* The **Java Foundation Classes (JFC)** are used for creating GUI programs
* The **Abstract Windowing Toolkit (AWT)** classes use peer classes in the operating system to draw elements on the screen; thus a program is limited to what is available on a specific OS, and behavior may vary
* The **Swing** classes introduced in Java 2 draw most of their own elements on the screen; these offer consistent appearance and behavior across multiple operating systems, and they are customizable
* GUI programs are **event-driven**
* An **event object** is generated when the user interacts with a GUI element in a particular manner
* A program must have an **event listener** ready to respond to the creation of the event object
* A JFrame object is an example of a **container**
* Containers that are displayable as windows are known as **frames**
* A program can package the method calls in another class by using **inheritance**
* The header of the class should have an extends clause with the name of the class being extended (here, JFrame)
* **Embedded main Methods**
* A class may set up various parts of a window using helper methods (usually private)
* The embedded main method instantiates the window class
* A programmer usually defines an event listener class as a private **inner class** nested within the class that requires the listener
* Each listener class must implement an **interface**
* Interfaces for event listeners are provided in the package java.awt.event.\*
* ActionListener interface, which requires the method actionPerformed

import javax.swing.\*; // Needed for Swing classes

import java.awt.event.\*; // Needed for ActionListener public class KiloConverter extends JFrame

{

private JPanel panel; // To reference a panel

private JLabel messageLabel;

private JTextField kiloTextField;

private JButton calcButton;

private final int WINDOW\_WIDTH = 310;

private final int WINDOW\_HEIGHT = 100;

public KiloConverter()

{

setTitle("Kilometer Converter");

setSize(WINDOW\_WIDTH, WINDOW\_HEIGHT);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

buildPanel();

add(panel);

setVisible(true);

}

private void buildPanel()

{

messageLabel = new JLabel("Enter a distance " +

"in kilometers");

kiloTextField = new JTextField(10);

calcButton = new JButton("Calculate");

calcButton.addActionListener(new CalcButtonListener());

panel = new JPanel();

panel.add(messageLabel);

panel.add(kiloTextField);

panel.add(calcButton);

}

private class CalcButtonListener implements ActionListener

{

public void actionPerformed(ActionEvent e)

{

final double CONVERSION = 0.6214;

String input; // To hold the user's input

double miles; // The number of miles

input = kiloTextField.getText();

miles = Double.parseDouble(input) \* CONVERSION;

JOptionPane.showMessageDialog(null, input +

" kilometers is " + miles + " miles.");

}

}

public static void main(String[] args)

{

new KiloConverter();

}

}

* getContentPane().setBackground(Color.BLUE);

import javax.swing.\*; // Needed for Swing classes

import java.awt.\*; // Needed for Color class

import java.awt.event.\*; // Needed for event listener

public class ColorWindow extends JFrame

{

private JLabel messageLabel; // To display a message

private JButton redButton; // Changes color to red

private JButton blueButton; // Changes color to blue

private JButton yellowButton; // Changes color to

private JPanel panel; // A panel to hold

private final int WINDOW\_WIDTH = 200; // Window

private final int WINDOW\_HEIGHT = 125; // Window

public ColorWindow()

{

setTitle("Colors");

setSize(WINDOW\_WIDTH, WINDOW\_HEIGHT);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

messageLabel = new JLabel("Click a button to " +

"select a color.");

redButton = new JButton("Red");

blueButton = new JButton("Blue");

yellowButton = new JButton("Yellow");

redButton.addActionListener(new RedButtonListener());

blueButton.addActionListener(new BlueButtonListener());

yellowButton.addActionListener(new YellowButtonListener());

panel = new JPanel();

panel.add(messageLabel);

panel.add(redButton);

panel.add(blueButton);

panel.add(yellowButton);

add(panel);

setVisible(true);

}

private class RedButtonListener implements ActionListener

{

public void actionPerformed(ActionEvent e)

{

panel.setBackground(Color.RED);

messageLabel.setForeground(Color.BLUE);

}

}

private class BlueButtonListener implements ActionListener

{

public void actionPerformed(ActionEvent e)

{

panel.setBackground(Color.BLUE);

messageLabel.setForeground(Color.YELLOW);

