Learning Notes for Unit 5 – Part A

GUI Programming:

Computer programs are far more user friendly when they contain graphical components to interact with users. This unit will introduce many common GUI components and how they are used in programs to enhance user interaction. You are encouraged to write programs to implement the code in this unit to see these components in action.

Common Components:

JFrame:

A JFrame is the basic starting block to create a GUI program. To start, you would create a program skeleton like we did back in units 1 and 2. Above the main method, you will need to import the JFrame class within the swing library. Within the main method, you will create a new JFrame object and set its size and visibility.

For example, you would define your frame composition in JDisplayFrame.java:

```
import javax.swing.JFrame;

public class JDisplayFrame extends JFrame {

        public JDisplayFrame() {

                super("Frame Title");

                setSize(500,500);

                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        }

}
```

Then you would create a program to implement your frame (e.g. JRunFrame.java):

```
public class JRunFrame {

        public static void main(String[] args) {

                JDisplayFrame myFrame =  new JDisplayFrame();

                myFrame.setVisible(true);

        }

}
```

To add content to the frame, you will need to use a layout manager. These layout managers control where content is placed within your frame. The simplest layout manager is FlowLayout() which places components beside each other in a row and when the row reaches the edge of the frame it will add a new row.

To add the FlowLayout manager to your frame, you need to import it and add the manager to your frame object.

You would modify JDisplayFrame.java as follows:

```
import javax.swing.JFrame;
import java.awt.FlowLayout;

public class JDisplayFrame extends JFrame {

	public JDisplayFrame() {

		super("Frame Title");

		setSize(500,500);

		setLayout(new FlowLayout());

		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

	}

}
```

Once you have indicated the layout type you will be using, you can add content to your frame. The most common components you will use are JLabels, JTextFields, and JButtons which will put labels, text boxes, and boxes in your program.

JLabel:

To add a label to your frame, you need to declare a JLabel object (you may need an import statement) and add it to the frame.

```
import javax.swing.JFrame;
import java.awt.FlowLayout;
import javax.swing.JLabel;

public class JDisplayFrame extends JFrame {

        private JLabel myLabel = new JLabel("hello");

        public JDisplayFrame() {

                super("Frame Title");

                setSize(500,500);

                setLayout(new FlowLayout());

                add(myLabel);

                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        }
```

JTextField:

To add a text box to your frame, you need to declare a JTextField object of a specific width (you may need an import statement) and add it to the frame.

```
import javax.swing.JFrame;
import java.awt.FlowLayout;
import javax.swing.JLabel;
import javax.swing.JTextField;

public class JDisplayFrame extends JFrame {

        private JLabel myLabel = new JLabel("hello");

        private JTextField myText = new JTextField(15);

        public JDisplayFrame() {

                super("Frame Title");

                setSize(500,500);

                setLayout(new FlowLayout());

                add(myLabel);

                add(myText);

                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        }

}
```

JButton:

To add a clickable button to your frame, you need to declare a JButton object (you may need an import statement) and add it to the frame. You will also need to add an **ActionListener** that will enable your program to react when a button action occurs. You must also define a method called **actionPerformed** in which you will add the code that is run when a specific button is pressed.

```
import javax.swing.JFrame;
import java.awt.FlowLayout;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class JDisplayFrame extends JFrame implements ActionListener {

        private JLabel myLabel = new JLabel("hello");
        private JTextField myText = new JTextField(15);
        private JButton button1 = new JButton ("click me");

        public JDisplayFrame() {

                super("Frame Title");
                setSize(500,500);
                setLayout(new FlowLayout());
                add(myLabel);
                add(myText);
                add(button1);
                button1.addActionListener(this);

                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        }
        public void actionPerformed(ActionEvent e) {

                if (e.getSource()==button1) {

                        myLabel.setText(myText.getText());

                }

        }
}
```

Throughout this course, we have kept the implementation classes (i.e. the programs) separated from the storage classes. In this GUI program, the implementation class serves to instantiate the custom Frame class. These two programs can be combined as follows:

```
import javax.swing.JFrame;
import java.awt.FlowLayout;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class JDisplayAndRunFrame extends JFrame implements ActionListener {

        private JLabel myLabel = new JLabel("hello");
        private JTextField myText = new JTextField(15);
        private JButton button1 = new JButton ("click me2");

        public JDisplayAndRunFrame() {

                super("New Frame Title");
                setSize(500,500);
                setLayout(new FlowLayout());
                add(myLabel);
                add(myText);
                add(button1);
                button1.addActionListener(this);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        }
        public void actionPerformed(ActionEvent e) {

                if (e.getSource()==button1) {

                        myLabel.setText(myText.getText());

                }

        }

        public static void main(String[] args) {

                JDisplayAndRunFrame myFrame =  new JDisplayAndRunFrame();
                myFrame.setVisible(true);

        }


}
```

Please read the rest of the assigned chapter to learn how to implement other commonly encountered GUI controls.

