Acronyms

- GUI: Graphical User Interface.
- API: Application Program Interface.

Java GUI API Toolkits

- AWT:
  - Abstract Windows Toolkit.
  - Used in early versions of Java.
- Swing:
  - Replaces AWT.
  - Built on top of AWT.
HelloWorldSwing

• Simplest possible Swing Program.
• Create a frame to hold the window.
• Create a label to display in the window.
• Add the label to the frame’s content pane.
• Arrange for the application to exit when window is closed.
• Size the frame to hold the label.
• Make the frame visible.

package helloworldswing;
import javax.swing.*;

public class HelloWorldSwing {
    public static void main(String[] args) {
        JFrame frame = new JFrame("HelloWorldSwing");
        final JLabel label = new JLabel("Hello World");
        frame.getContentPane().add(label);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);
    }
}

SwingApplication

• Behavior:
  – Display a window with a button and a label.
  – When the user clicks the button, the label is updated.

• Implementation:
  – Creates an ActionListener object.
  – Registers the ActionListener to be notified of button click events.
  – Listener’s actionPerformed method processes click events.
package swingapplication;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class SwingApplication {
    private static String labelPrefix = "Number of button clicks: ";
    private int numClicks = 0;
    // ... Omitted ...
    public static void main(String[] args) {
        JFrame frame = new JFrame("SwingApplication");
        SwingApplication app = new SwingApplication();
        Component contents = app.createComponent();
        frame.getContentPane().add(contents, BorderLayout.CENTER);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);
    }

    public Component createComponents() {
        final JLabel label = new JLabel(labelPrefix + "0");
        JButton button = new JButton("I'm a Swing button!");
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                numClicks++;
                label.setText(labelPrefix + numClicks);
            }
        });
        label.setLabelFor(button);
        JPanel pane = new JPanel();
        pane.setBorder(BorderFactory.createEmptyBorder(30,30,10,30));
        pane.setLayout(new GridLayout(2,1));
        pane.add(button);
        pane.add(label);
        return pane;
    }
}

---

Event Driven Programming

- An application's main method:
  - Constructs the GUI.
  - Registers event listeners on GUI components.
  - Returns.
- Java Virtual Machine:
  - Notices whenever a GUI event has occurred.
  - Runs the appropriate event listeners to process the event.
Event Driven Interaction

- Changes in mouse or keyboard state are called “events”.
- Each event is associated with a GUI object.
- Device Manager Thread stores (event, object) pairs in a queue.
- Event Processing Thread repeatedly:
  - Takes an (event, object) pair out of the queue.
  - Finds all the event listeners registered on the object.
  - Invokes an event processing method on each listener.
  - Sends the event as its parameter.

Event Driven Architecture

Two Ways of Defining Event Listeners

1. Define a class that implements the appropriate event listener interface.
   E.g., A class called “MyButtonListener” that implements the ActionListener interface.
2. Define an anonymous inner class:
   - Class defined within the body of a method.
   - Definition of methods only, no class name.
   - E.g., Defining only the actionPerformed method.
   - Allows listener definitions to be located near registration code.
Registering Event Listeners

- Identify the GUI object that will be associated with the listener.
  - E.g., A Jbutton object called “button”.
- Create a new object of the appropriate event listener type.
  - E.g., A new ActionListener().
- Invoke the GUI object’s add<ListenerType> method with the listener object as parameter.

Event Listener Types

- ActionListener: Component-defined events.
- KeyListener: Keyboard events.
- MouseListener: Mouse click events.
- MouseMotionListener: Mouse motion events.
- ItemListener: Menu selection/deselection events.
- WindowListener: Window open/close etc. events.
- …Many More…

Event Processing Thread

- Event listeners run in the event processing thread.
- While they run, they hold a lock on the user interface.
- During this time, the GUI will not respond to user initiated events.
- Event listeners performing time consuming tasks should create their own threads to run those tasks.
Some Swing Components

• JFrame: Object implementing window with decorations such as border, title, and buttons for closing and iconifying.
• JPanel: Object for organizing and positioning other components.
• JLabel: Object for displaying short text strings or images or both.
• JButton: Object that responds to mouse clicks.
• JTextField: Object in which user can type text.
• JComboBox: Object that combines button or text field with drop down list.

