Creating Graphical User Interfaces with JAVA

Dr. Krishnendu Guha

Assistant Professor/ Lecturer

School of Computer Science and Information Technology

University College Cork

Introduction

- We need to ***reuse*** **the graphics classes** provided in JDK for constructing our own Graphical User Interface (GUI) applications
- Writing own graphics is not impossible
- The existing graphics classes that are developed by expert programmers, are highly complex and involve many advanced *design patterns*.
- However, re-using them are not so difficult, if we follow the API documentation, samples and templates provided.

- There are current three sets of Java APIs for graphics programming:
  - AWT (Abstract Windowing Toolkit),
  - Swing and
  - JavaFX.

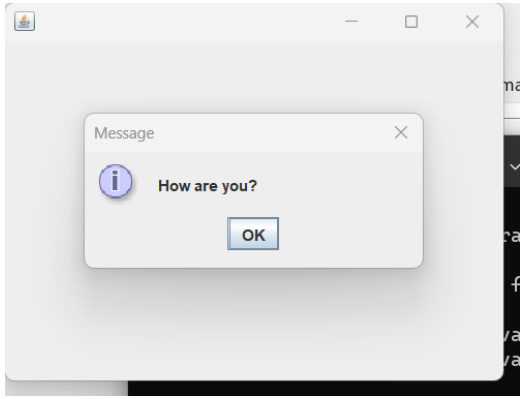
1. AWT API was introduced in JDK 1.0. Most of the AWT UI components have become obsolete and are replaced by newer Swing UI components.
2. Swing API, a much more comprehensive set of graphics libraries that enhances the AWT, was introduced as part of Java Foundation Classes (JFC) after the release of JDK 1.1. JFC consists of Swing, Java2D, Accessibility, Internationalization, and Pluggable Look-and-Feel Support APIs.
3. The latest JavaFX, which was integrated into JDK 8, was meant to replace Swing. JavaFX was moved out from the JDK in JDK 11, but still available as a separate module.

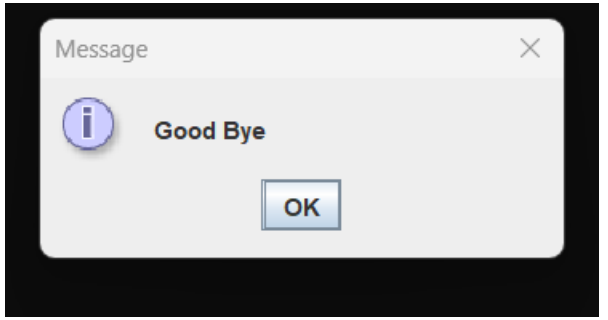
- Other than AWT/Swing/JavaFX graphics APIs provided in JDK, other organizations/vendors have also provided graphics APIs that work with Java, such as Eclipse's Standard Widget Toolkit (SWT) (used in Eclipse), Google Web Toolkit (GWT) (used in Android), 3D Graphics API such as Java bindings for OpenGL (JOGL), Java3D, and etc.

Simple GUI I/O with JOption Pane

- One of the easiest ways to provide a simple GUI-based input and output is by using the JOptionPane class.
- For example, when we execute the statement
  - JOptionPane.showMessageDialog**(null**, “This is UCC"**)**;

```
import javax.swing.*;
class GUI {
public static void main(String[] args) {
        JFrame jFrame;
        jFrame = new JFrame();
        jFrame.setSize(400,300);
        jFrame.setVisible(true);
        JOptionPane.showMessageDialog(jFrame, "How are you?");
        JOptionPane.showMessageDialog(null, "Good Bye");
        }
}
```

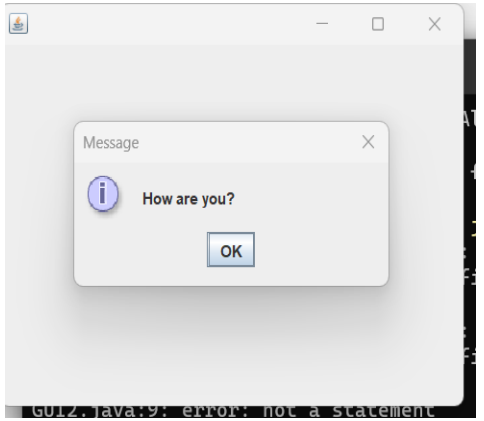




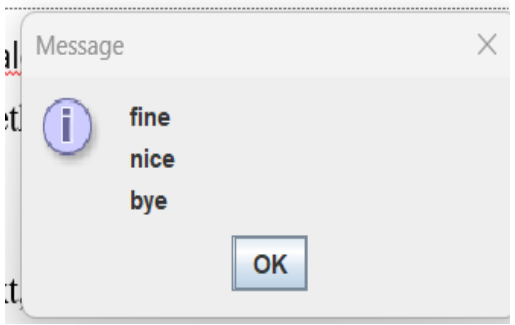
- We are not creating an instance of the JDialog class directly by ourselves.
- When we call the showMessageDialog method, the JOptionPane class is actually creating an instance of JDialog internally.

