr = "" + nf.format(hours) + ":"**  **+ nf.format(minutes % 60) + ":"**  **+ nf.format(seconds % 60) + "."**  **+ nf.format(centisecs % 100);**  **timeField.setText(timeStr);**  **}**  public static void main(String[] args) {  new StopWatch();  }  } |

# Exercise 6: Sketch Pad Application

Create a new project and implement the sketch pad application from the lecture notes. This will require the creation of two clases, SketchPanel and Sketch, which are set out in and . Run the project and check that it behaves as it should

Table The SketchPanel Class

|  |
| --- |
| import java.awt.Color;  import java.awt.Graphics;  import java.awt.Point;  import java.util.List;  import javax.swing.JPanel;  public class SketchPanel extends JPanel {  private static final int DIAMETER = 16;  private List<Point> points;  public SketchPanel(List<Point> points) {  this.points = points;  }  @Override  public void paintComponent(Graphics g) {  super.paintComponent(g);  g.setColor(Color.RED);  int size = points.size();  int[] x = new int[size];  int[] y = new int[size];  int i = 0;  for (Point p : points) {  x[i] = p.x;  y[i] = p.y;  i++;  }  g.drawPolyline(x, y, i);  if (size > 0) {  g.fillOval(x[size - 1] - DIAMETER / 2, y[size - 1] - DIAMETER / 2,  DIAMETER, DIAMETER);  }  }  } |

Table The Sketch Class

|  |
| --- |
| import java.awt.Container;  import java.awt.Dimension;  import java.awt.Point;  import java.awt.event.MouseEvent;  import java.awt.event.MouseListener;  import java.util.ArrayList;  import javax.swing.BoxLayout;  import javax.swing.JFrame;  public class Sketch extends JFrame implements MouseListener {  private static final Dimension SKETCH\_SIZE = new Dimension(360, 360);  ArrayList<Point> points = new ArrayList<Point>();  SketchPanel sketchPanel = new SketchPanel(points);  public Sketch() {  super("Sketch");  Container contentPane = getContentPane();  contentPane.setLayout(new BoxLayout(contentPane, BoxLayout.Y\_AXIS));  sketchPanel.addMouseListener(this);  sketchPanel.setPreferredSize(SKETCH\_SIZE);  contentPane.add(sketchPanel);  setDefaultCloseOperation(EXIT\_ON\_CLOSE);  pack();  setResizable(false);  setVisible(true);  }  public void mouseClicked(MouseEvent e) {  points.add(e.getPoint());  repaint();  }  public void mousePressed(MouseEvent e) {  }  public void mouseReleased(MouseEvent e) {  }  public void mouseEntered(MouseEvent e) {  }  public void mouseExited(MouseEvent e) {  }  public static void main(String[] args) {  new Sketch();  }  } |

# Exercise 7: Add a Pause Button to the Stopwatch

Modify the stopwatch code from Exercise 5 so that the stopwatch has a pause button, as in . The pause button should behave in the same way as the stop button except that the clock is not reset to zero after a pause. In other words if you pressed start, let the stopwatch run for five seconds, and then pressed Pause, the watch would read ‘00.00.05.00’ (as it would do if you had pressed start). However if you leave it paused for say 2 seconds and then hit ‘start’ it begins counting from ‘00.00.07.00’ (because seven seconds have elapsed since you initially pressed start).

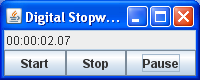


Figure Stopwatch with Pause Button

# Exercise 8 Sketch Application with a Delete Button

Add a ‘Delete’ button to the sketch pad application you created in Exercise 6. The user interface should be as in . The effect of pressing the delete button should be to delete the last point in the sketch (the one indicated by the red blob) and then to repaint the sketch.

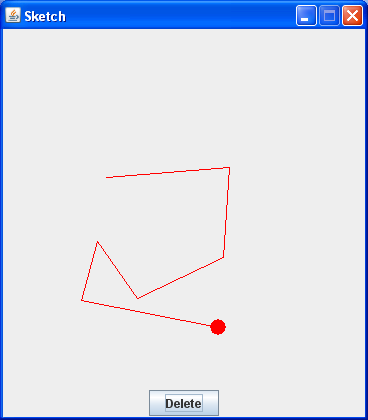


Figure Sketch Application with Delete Button

# Exercise 9: Sketch Application with a Mouse Motion Listener

Take the code from Exercise 8 and modify it so that there are two text fields at the bottom of the user interface which indicate the position of the cursor, as shown in . The X position of the cursor is the vertical distance, in pixels, between the cursor and the top of the sketch. The Y position is the distance, in pixels, between the cursor and the left hand side of the sketch. The fields should be updated every time the cursor is moved.

I would suggest that you put the two text fields, along with the delete button, into a JPanel (in the same way as you did for the buttons in the stop watch application).

You will also need to make the Sketch class implement the MouseMotionListener interface. This interface defines two methods

* mouseMoved(MouseEvent e)
* mouseDragged(MouseEvent e)

the mouseMoved method is called whenever the mouse is moved. You can obtain the coordinates of the point to which it was moved by calling e.getX() and e.getY(), where e is the MouseEvent argument passed to the method.

The mouseDragged method must be implemented, but its implementation does not need to do anything.

You also need to add call sketchPanel.addMouseMotionListener in order to make the sketchPanel listen to movements of the mouse.

You can find documentation for all the classes and interfaces mentioned here by visiting <http://docs.oracle.com/javase/7/docs/api/>

# What you should submit

Please submit TWO files:

* The source code for Exercise 9 as a zip file.
* The same source code pasted into a single Word file.

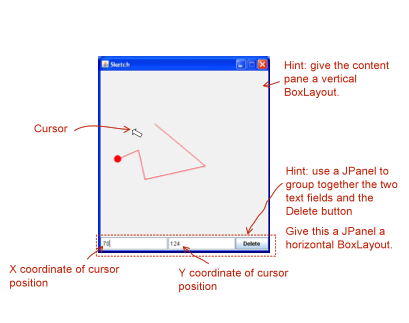


Figure Sketch Application with Mouse Motion Listener