Building Components using Swing and JavaFX

Prativa Nyaupane

Recap

- Final classes and methods
- Abstract classes and methods
- Upcasting and Down casting
- Interfaces and Implementation

Today's Objective

1. Introduction to AWT and Swing:
    a. Concept
    b. Applets
    c. Swing Class Hierarcy
    d. Components/Containers
2. Layout Management

Introduction to AWT

- The most important feature in Java is its ability to draw graphics.
- AWT(Abstract Window Toolkit) provides the foundation for creating GUI components in Java. It was introduced in the mid-1990s with the first version of Java.
- All the classes are contained in java.awt package.
- These classes are hierarchically arranged inside the AWT package that each successive level in the hierarchy adds certain attributes to the GUI application.
- AWT provides support for both standard and applet windows.

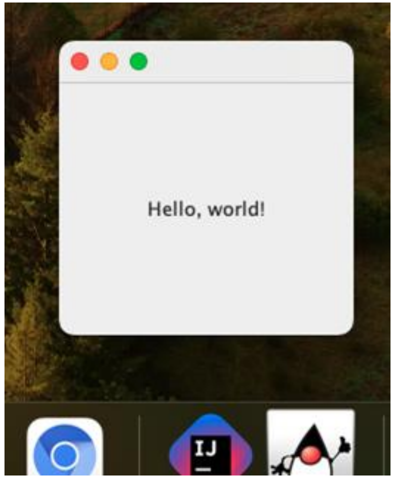
AWT Contd...

- AWT is built upon the native windowing system of the underlying operating system. This means that the appearance and behavior of AWT components are determined by the host operating system's graphical environment.
- AWT components are heavyweight components, meaning that they directly map to and utilize the native windowing system's resources.
- AWT provides a basic set of GUI components, including buttons, text fields, checkboxes and menus.

Initializing Frame by inheritance:

```
import java.awt.*;

public class FirstAWTClass extends Frame {
   public FirstAWTClass(){
       Label label = new Label("Hello, world!");//Label is a component
       label.setAlignment(Label.CENTER);//Aligned the text to center
       add(label);//instruct the container to include the specified component ie Frame
       setSize(200,200);
       setVisible(true);//shows the container
   }
}
public class Main {
   public static void main(String[] args) {
         new FirstAWTClass();
   }
}
```

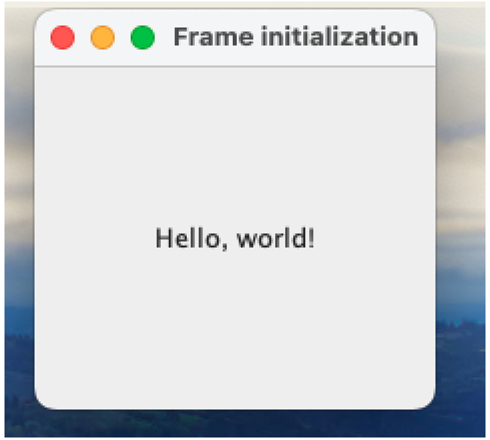
Output:



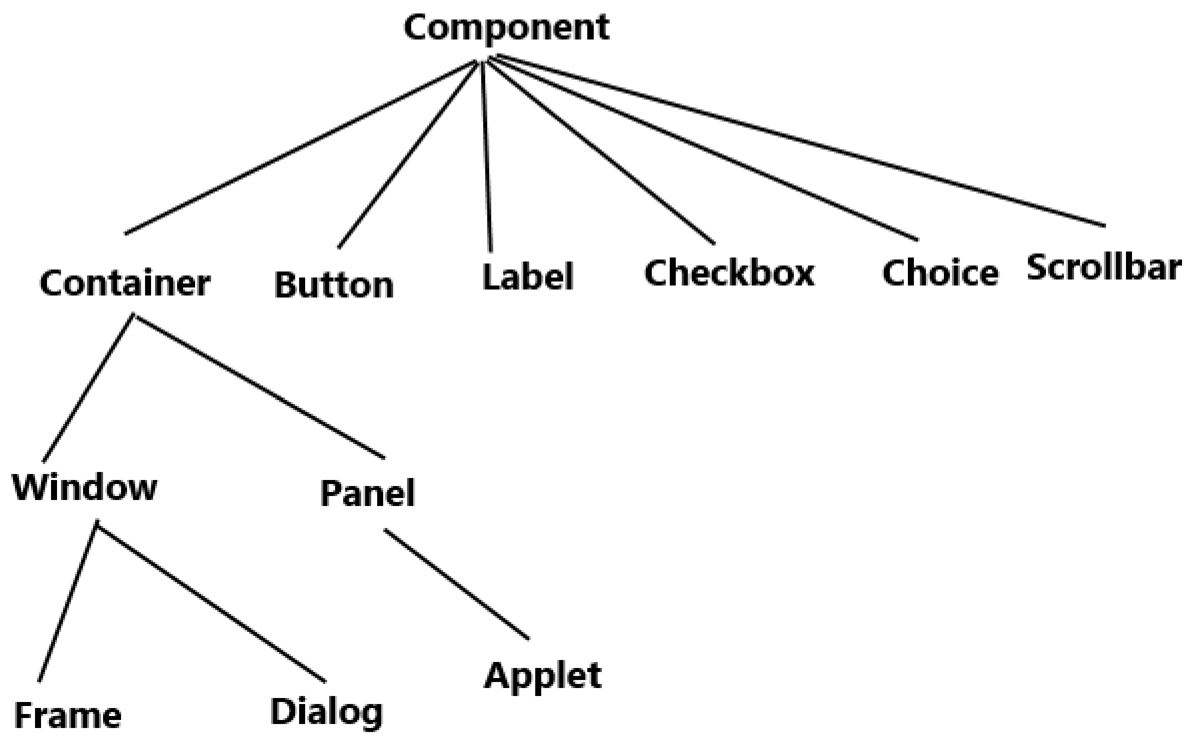
Initializing Frame by Association(instance creation):

```
import java.awt.*;

public class SecondAWTClass {
    public SecondAWTClass() {
        Frame frame = new Frame("Frame initialization");
        Label label = new Label("Hello, world!");
        label.setAlignment(Label.CENTER);
        frame.add(label);
        frame.setSize(200, 200);
        frame.setVisible(true);

    }

}
public class Main {
    public static void main(String[] args) {
        new SecondAWTClass();
    }
}
```