- If we want to display multiple lines of text, we can use a special character sequence \n to separate the lines, as follows:

```
JOptionPane.showMessageDialog(null, “fine\nnice\nbye");
```




```
import javax.swing.*;
class GUI2 {
public static void main(String[] args) {
        JFrame jFrame;
        jFrame = new JFrame();
        jFrame.setSize(400,300);
        jFrame.setVisible(true);
        JOptionPane.showMessageDialog(jFrame, "How are you?");
        JOptionPane.showMessageDialog(null, “fine\nnice\nbye");
        }
}
```

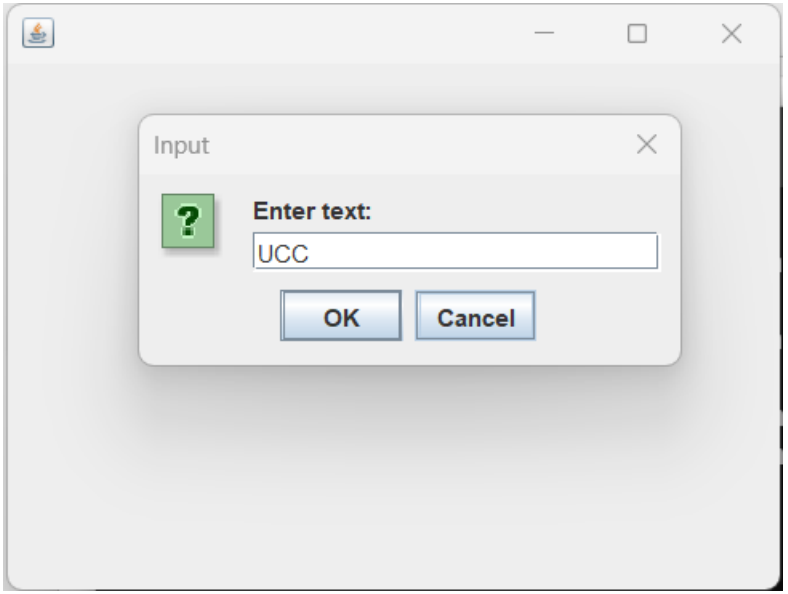


- We can also use the JOptionPane class for input by using its showInputDialog method.
- To assign the name input to an input string, we write

  String input;

  input = JOptionPane.showInputDialog**(null**, "Enter text:"**)**;
- Unlike the Scanner class that supports different input methods for specific data types, that is, nextInt and nextDouble, the JOptionPane supports only a string input.
- NOTE: To input a numerical value, we need to perform the string conversion ourselves.

```
import javax.swing.*;
class GUI3 {
public static void main(String[] args) {
        JFrame jFrame;
        jFrame = new JFrame();
        jFrame.setSize(400,300);
        jFrame.setVisible(true);
        String input;
        input = JOptionPane.showInputDialog(null, "Enter text:");
        }
}
```