Layout Managers

• A JPanel object has a setLayout method that takes a LayoutManager object as its parameter.
• The LayoutManager object controls the arrangement of components inside the JPanel object.
• Examples of LayoutManagers: BorderLayout, GridLayout, FlowLayout, GridBagLayout, BoxLayout, etc.

CelsiusConverter.java

• User enters Celsius temperature in a JTextField.
  – Program uses getText() method of JTextField.
• Program converts temperature to Fahrenheit and displays it in a JLabel.
  – Program uses setText() method of JLabel.
• Reading text, parsing text and converting temperature are triggered by a button click.
package celsiusconverter;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class CelsiusConverter implements ActionListener {
    JFrame converterFrame;
    JPanel converterPanel;
    JTextField tempCelsius;
    JLabel celsiusLabel, fahrenheitLabel;
    JButton convertTemp;
    static final int WIDTH = 400;
    static final int HEIGHT = 100;

    // ... Omitted ...

    public static void main(String[] args) {
        CelsiusConverter converter = new CelsiusConverter();
    }

    public CelsiusConverter() {
        converterFrame = new JFrame("Convert Celsius to Fahrenheit");
        converterFrame.setSize(WIDTH,HEIGHT);
        converterPanel = new JPanel();
        converterPanel.setLayout(new GridLayout(2, 2));
        addWidgets();
        converterPanel.add(tempCelsius);
        converterPanel.add(celsiusLabel);
        converterPanel.add(convertTemp);
        converterPanel.add(fahrenheitLabel);
        celsiusLabel.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        fahrenheitLabel.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        converterFrame.getContentPane().add(converterPanel, BorderLayout.CENTER);
        converterFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        converterFrame.setVisible(true);
    }

    private void addWidgets() {
        tempCelsius = new JTextField();
        celsiusLabel = new JLabel("Celsius", SwingConstants.LEFT);
        convertTemp = new JButton("Convert...");
        fahrenheitLabel = new JLabel("Fahrenheit", SwingConstants.LEFT);
        convertTemp.addActionListener(this);
        converterPanel.add(tempCelsius);
        converterPanel.add(celsiusLabel);
        converterPanel.add(convertTemp);
        converterPanel.add(fahrenheitLabel);
        celsiusLabel.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        fahrenheitLabel.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
    }
}
public void actionPerformed(ActionEvent event) {
    double tc = Double.parseDouble(tempCelsius.getText());
    int tf = (int) (tc * 1.8 + 32);
    fahrenheitLabel.setText(tf + " Fahrenheit");
}

• User selects a phase of the moon from a dropdown menu.
  – JComboBox implements the dropdown menu.
  – JComboBox’s getSelectedIndex method determines the
    user’s selection.
• Program displays an image of the moon in the
  selected phase.
  – Images are read from files into URL objects.
  – URL objects are converted into ImageIcon objects.
  – JLabel’s setIcon method displays the image.

package lunarphases;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.net.URL;
public class LunarPhases implements ActionListener {
    final static int NUM_IMAGES = 8;
    final static int START_INDEX = 3;
    ImageIcon[] images = new ImageIcon[NUM_IMAGES];