}

}

private class YellowButtonListener implements ActionListener

{

public void actionPerformed(ActionEvent e)

{

panel.setBackground(Color.YELLOW);

messageLabel.setForeground(Color.BLACK);

}

}

public static void main(String[] args)

{

new ColorWindow();

}

}

* A **layout manager** is an object that allows a container to place GUI components in specific positions
* The BorderLayout manager divides a container into regions, where each region can contain only one GUI component; north, south, east, west, center
* The pack() method automatically sizes the window to hold the GUI components it contains
* The GridLayout manager is similar to a table or spreadsheet: it divides the container into rows and columns of cells
* isSelected() returns a boolean value indicating the current status of the button
* doClick() selects the button as if the user had clicked it
* JCheckBox objects also support isSelected() and doClick()
* Since clicking on a check box may deselect an already-selected check box, a program uses the isSelected() method to determine whether the click selected or deselected the check box

import javax.swing.\*;

import java.awt.\*;

public class ColorCheckBoxWindow extends JFrame

{

private JLabel messageLabel; // A message to the

private JCheckBox yellowCheckBox; // To select yellow

private JCheckBox redCheckBox; // To select red

private final int WINDOW\_WIDTH = 300; // Window

private final int WINDOW\_HEIGHT = 100; // Window

public ColorCheckBoxWindow()

{

setTitle("Color Check Boxes");

setSize(WINDOW\_WIDTH, WINDOW\_HEIGHT);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

messageLabel = new JLabel("Select the check " +

"boxes to change colors.");

yellowCheckBox = new JCheckBox("Yellow background");

redCheckBox = new JCheckBox("Red foreground");

yellowCheckBox.addItemListener(new CheckBoxListener());

redCheckBox.addItemListener(new CheckBoxListener());

setLayout(new FlowLayout());

pane.

add(messageLabel);

add(yellowCheckBox);

add(redCheckBox);

setVisible(true);

}

private class CheckBoxListener implements ItemListener

{

public void itemStateChanged(ItemEvent e)

{

if (e.getSource() == yellowCheckBox)

{

if (yellowCheckBox.isSelected())

{

getContentPane().setBackground(Color.yellow);

yellowCheckBox.setBackground(Color.yellow);

redCheckBox.setBackground(Color.yellow);

}

else

{

getContentPane().setBackground(Color.lightGray);

yellowCheckBox.setBackground(Color.lightGray);

redCheckBox.setBackground(Color.lightGray);

}

}

else if (e.getSource() == redCheckBox)

{

if (redCheckBox.isSelected())

{

messageLabel.setForeground(Color.red);

yellowCheckBox.setForeground(Color.red);

redCheckBox.setForeground(Color.red);

}

else

{

messageLabel.setForeground(Color.black);

yellowCheckBox.setForeground(Color.black);

redCheckBox.setForeground(Color.black);

}

}

}

}

public static void main(String[] args)

{

new ColorCheckBoxWindow();

}

}

* createEmptyBorder(top, bottom, left, right)
* createLineBorder(color, thickness)
* createTitledBorder(title)
* panel.setBorder(BorderFactory.createLineBorder(Color.RED, 1));

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.text.DecimalFormat;

public class BSindala\_Project3 extends JFrame

{

private JPanel destinationsPanel;

private JPanel optionsPanel;

private JPanel infoPanel;

private JPanel buttonPanel;

private JButton fareButton;

private JLabel fareLabel;

private JRadioButton atlanta;

private JRadioButton tennessee;

private JRadioButton miami;

private JRadioButton louisville;

private ButtonGroup bg;

private JCheckBox firstClass;

private JCheckBox roundTrip;

private JTextField bagsTextField;

private JLabel bagsLabel;

private JLabel infoLabel;

private final double ATLANTA = 205.87;

private final double TENNESSEE = 255.99;

private final double MIAMI = 301.45;

private final double LOUISVILLE = 365.00;

private final double FIRST\_CLASS = 200.00;

private final double BAG = 34.00;

private final double TAX\_RATE = 0.16;

public BSindala\_Project3()