Output:



AWT Hierarchy



AWT Hierarchy Contd…

Components: All the elements like buttons, text fields are known as components. Component class is the super class to all the other class from which various GUI elements are realized. It is responsible for affecting the display of a graphic object on the screen.

Container: A container is like a screen wherein we are placing components like buttons, text fields, checkbox, ect. A container contains and controls the layout of the components.

Window: Frame and Dialog are subclass of Window class. The Window is an instance of Window class. Typically used Window is Frame.

AWT Hierarchy Contd...

Panel: The superclass of applet, Panel, represents a window space on which the application’s output is displayed. Panel does not contain title bar, menu bar or border. It is a generic container for holding components.

Frame: A frame has title, border and menu bars. It can contain several components like buttons, text fields, scrollbars.

Applet

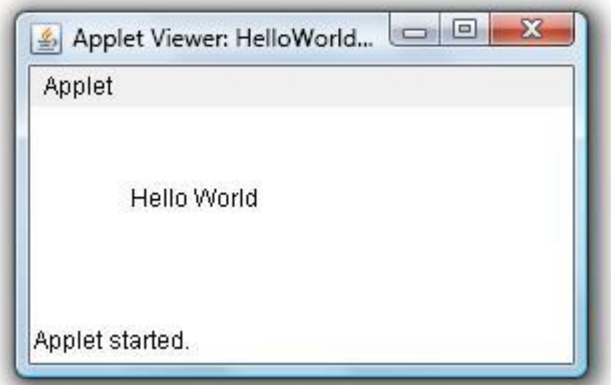
- In Java, an applet is a small Java program that runs within a web browser, typically to provide interactive features to web application.
- Applets are embedded in webpages.
- Applets use AWT or Swing libraries to create GUIs, allowing developers to create interactive components.
- Overtime, due to security vulnerabilities and the evolution of web technologies, Java applets are less common. As of Java 9, the browser plugin that supported Java applets was deprecated, and as of Java 11, it was removed.

```
import java.applet.Applet;
import java.awt.*;

public class FirstAppletClass extends Applet {
   @Override
   public void paint(Graphics g){
       g.drawString("Hello World", 20, 20);
   }
}
```

To run java applets, we use the following command line:

- javac FirstAppletClass.java
- appletviewer FirstAppletClass.java

Output:



```
Hello World
```

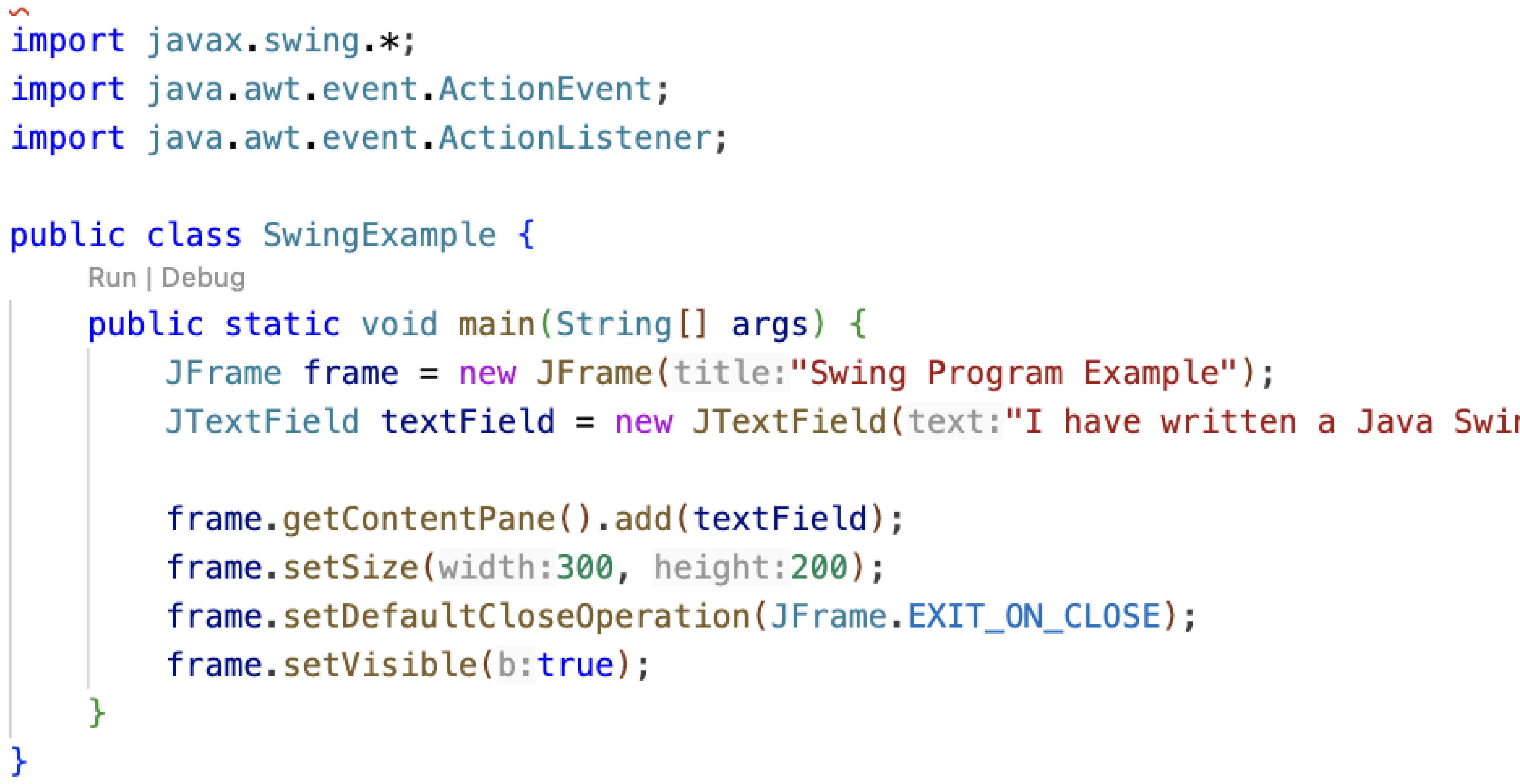
Applet started.

Swing

- Swing is a more advanced GUI framework built on top of AWT, introduced with the release of Java 2(JDK 1.2) in 1998.
- It is often referred to as the “Swing toolkit” or “Swing library.”
- Swing components are written entirely in Java and do not rely on the native windowing system. This allows Swing to have a consistent look and feel across different operating systems.
- Swing components are lightweight components, meaning they are not bound to the native windowing system and do not utilize its resources directly. Instead, they are rendered using Java code and can be customized extensively.
- Swing provides a rich set of GUI components, including advanced controls like tables, trees, tabbed panes and more.

AWT/SWING

- In summary, AWT is the foundation of Java’s GUI capabilities and provides a basic set of GUI components that utilize the native windowing system.
- Swing, on the other hand, is a more advanced and flexible GUI framework built on top of AWT, offering lightweight components with consistent appearance and behavior across different platforms.
- Swing components are written in Java and offer greater customization options compared to AWT.



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

public class SwingExample {
    Run | Debug
    public static void main(String[] args) {
        JFrame frame = new JFrame(title:"Swing Program Example");
        JTextField textField = new JTextField(text:"I have written a Java Swi

        frame.getContentPane().add(textField);
        frame.setSize(width:300, height:200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(b:true);
    }
}
```

Layout Management

- Layout managers control the arrangement and sizing of components within a container.
- Different layout managers provide different strategies for organizing components.
    - FlowLayout: Arranges components in a left-to-right flow.
    - BorderLayout: Divides the container into five regions(North, South, East, West, Center)
    - GridLayout: Arranges components in a grid of rows and columns.
    - BoxLayout: Arranges components in in a single row or colum.
    - CardLayout: Manages multiple components, showing only one at a time like a deck of cards.

Layout Management Contd...

- Layout managers are set using the setLayout method on a container.

```
    JFrame frame = new JFrame("BorderLayout Example");
    frame.setLayout(new BorderLayout());
```

- The choice of layout manager depends on the desired GUI structure and behavior.
- Layout managers in Java are used to handle the arrangement and sizing of components within a container.
- They play a crucial role in GUI programming by providing a flexible and dynamic way to organize graphical elements within a window.