- **Simple subclass of Jframe that changes the background color to blue**

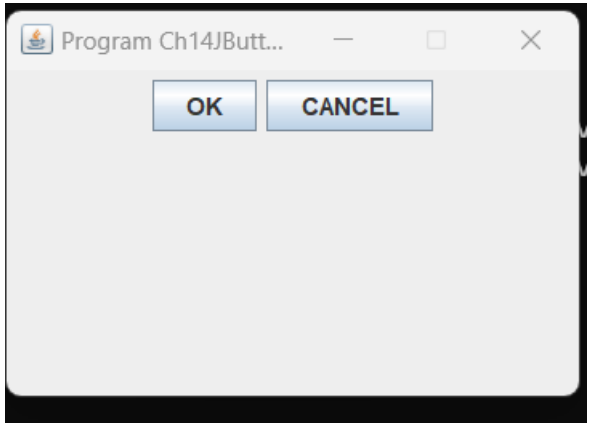
```
import javax.swing.*;
import java.awt.*;
class Background extends JFrame {
        private static final int FRAME_WIDTH = 300;
        private static final int FRAME_HEIGHT = 200;
        private static final int FRAME_X_ORIGIN = 150;
        private static final int FRAME_Y_ORIGIN = 250;
        public static void main(String[] args) {
        Background frame = new Background();
        frame.setVisible(true);
}

public Background() {
        //set the frame default properties
        setTitle ("Blue Background JFrame Subclass");
        setSize (FRAME_WIDTH, FRAME_HEIGHT);
        setLocation (FRAME_X_ORIGIN, FRAME_Y_ORIGIN);
        //register 'Exit upon closing' as a default close operation
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        changeBkColor();
        }
private void changeBkColor() {
        Container contentPane = getContentPane();
        contentPane.setBackground(Color.BLUE);
        }
}
```

Button Placement

- The type of button we use here is called a ***pushbutton***
- To use a button in a program, we create an instance of the javax.swing.JButton class.
- We will create two buttons and place them on the frame's content pane in the constructor.
- Let's name the two buttons cancelButton and okButton.
- We declare and create these buttons in the following manner:

```
import javax.swing.*;
...
JButton cancelButton, okButton;
cancelButton = new JButton("CANCEL");
okButton = new JButton("OK");
```

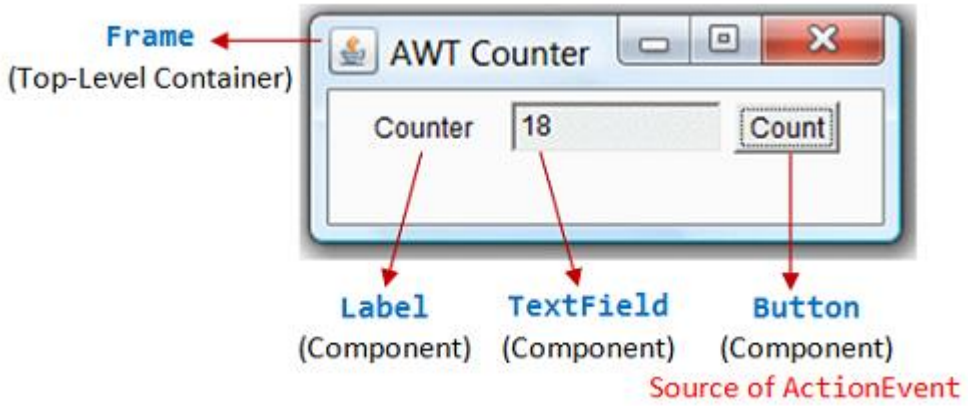


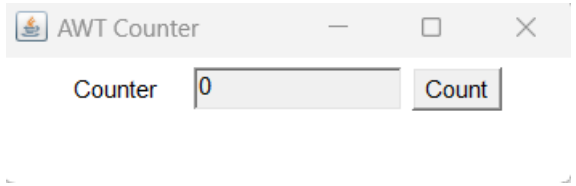
Implementing a Counter with AWT

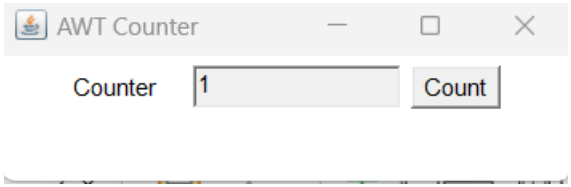
The program has a top-level container Frame, which contains three components

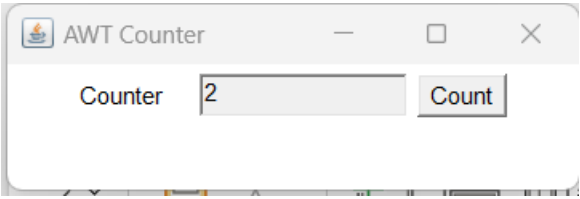
- a Label "Counter",
- a non-editable TextField to display the current count, and
- a "Count" Button.

- The TextField shall display count of 0 initially.
- Each time the button is clicked, the counter's value increases by 1.