{

setTitle("AirPath Flights");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(new BorderLayout());

infoPanel = new JPanel();

destinationsPanel = new JPanel();

optionsPanel = new JPanel();

buildPanel();

add(infoPanel, BorderLayout.NORTH);

add(destinationsPanel, BorderLayout.WEST);

add(optionsPanel, BorderLayout.EAST);

add(buttonPanel, BorderLayout.SOUTH);

pack();

setVisible(true);

}

public double getDestinationCost()

{

double destinationCost = 0.00;

if (atlanta.isSelected())

destinationCost = ATLANTA;

else if (tennessee.isSelected())

destinationCost = TENNESSEE;

else if (miami.isSelected())

destinationCost = MIAMI;

else if (louisville.isSelected())

destinationCost = LOUISVILLE;

return destinationCost;

}

the options to be added to the destination cost

public double getOptionsCost()

{

double optionsCost = 0.00, bags, bagsCost = 0.00;

String input;

input = bagsTextField.getText();

bags = Double.parseDouble(input);

bagsCost = BAG \* bags;

if (firstClass.isSelected())

optionsCost += FIRST\_CLASS;

if (bags > 0)

optionsCost += bagsCost;

else

{

JOptionPane.showMessageDialog(null, "Error! Number of Checked Bags cannot be Negative.");

System.exit(0);

}

return optionsCost;

}

private void buildPanel()

{

buttonPanel = new JPanel();

infoLabel = new JLabel("Please Enter the Info for your flight from BHM.");

infoPanel.add(infoLabel);

destinationsPanel.setLayout(new GridLayout(4, 1));

atlanta = new JRadioButton("Atlanta", true);

tennessee = new JRadioButton("Tennessee");

miami = new JRadioButton("Miami");

louisville = new JRadioButton("Louisville");

bg = new ButtonGroup();

bg.add(atlanta);

bg.add(tennessee);

bg.add(miami);

bg.add(louisville);

destinationsPanel.setBorder(BorderFactory.createTitledBorder("Destination"));

destinationsPanel.add(atlanta);

destinationsPanel.add(tennessee);

destinationsPanel.add(miami);

destinationsPanel.add(louisville);

optionsPanel.setLayout(new GridLayout(3, 1));

firstClass = new JCheckBox("First Class");

roundTrip = new JCheckBox("Round Trip");

bagsTextField = new JTextField(1);

bagsLabel = new JLabel("Checked Bags");

optionsPanel.setBorder(BorderFactory.createTitledBorder("Options"));

optionsPanel.add(firstClass);

optionsPanel.add(roundTrip);

optionsPanel.add(bagsTextField);

optionsPanel.add(bagsLabel);

fareButton = new JButton("Fare");

fareLabel = new JLabel();

fareButton.addActionListener(new FareButtonListener());

buttonPanel.add(fareLabel);

buttonPanel.add(fareButton);

}

private class FareButtonListener implements ActionListener

{

public void actionPerformed(ActionEvent e)

{

double subtotal, tax, total;

subtotal = getDestinationCost() +

getOptionsCost();

tax = subtotal \* TAX\_RATE;

total = subtotal + tax;

if(roundTrip.isSelected())

total = total \* 2;

DecimalFormat formatter = new DecimalFormat("0.00");

fareLabel.setText("Flight Fare: $" + formatter.format(total));

}

}

public static void main(String[] args)

{

new BSindala\_Project3();

