1495 UNIVERSITY OF ABERDEEN

# JC2002 Java Programming

Lecture 19: Introduction to graphical user interfaces

# References and learning objectives

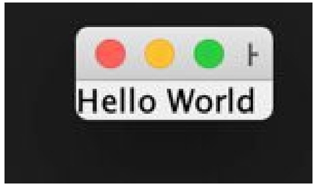
- Today's sessions are mostly based on:
  - Sierra et al., ***Head First Java*** (2nd Ed), O'Reilly, Chapters 12, 13 (available in the library)
  - https://docs.oracle.com/javase/tutorial/uiswing/components/menu.html
  - ***Introduction to Programming with Java***, Chapter 2.14 https://runestone.academy/runestone/books/published/csjava/index.html
- After today's session, you should be able to:
  - Explain the main concepts and terms of graphical user interfaces and
  - Implement simple user interfaces using standard Swing components
  - Select appropriate GUI components for different purposes

## Graphical user interface (GUI)

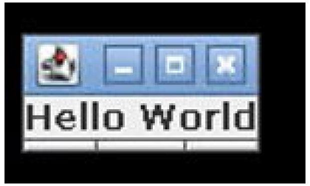
- A **graphical user interface (GUI)** presents a user-friendly mechanism for interacting with an app:
  - GUI (pronounced “GOO-ee”) gives an app a distinctive *look-and-feel*
  - GUI provides apps with consistent, intuitive user interface components giving users a sense of familiarity even with a new app
- GUIs are built from ***GUI components***, also called ***controls*** or ***widgets*** (short for window gadgets)
  - A GUI components is an object with which the user interacts via the mouse, the keyboard or another form of input, e.g., voice recognition

# Java GUI on different platforms

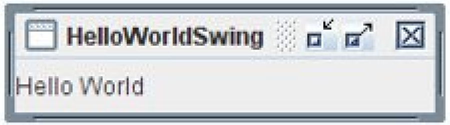
- Java code is *platform independent*: Java GUI uses the GUI components provided by the underlying platform
  - Different platforms give different look-and-feels



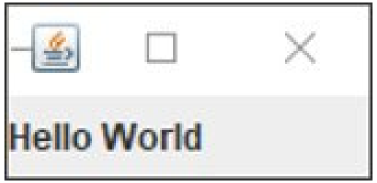
Mac OS



Linux OS



Solaris (Unix) OS



Windows OS

## Java GUI libraries

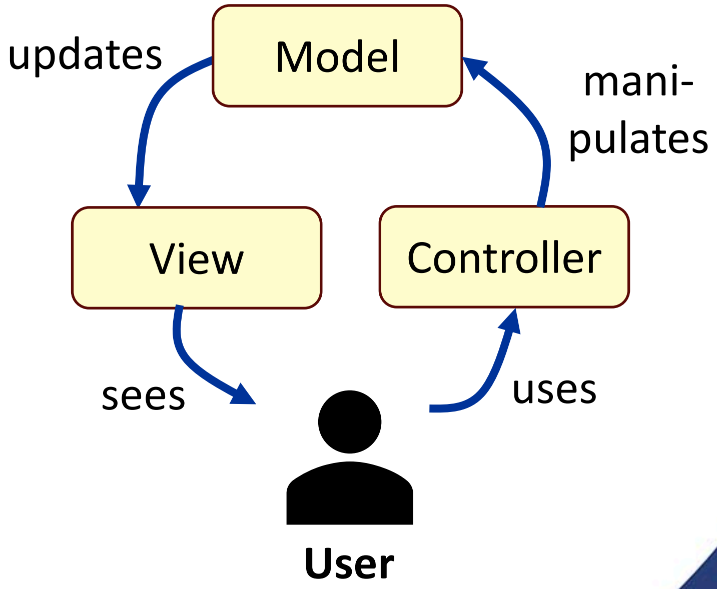
- There are different GUI libraries for Java:
  - **Abstract Window Toolkit (AWT)** was Java’s original GUI library (the oldest of all Java GUIs):
    - AWT is heavyweight and platform dependent.
  - **Swing** was added to the platform in Java SE 1.2:
    - Until recently, Swing was the primary Java GUI technology.
    - Lighter, platform independent, purely for desktop.
  - **JavaFX** was announced in 2007 and released in 2008 as a competitor to Adobe Flash and Microsoft Silverlight:
    - Smaller number of components, better integration to modern devices.

# Java Swing library

- In this course, we use primarily Swing for GUIs
  - Swing is still widely used and focused
  - Swing does the "heavy lifting" in desktop applications
  - Swing uses the common **model-view-controller (MVC)** design pattern
  - Swing is cross-platform (as Java in general) with suitable look-and-feel
  - Swing is extensive and what is learned with Swing is easy to move to JavaFX in the future

# Model-view-controller (MVC)

- In general, a visual component is a composite of three distinct aspects:
    - The way that the component looks when rendered on the screen (*view*)
    - The way the component reacts to the user (controller)
    - The state information associated with the component (model)
- Over the years, MVC architecture has proven itself to be exceptionally effective