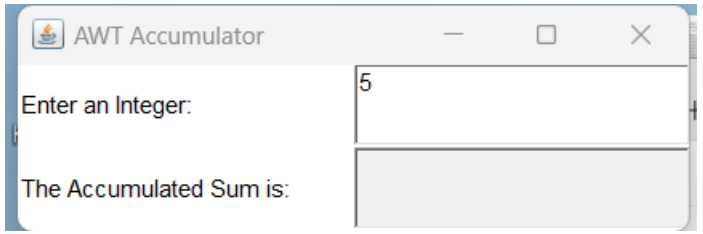


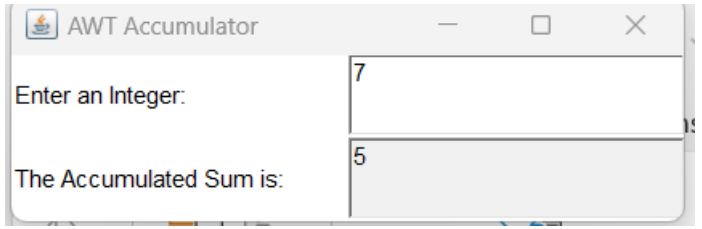


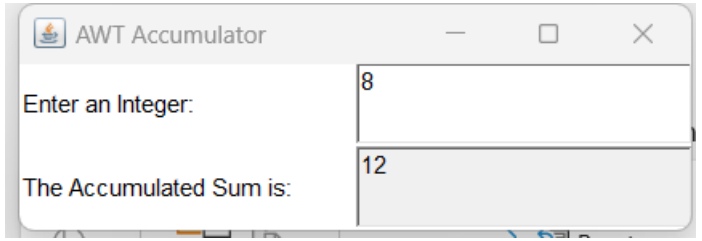


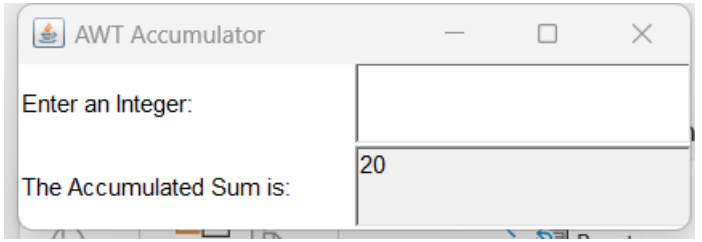
AWT Accumulator

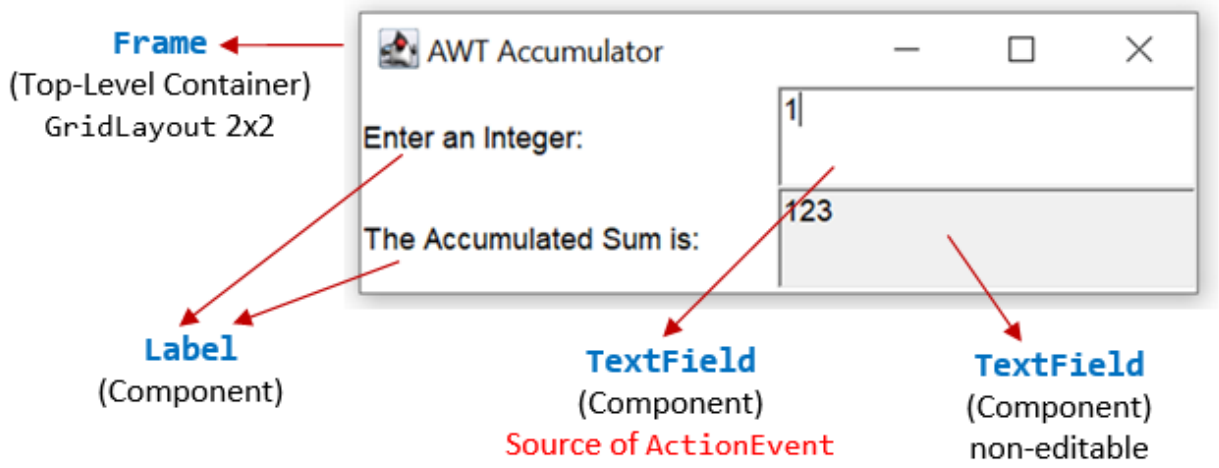
- The top-level container is again the typical java.awt.Frame.
- It contains 4 components: a Label "Enter an Integer", a TextField for accepting user input
- Another Label "The Accumulated Sum is", and another non-editable TextField for displaying the sum.
- The components are arranged in GridLayout of 2 rows 2 columns.
- The program shall accumulate the number entered into the *input* TextField and display the sum in the *output* TextField.












THANK YOU
ANY QUESTION?