**Name**

**Advanced Programming in Java**

**Creating Basic GUI Applications**

**Lab Exercise 10/31/2022**

In this course, we have been creating applications that were non-graphical in nature. This might be fine for academic endeavors, but in the real world, it is important too have software that can be marketed. This means a Graphical User Interface (GUI). In this exercise you will start by creating a basic application that creates a simple graphical window. Any resources you require can be found in the Data Files Folder on the server.

import javax.swing.JFrame;

import java.awt.Container;

// other import statements here as needed

public class ShellGUIApplication extends JFrame

{

private Container contents;

// declare other instance variables

// constructor

public ShellGUIApplication( )

{

// call JFrame constructor with title bar text

super( "A Shell GUI Application" );

// get container for components

contents = getContentPane( );

// set the layout manager

// instantiate GUI components and other instance variables

// add GUI components to the content pane

// set original size of window

setSize( 300, 200 );

// make window visible

setVisible( true );

}

public static void main( String [] args )

{

ShellGUIApplication basicGui = new ShellGUIApplication( );

basicGui.setDefaultCloseOperation( JFrame.EXIT\_ON\_CLOSE );

}

}

This application will create a basic window that is constructed from the main method. The sole purpose of the main method in this application is to construct the ShellGUIApplication object.

Several operations still need to be performed in order to make this a meaningful application; specifically:

1. Setting the layout manager
2. Constructing GUI objects
3. Adding GUI components to the content pane
4. Adding the appropriate “listeners” to the GUI objects

Now let us create an application with some actual GUI components; a system login screen.

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JTextField;

import javax.swing.JPasswordField;

import javax.swing.JTextArea;

import java.awt.Container;

import java.awt.FlowLayout;

import java.awt.Color;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

public class Login extends JFrame implements ActionListener

{

private Container contents;

private JLabel idLabel, passwordLabel, message;

private JTextField id;

private JPasswordField password;

private JTextArea legal;

// Constructor

public Login( )

{

super( "Login Screen" );

contents = getContentPane( );

contents.setLayout( new FlowLayout( ) );

idLabel = new JLabel( "Enter id" ); // label for ID

id = new JTextField( "", 12 ); // instantiate ID text field

passwordLabel = new JLabel( "Enter password" ); // password label

password = new JPasswordField( 8 ); // instantiate password field

password.setEchoChar( '?' ); // set echo character to '?'

message = new JLabel( "Log in above" ); // label to hold messages

// instantiate JTextArea with legal warning

legal = new JTextArea( "Warning: Any attempt to illegally\n"

+ "log in to this server is punishable by law.\n"

+ "This corporation will not tolerate hacking,\n"

+ "virus attacks, or other malicious acts." );

legal.setEditable( false ); // disable typing in this field

// add all components to the window

contents.add( idLabel );

contents.add( id );

contents.add( passwordLabel );

contents.add( password );

contents.add( message );

contents.add( legal );

// add event handler as listener for ID and password fields

id.addActionListener( this );

password.addActionListener( this );

setSize( 250,200 );

setVisible( true );

} // end of constructor method

public void actionPerformed( ActionEvent e )

{

if ( id.getText( ).equals( "open" )

&& ( new String( password.getPassword( ) ) ).equals( "sesame" ) )

{

message.setForeground( Color.BLACK );

message.setText( "Welcome!" );

}

else

{

message.setForeground( Color.RED );

message.setText( "Sorry: wrong login" );

}

}

public static void main( String [] args )

{

Login login = new Login( );

login.setDefaultCloseOperation( JFrame.EXIT\_ON\_CLOSE );

}

}

This is an example of an “event driven” program. A couple of things to note; all imports can be replaced by:

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

Also note that the Login class in addition to extending JFrame (inheritance) also implements the interface ActionListener. The ActionListener class has an abstract method called actionPerformed which must be written with any class that implements this interface. The code in actionPerformed is executed any time the “Enter” key is hit with the cursor in either the id or password fields.

Now let’s do something with a different type of component.