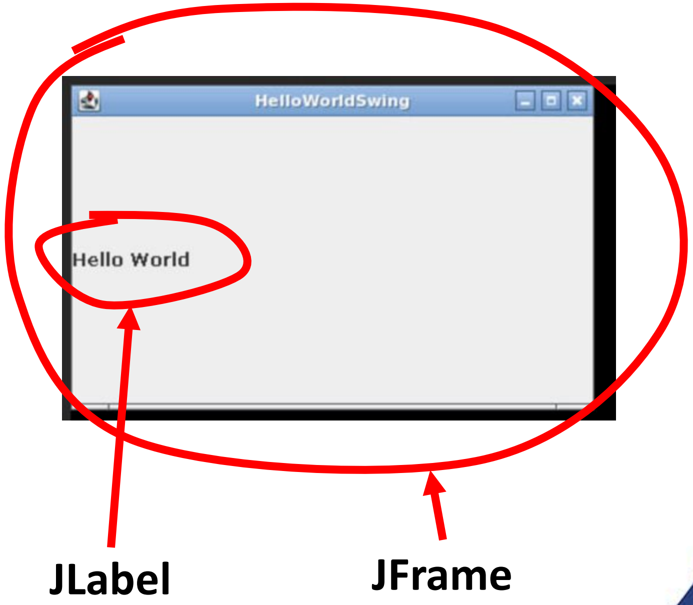


## Java foundation classes (JFC) and Swing

- JFC encompass a group of features for building GUIs and adding rich graphics functionality and interactivity to Java applications
- JFC contains the features:
  - Swing GUI Components
  - Pluggable look-and-feel Support
  - Accessibility API
  - Java 2D API
  - Internationalisation

## Swing components

- In Swing, the GUI is composed of graphical components
  - The graphical components are classes, and to use them, you must declare objects of them
  - Usually, a GUI is built on a **`JFrame`** component, which is a *window* or *container* for GUI
  - Other commonly used components include e.g., **`JLabel`** that can include static text or an image, and **`JButton`** that implements a button that can be pressed



## Example of using JFrame (1)

```java
import javax.swing.*;
public class FrameTest {
  private static void createAndShowGUI() {

    JFrame frame = new JFrame("HelloWorldSwing");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JLabel label = new JLabel("Hello World");
    frame.getContentPane().add(label);

    frame.pack();
    frame.setVisible(true);
  }
  public static void main(String[] args) {
    javax.swing.SwingUtilities.invokeLater(new Runnable() {
      public void run() {
        createAndShowGUI();
      }
    });
  }
}
```

1495 UNIVERSITY OF ABERDEEN

# Example of using JFrame (2)

```java
import javax.swing.*;
public class FrameTest {
  private static void createAndShowGUI() {

    JFrame frame = new JFrame("HelloWorldSwing");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JLabel label = new JLabel("Hello World");
    frame.getContentPane().add(label);

    frame.pack();
    frame.setVisible(true);
  }
  public static void main(String[] args) {
    javax.swing.SwingUtilities.invokeLater(new Runnable() {
      public void run() {
        createAndShowGUI();
      }
    });
  }
}
```

Import Swing classes

Create JFrame component

Create JLabel component

Standard code to run Swing GUI apps

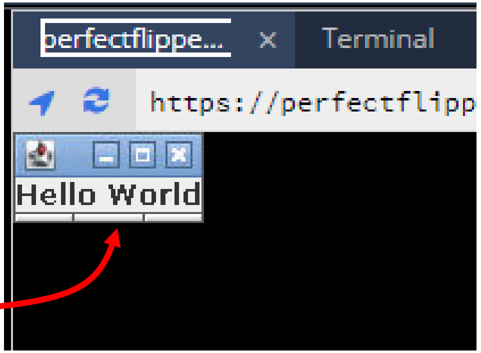
## Example of using JFrame (3)

```java
import javax.swing.*;
public class FrameTest {
  private static void createAndShowGUI() {

    JFrame frame = new JFrame("HelloWorldSwing");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JLabel label = new JLabel("Hello World");
    frame.getContentPane().add(label);

    frame.pack();
    frame.setVisible(true);
  }
  public static void main(String[] args) {
    javax.swing.SwingUtilities.invokeLater(new Runnable() {
      public void run() {
        createAndShowGUI();
      }
    });
  }
}
```

Console:

```
$ javac FrameTest.java
$ java FrameTest
█
```

Virtual Desktop:



You can use mouse to resize the app

## Setting frame size

- You need to determine how big you want the frame to be
  - You can delegate decision to the app with `frame.pack()`
  - The pack method sizes the frame so that all its contents are at or above their preferred sizes
  - An alternative to `pack()` is to establish a frame size explicitly by calling `setSize()` or `setBounds()` (which also sets the frame location)
  - In general, using `pack()` is preferable to calling `setSize()`, since `pack()` layout managers are good at adjusting to platform dependencies and other factors that affect component size

## Example of using setSize

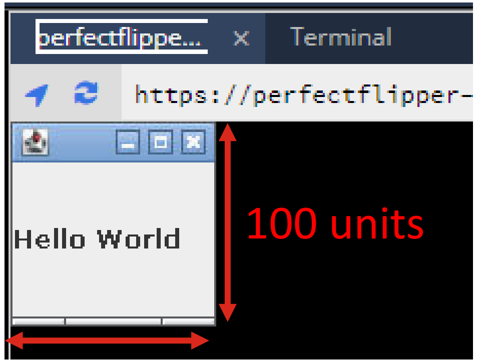
```java
import javax.swing.*;
public class SetSizeTest {
  private static void createAndS

    JFrame frame = new JFrame("H
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JLabel label = new JLabel("Hello World");
    frame.getContentPane().add(label);

    frame.setSize(100,100);
    frame.setVisible(true);
  }
  public static void main(String[] args) {
    javax.swing.SwingUtilities.invokeLater(new Runnable() {
      public void run() {
        createAndShowGUI();
      }
    });
  }
}
```

Using `setSize()` to set the frame size

Console:

```
$ javac SetSizeTest.java
$ java SetSizeTest
█
```

Virtual Desktop:



100 units

## Questions, comments?