    JPanel mainPanel, selectPanel, displayPanel;
    JComboBox phaseChoices = null;
    JLabel phaseIconLabel = null;
    // ... Omitted ...
    public static void main(String[] args) {
        LunarPhases phases = new LunarPhases();
        JFrame lunarPhasesFrame = new JFrame("Lunar Phases");
        lunarPhasesFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        lunarPhasesFrame.pack();
        lunarPhasesFrame.setVisible(true);
    }
}
public LunarPhases() {
    selectPanel = new JPanel();
    displayPanel = new JPanel();
    initSelectPanel();
    initDisplayPanel();
    mainPanel = new JPanel();
    mainPanel.setLayout(new GridLayout(2, 1, 5, 5));
    mainPanel.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
    mainPanel.add(selectPanel);
    mainPanel.add(displayPanel);
}

private void initSelectPanel() {
    String[] phases = {"New", "Waxing Crescent", "First Quarter", "Waxing Gibbous", "Full", "Waning Gibbous", "Third Quarter", "Waning Crescent"};
    phaseChoices = new JComboBox(phases);
    phaseChoices.setSelectedIndex(START_INDEX);
    selectPanel.setBorder(BorderFactory.createCompoundBorder(  
        BorderFactory.createTitledBorder("Select Phase"),  
        BorderFactory.createEmptyBorder(5, 5, 5, 5)));
    displayPanel.setBorder(BorderFactory.createCompoundBorder(  
        BorderFactory.createTitledBorder("Display Phase"),  
        BorderFactory.createEmptyBorder(5, 5, 5, 5)));
    phaseChoices.addActionListener(this);
    selectPanel.add(phaseChoices);
}

private void initDisplayPanel() {
    for (int i = 0; i < NUM_IMAGES; i++) {
        String imageName = "images/image" + i + ".jpg";
        URL iconURL = ClassLoader.getSystemResource(imageName);
        images[i] = new ImageIcon(iconURL);
        phaseIconLabel = new JLabel();
        phaseIconLabel.setHorizontalAlignment(JLabel.CENTER);
        phaseIconLabel.setVerticalAlignment(JLabel.CENTER);
        phaseIconLabel.setVerticalTextPosition(JLabel.CENTER);
        phaseIconLabel.setHorizontalTextPosition(JLabel.CENTER);
        phaseIconLabel.setBorder(BorderFactory.createCompoundBorder(  
            BorderFactory.createTitledBorder("Display Phase"),  
            BorderFactory.createEmptyBorder(5, 5, 5, 5)));
        displayPanel.add(phaseIconLabel);
    }
}
public void actionPerformed(ActionEvent event) {
    if ("comboBoxChanged".equals(event.getActionCommand())) {
        phaseIconLabel.setIcon(images[phaseChoices.getSelectedIndex()]);
    }
}

VoteDialog.java

- User selects one candidate from a list.
  - ButtonGroup organizes JRadioButton objects.
  - Selecting one button in group deselects any other selection.
  - Each JRadioButton has an associated action command string.
- Vote button initiates a dialogue:
  - ButtonGroup’s getSelection method gets the currently selected button of the group.
  - JRadioButton’s getActionCommand method gets the button’s action command string.
  - Action command string determines the subsequent dialog.

Dialog Boxes

- Class JOptionPane implements simple dialog methods:
  - ShowMessageDialog.
  - ShowConfirmDialog.
  - ShowOptionDialog.
- Dialogs are modal, i.e., while dialog is visible, it blocks all other input to the program.
Java Applets

- Java programs designed to run in web browsers.
- Implemented as classes that extend the `Applet` or `JApplet` abstract classes.
  - AWT Applets: Derived from `Applet` class.
  - Swing Applets: Derived from `JApplet` class.
- These abstract classes specify the methods that Java applets must implement.