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class MixingColors extends JFrame implements ItemListener

{

private Container contents;

private JCheckBox red, green, blue;

private int redValue, greenValue, blueValue;

private JLabel label;

public MixingColors( )

{

super( "Selecting a color" );

contents = getContentPane( );

contents.setLayout( new FlowLayout( ) );

red = new JCheckBox( "red" );

green = new JCheckBox( "green" );

blue = new JCheckBox( "blue" );

label = new JLabel( "Watch my background" );

label.setOpaque( true );

label.setForeground( Color.GRAY );

label.setBackground( new Color ( 0, 0, 0 ) );

contents.add( red );

contents.add( green );

contents.add( blue );

contents.add( label );

red.addItemListener( this);

green.addItemListener( this );

blue.addItemListener( this );

setSize( 225,200 );

setVisible( true );

}

public void itemStateChanged( ItemEvent ie )

{

if ( ie.getSource( ) == red )

{

if ( ie.getStateChange( ) == ItemEvent.SELECTED )

redValue = 255;

else

redValue = 0;

}

else if ( ie.getSource( ) == green )

{

if ( ie.getStateChange( ) == ItemEvent.SELECTED )

greenValue = 255;

else

greenValue = 0;

}

else if ( ie.getSource( ) == blue )

{

if ( ie.getStateChange( ) == ItemEvent.SELECTED )

blueValue = 255;

else

blueValue = 0;

}

label.setBackground(new Color( redValue, greenValue, blueValue ) );

}

public static void main( String [] args )

{

MixingColors mc = new MixingColors( );

mc.setDefaultCloseOperation( JFrame.EXIT\_ON\_CLOSE );

}

}

This example is with checkbox components. Now let try something similar with radiobuttons. In this example we will still use an ItemListener interface which requires us to write an itemStateChanged method for our ChangingColors class.

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class ChangingColors extends JFrame implements ItemListener

{

private Container contents;

private JRadioButton red, green, blue;

private ButtonGroup colorGroup;

private JLabel label;

private Color selectedColor = Color.RED;

public ChangingColors( )

{

super( "Selecting a color" );

contents = getContentPane( );

contents.setLayout( new FlowLayout( ) );

red = new JRadioButton( "red", true );

green = new JRadioButton( "green" );

blue = new JRadioButton( "blue" );

label = new JLabel( "Watch my background" );

label.setForeground( Color.GRAY );

label.setOpaque( true );

label.setBackground( selectedColor );

contents.add( red );

contents.add( green );

contents.add( blue );

contents.add( label );

// create button group

colorGroup = new ButtonGroup( );

colorGroup.add( red );

colorGroup.add( green );

colorGroup.add( blue );

red.addItemListener( this );

green.addItemListener( this );

blue.addItemListener( this );

setSize( 225, 200 );

setVisible( true );

}

public void itemStateChanged( ItemEvent ie )

{

if ( ie.getSource( ) == red )

selectedColor = Color.RED;

else if ( ie.getSource( ) == green )

selectedColor = Color.GREEN;

else if ( ie.getSource( ) == blue )

selectedColor = Color.BLUE;

label.setBackground( selectedColor );

}

public static void main( String [] args )

{

ChangingColors cc = new ChangingColors( );

cc.setDefaultCloseOperation( JFrame.EXIT\_ON\_CLOSE );

}

}

Finally as out last example, let us show how to add bitmapped images to our applications.

import javax.swing.\*;

import java.awt.\*;

import javax.swing.event.\*;

public class FoodSamplings extends JFrame implements ListSelectionListener

{

private Container contents;

private JList countries;

private JLabel foodImage;

private String [] countryList = { "France", "Greece", "Italy", "Japan", "USA" };

private ImageIcon [] foods =

{ new ImageIcon( "cheese.jpg" ),

new ImageIcon( "fetaSalad.jpg" ),

new ImageIcon( "pizza.jpg" ),

new ImageIcon( "sushi.jpg" ),

new ImageIcon( "hamburger.jpg" ) };

public FoodSamplings( )

{

super( "Food samplings of various countries" );

contents = getContentPane( );

contents.setLayout( new FlowLayout( ) );

// instantiate the components

countries = new JList( countryList );

foodImage = new JLabel( foods[0] );

// allow single selections only

countries.setSelectionMode( ListSelectionModel.SINGLE\_SELECTION );

countries.setSelectedIndex( 0 );

// add components to the content pane

contents.add( countries );

contents.add( foodImage );

countries.addListSelectionListener( this );

setSize( 350, 150 );

setVisible( true );

}

public void valueChanged( ListSelectionEvent lse )

{

foodImage.setIcon( foods[countries.getSelectedIndex( )] );

}

public static void main( String [] args )

{

FoodSamplings fs = new FoodSamplings( );

fs.setDefaultCloseOperation( JFrame.EXIT\_ON\_CLOSE );

}

}

Note that in this program, we use a JList to store our country names and we also create an array if ImageIcon objects.