Learning Notes for Unit 5 – Part B

Content Panes:

Within the frame, you can place containers which are essentially boxes to put content into. Container allow you to more easily group related components together in your GUI. To implement, declare a Container object and add your components to the container rather than directly to the Frame.

```
import javax.swing.JFrame;
import java.awt.FlowLayout;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.Container;

public class JwithContentPane extends JFrame implements ActionListener {

        private JLabel myLabel = new JLabel("hello");
        private JTextField myText = new JTextField(15);
        private JButton button1 = new JButton ("click me");
        private Container con = getContentPane();

        public JwithContentPane() {

                super("Frame Title");
                setSize(500,500);
                con.setLayout(new FlowLayout());
                con.add(myLabel);
                con.add(myText);
                con.add(button1);
                button1.addActionListener(this);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        }

        public void actionPerformed(ActionEvent e) {

                if (e.getSource()==button1) { myLabel.setText(myText.getText()); }

        }

        public static void main(String[] args) {

                JwithContentPane myFrame =  new JwithContentPane ();
                myFrame.setVisible(true);

        }

}
```

Layout Managers:

There are a variety of other layout managers that can be used to control where content is displayed.

BorderLayout: (5 possible regions for content – top, bottom, left, right, center)

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class JBorderExample extends JFrame implements ActionListener {

        private JButton button1 = new JButton ("North"),
                  button2 = new JButton ("South"),
                  button3 = new JButton ("East"),
                        button4 = new JButton ("West");
        private JLabel myLabel = new JLabel("");
        private Container con = getContentPane();

        public JBorderExample() {

                super("Border Example");
                setSize(500,500);
                con.setLayout(new BorderLayout());
                con.add(button1, BorderLayout.NORTH);
                con.add(button2, BorderLayout.SOUTH);
                con.add(button3, BorderLayout.EAST);
                con.add(button4, BorderLayout.WEST);
                con.add(myLabel, BorderLayout.CENTER);

                button1.addActionListener(this);
                button2.addActionListener(this);
                button3.addActionListener(this);
        button4.addActionListener(this);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    }

    public void actionPerformed(ActionEvent e) {

        if (e.getSource()==button1) { myLabel.setText("North Pressed"); }
        if (e.getSource()==button2) { myLabel.setText("South Pressed"); }
        if (e.getSource()==button3) { myLabel.setText("East Pressed");  }
        if (e.getSource()==button4) { myLabel.setText("West Pressed");  }

    }

    public static void main(String[] args) {

        JBorderExample myFrame =  new JBorderExample();
        myFrame.setVisible(true);

    }

}
```

GridLayout: (define the number of rows and columns. Controls are added to each grid position from left to right, top to bottom)

```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class JGridExample extends JFrame implements ActionListener {

        private JButton button1 = new JButton ("Button 1"),
                  button2 = new JButton ("Button 2"),
                  button3 = new JButton ("Button 3");

        private JLabel label1 = new JLabel(""),
                    label2 = new JLabel(""),
                    label3 = new JLabel("");

        private Container con = getContentPane();

        public JGridExample() {

                super("Grid Example");
                setSize(500,500);

                con.setLayout(new GridLayout(3,2));
                con.add(button1);
                con.add(label1);
                con.add(button2);
                con.add(label2);
                con.add(button3);
                con.add(label3);
                button1.addActionListener(this);
                button2.addActionListener(this);
                button3.addActionListener(this);

                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        }

        public void actionPerformed(ActionEvent e) {

                if (e.getSource()==button1) { label1.setText("B1 Pressed"); }
                if (e.getSource()==button2) { label2.setText("B2 Pressed"); }
                if (e.getSource()==button3) { label3.setText("B3 Pressed"); }

        }

        public static void main(String[] args) {

                JGridExample myFrame =  new JGridExample();
                myFrame.setVisible(true);

        }

}
```

JPanel:

In the previous examples, you have seen how the overall placement of components can be controlled using various layout managers. Unfortunately, each layout "compartment" is only capable of having a single item placed into it. Fortunately, JPanels are available to place multiple components in and the panel can be placed into the desired "compartment"

More Event Handling:

Mouse Events: (The following example will display the coordinate at the click point)

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;

public class JMouser extends JFrame implements MouseListener {

        private JLabel mouseLabel = new JLabel("");
        private Container con = getContentPane();

        public JMouser() {

                super("Mouse Events");
                setSize(500,500);

                con.setLayout(new FlowLayout());
                con.add(mouseLabel);
                addMouseListener(this);

                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        }

        public static void main(String[] args) {

                JMouser myFrame =  new JMouser();
                myFrame.setVisible(true);

        }
        public void mouseClicked(MouseEvent evt) {      }

        public void mouseEntered(MouseEvent evt) {     }

        public void mouseExited(MouseEvent evt) {       }

        public void mouseReleased(MouseEvent evt) { }

        public void mousePressed(MouseEvent evt) {

                int x = evt.getX();

                int y = evt.getY();

                mouseLabel.setText(x+","+y);

        }

}
```

Keyboard Events: (The following example will display the key that has been pressed)

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

public class JKeyBo extends JFrame implements KeyListener {

        private JLabel keyLabel = new JLabel("");
        private Container con = getContentPane();

        public JKeyBo() {

                super("Keyboard Events");
                setSize(500,500);

                con.setLayout(new FlowLayout());
                con.add(keyLabel);
                addKeyListener(this);

                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        }

        public static void main(String[] args) {

                JKeyBo myFrame =  new JKeyBo();
                myFrame.setVisible(true);

        }
        public void keyPressed(KeyEvent evt) {

                char c = evt.getKeyChar();
                keyLabel.setText(c+" has been pressed");

        }

        public void keyReleased(KeyEvent evt) { }

        public void keyTyped(KeyEvent evt) { }

}
```

Please read the rest of the assigned chapter to learn how to implement other controls and respond to various events.