HelloInitApplet

- Uses the `init()` method to do all its work:
  - Creates a label with a message.
  - Sets its location and give it a border.
  - Adds it to the content pane.
- The content pane will automatically repaint itself, and the label, when necessary.

```java
package helloinitapplet;
import java.awt.*;  
import javax.swing.*;
public class HelloInitApplet extends JApplet
{
    public void init()
    {
        JLabel label = new JLabel("Hello World ... I'm Initializing!");
        label.setHorizontalAlignment(JLabel.CENTER);
        label.setBorder(BorderFactory.createMatteBorder(1,1,2,2,Color.black));
        getContentPane().add(label, BorderLayout.CENTER);
    }
}
```
HelloPaintApplet

https://vspace.vassar.edu/thellman/web/102/applets/HelloPaintApplet.html

• Uses the `paint(Graphics g)` method to do all its work:
  – Draws a string with a message.
  – Uses the `drawString` method of the `Graphics` class.
• The browser calls the applet’s paint method whenever the applet’s display area needs to be updated.

```java
package hellopaintapplet;
import java.awt.*;
import javax.swing.*;
public class HelloPaintApplet extends JApplet {
    public void paint(Graphics g) {
        g.drawString("Hello world ... I'm Painting!", 50, 20);
    }
}
```

Methods Implementing the Applet Life Cycle

• **void init()**: Called once, before the applet starts running. Used to load resources.
• **void start()**: Called after initialization, and again whenever the browser returns to the page holding the applet.
• **void paint()**: Called whenever the applet’s display area needs to be updated.
• **void stop()**: Called whenever the browser leaves the page holding the applet, or when the browser exits.
Running an Applet in a Browser

- Embed the applet in an HTML document.

- Use the `<APPLET>` tag:
  - Use the `<CODE>` field to specify the location of class files.
  - Use the `<WIDTH>` and `<HEIGHT>` fields to specify the applet’s area on the web page.

HelloInitApplet.html

```html
<html>
<head>
<title>HelloInitApplet</title>
</head>
<body>
<H1>HelloInitApplet</H1>

<applet
  CODE = "helloinitapplet/HelloInitApplet.class"
  WIDTH = 400
  HEIGHT = 50
>
</applet>
</body>
</html>
```

SimpleScrollingApplet

https://vspace.vassar.edu/thellman/web/102/applets/SimpleScrollingApplet.html

- Uses the `<init>` method to do all its work:
  - Creates a text field and puts some text in it.
  - Adds the text field the content pane, using a layout manager.
  - User may type text into the field and scroll back and forth to view his typed text.
- The content pane will automatically repaint itself, and the text field, when necessary.
package simplescrollingapplet;
import java.awt.*;
import javax.swing.*;
public class SimpleScrollingApplet extends JApplet {
    JTextField field;
    public void init() {
        field = new JTextField();
        field.setText("Type and Scroll Here!");
        getContentPane().setLayout(new GridLayout(1,0));
        getContentPane().add(field);
    }
}

SimpleClickingApplet
https://vspace.vassar.edu/thellman/web/102/applets/SimpleClickingApplet.html

- Implements the MouseListener interface.
- Registers itself as a mouse listener attached to its display component.
- MouseClicked method updates a StringBuffer and calls repaint which calls paint.
- The paint method draws the updated string in the applet’s display area.

package simpleclickingapplet;
import java.awt.*;
import javax.swing.*;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
public class SimpleClickingApplet extends JApplet implements MouseListener {
    StringBuffer buffer;
    public void init() {
        // ... omitted ...
    }
    public void paint(Graphics g) {
        // ... omitted ...
    }
    // ... omitted ...
}
Threads and Applets

- The standard methods of the applet classes may be running in an event-processing thread.
- If these methods take a long time, they will may freeze up the interface.
- Applets should therefore create their own threads to carry out time-consuming operations.

The Runnable Interface

- An applet cannot itself be a `Thread` object.
- Nevertheless, an applet can create a `Thread` object to do its work:
  - Declare the applet to implement the `Runnable` interface.
  - Provide the applet with a `run` method.
  - In applet’s `start` method, create and start a `Thread` with this applet as its “target”.
Targets of Threads

- The “target” of a thread is the object on which the thread’s `start` method initiates the `run` method.
- By default, the target of a thread is normally the thread itself.
- The target may also be a different object.
- E.g., the applet that creates the thread.