}

}

* Java applets allow more application-like functionality on a Web page than HTML alone
* Applet classes extend JApplet instead of JFrame
* Classes that inherit from JApplet do not require a constructor: they contain a public void method named init with the necessary setup operations (excluding setTitle, setSize, setDefaultCloseOperation, pack, and setVisible)
* You do not need to instantiate the class (there is no main method)
* To run an applet embedded within a Web page without launching a browser, use the command-line tool appletviewer with the Web page’s file name:
* appletviewer FileName.html
* For the most part, AWT class names are the same as Swing class names without the “J”
* An additional import statement may be required: import java.applet.Applet;
* Standard GUI components (buttons, text fields, labels) are placed in an applet via the init method (or a helper method)
* Shapes are drawn in an applet using a separate method, paint
* Inherited from JApplet, JFrame, Applet, or Frame, Must be overridden (285), The parameter is an object of the Graphics class, The method calls the inherited version of the method via the super call (285) before actually invoking any of the graphics methods on the Graphics object
* If a program must trigger the paint method at another time (e.g., when an event occurs), it calls the void method repaint()
* drawLine(x1, y1, x2, y2)
* drawRect(x, y, width, height)
* drawOval(x, y, width, height)
* drawArc(x, y, width, height, startAngle, arcAngle)
* fillRect(x, y, width, height)
* fillOval(x, y, width, height)
* fillArc(x, y, width, height, startAngle, arcAngle)
* drawPolygon(xPoints, yPoints, numPoints) accepts two arrays holding the x and y coordinates of each point and the number of points stored in the arrays and draws a hollow polygon composed of the lines between the points (it automatically connects the last point to the first point)
* fillPolygon(xPoints, yPoints, numPoints)
* drawstring(str, x, y) draws the String argument str in a position with the bottom left corner of the text at (x, y)
* The paint method uses the entire applet area as the canvas

import javax.swing.\*;

import java.awt.\*;

public class DrawingPanel extends JPanel

{

private JCheckBox[] checkBoxArray;

public DrawingPanel(JCheckBox[] cbArray)

{

checkBoxArray = cbArray;

setBackground(Color.white);

setPreferredSize(new Dimension(300, 200));

}

public void paintComponent(Graphics g)

{

super.paintComponent(g);

if (checkBoxArray[0].isSelected())

{

g.setColor(Color.black);

g.drawLine(10, 10, 290, 190);

}

if (checkBoxArray[1].isSelected())

{

g.setColor(Color.black);

g.drawRect(20, 20, 50, 50);

}

if (checkBoxArray[2].isSelected())

{

g.setColor(Color.red);

g.fillRect(50, 30, 120, 120);

}

if (checkBoxArray[3].isSelected())

{

g.setColor(Color.black);

g.drawOval(40, 155, 75, 50);

}

if (checkBoxArray[4].isSelected())

{

g.setColor(Color.blue);

g.fillOval(200, 125, 75, 50);

}

if (checkBoxArray[5].isSelected())

{

g.setColor(Color.black);

g.drawArc(200, 40, 75, 50, 0, 90);

}

if (checkBoxArray[6].isSelected())

{

g.setColor(Color.green);

g.fillArc(100, 155, 75, 50, 0, 90);

}

}

}

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class GraphicsWindow extends JApplet

{

private JCheckBox[] checkBoxes;

private String[] titles = { "Line", "Rectangle",

"Filled Rectangle",

"Oval", "Filled Oval",

"Arc", "Filled Arc" };

private JPanel checkBoxPanel;

private DrawingPanel drawingPanel;

public void init()

{

buildCheckBoxPanel();

drawingPanel = new DrawingPanel(checkBoxes);

add(checkBoxPanel, BorderLayout.EAST);

add(drawingPanel, BorderLayout.CENTER);

}

private void buildCheckBoxPanel()

{

checkBoxPanel = new JPanel();

checkBoxPanel.setLayout(new GridLayout(7, 1));

checkBoxes = new JCheckBox[7];

for (int i = 0; i < checkBoxes.length; i++)

{

checkBoxes[i] = new JCheckBox(titles[i]);

checkBoxes[i].addItemListener(

new CheckBoxListener());

checkBoxPanel.add(checkBoxes[i]);

}

}

private class CheckBoxListener implements ItemListener

{

public void itemStateChanged(ItemEvent e)

{

drawingPanel.repaint();

}

}

}

* The Java class Applet has a method named play that supports playing an audio file one time
* The method getDocumentBase() returns a URL object with the location of the HTML file embedding the applet
* The method getCodeBase() returns a URL object with the location of the class file
* A more flexible approach from the java.applet package is to use the class AudioClip
* Instead of constructing an AudioClip object, assign a call to getAudioClip (with the same arguments as to play above) to the AudioClip variable
* AudioClip objects support the methods play(), loop(), and stop() (no arguments needed)
* Since an application does not extend JApplet, the syntax is a little different
* File file = new File("step.wav");
* URI uri = file.toURI();
* URL url = uri.toURL();
* sound = Applet.newAudioClip(url);