Clock

- Implements the `Runnable` interface.
- The applet’s `start` method sets up a `Thread` with this applet as its target.
- The thread’s `start` method initiates the applet’s `run` method.
- The applet’s `run` method calls `repaint` and `paint` once each second.
- The applet’s `paint` method displays the current time in the applet’s display area.
- The applet’s `stop` method sets a variable that causes the `run` method to exit, terminating the thread.

```java
package clockapplet;
import java.awt.Graphics;
import java.awt.Color;
import java.util.*;
import java.text.DateFormat;
import javax.swing.*;
public class ClockApplet extends JApplet implements Runnable {
    private Thread clockThread = null;
    public void init() {
        getContentPane().setBackground(Color.white);
    }
    public void start() {
        if (clockThread == null) {
            clockThread = new Thread(this, "Clock");
            clockThread.start();
        }
    }
    public void stop() {
        clockThread = null;
    }
    // ... Omitted ...
    public void start() {
        if (clockThread == null) {
            clockThread = new Thread(this, "Clock");
            clockThread.start();
        }
    }
    // ... Omitted ...
    public void stop() {
        clockThread = null;
    }
}
```
public void run() {
    Thread myThread = Thread.currentThread();
    while (clockThread == myThread) {
        repaint();
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) { }
    }
}

public void paint(Graphics g) {
    Calendar cal = Calendar.getInstance();
    Date date = cal.getTime();
    DateFormat dateFormatter = DateFormat.getTimeInstance();
    getContentPane().paint(g);
    g.drawString(dateFormatter.format(date), 5, 10);
}

Accessing Resources from Applets

• The Java class files implementing the applet are normally located in the same directory as the web page HTML document.
• The HTML document may specify a CODEBASE parameter giving the URL of the directory or folder that contains the apple’s code.

Passing Parameters to Applets

• How can a single Java applet be customized differently in different web pages?
• Notice that the applet’s methods take application-independent parameters.
• Solution is to specify parameters in the HTML document.
• The applet uses the getParameter(String) method to access its parameters.
AnimationApplet
https://vspace.vassar.edu/thellman/web/102/applets/AnimationApplet.html

- Applet uses a Timer object to implement the thread that does the animation.
- Applet creates the Timer object passing itself as the timer's target.
- The Timer periodically calls the applet's actionPerformed method.
- The applet’s actionPerformed method updates the animation.

SoundApplet
https://vspace.vassar.edu/thellman/web/102/applets/SoundApplet.html

- Applet creates four buttons to control playing of sounds.
- Applet adds itself as an ActionListener associated with each button.
- When a button is pressed, the applet’s actionPerformed method is called to carry out the requested operation.
ACMProgramApplet

Applet runs a program that uses the ACM Student Package.

The applet’s init method:
- Creates an instance of the ACM program object.
- Calls start on the instance.

```java
package acmprogramapplet;
import javax.swing.*;

public class ACMProgramApplet extends JApplet {
    public void init() {
        new EasyInteraction().start();
    }
}

package acmprogramapplet;
import acm.program.*;

public class EasyInteraction extends ConsoleProgram {
    public void run() {
        println("Welcome to CMPU102: Topics in Computer Science");
        String firstName = readLine("Please enter your first name: ");
        String lastName = readLine("Please enter your last name: ");
        println("Hello, " + firstName + " " + lastName + ".");
        println("It's a pleasure to have you with us!");
    }
}
```
Making the ACM Student Package Available to the Browser

- Use the ARCHIVE field in the APPLET entry on the HTML file to specify the .jar file containing the ACM Student package classes.
- Place the acm.jar file on the web site at the location specified in the HTML file.

```html
<html>
<head>
<title>ACMProgramApplet</title>
</head>
<body>
<H1>ACMProgramApplet</H1>
<applet archive="acm.jar"
        code="acmprogramapplet/ACMProgramApplet.class"
        width=600 height=400>
</applet>
</body>
</html>
```