GUI Programming

TAO Yida

taoyd@sustech.edu.cn

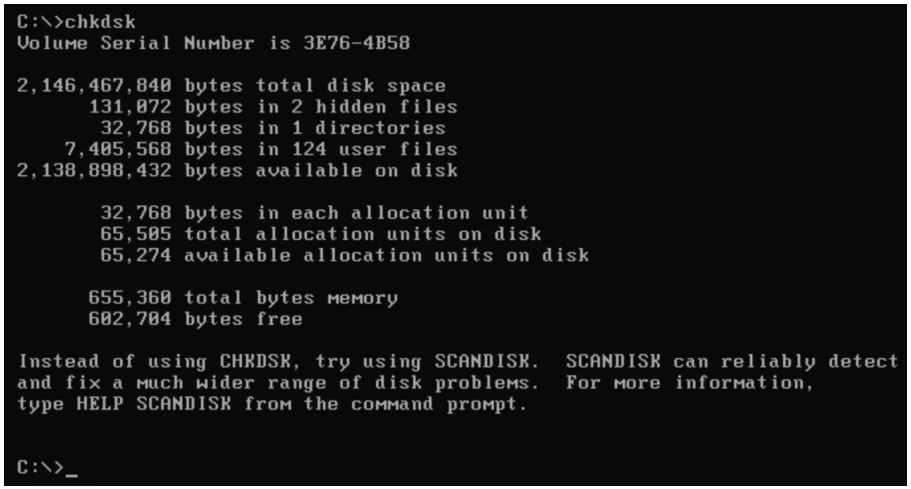
What is GUI?

- The Graphical User Interface (GUI, 图形用户界面), is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators.

Windows 10

GUI vs. CLI

- Before GUI became popular, text-based Command-Line Interface (CLI, 命令行界面) was widely-used (mainly in 1970s and 1980s).
- Because CLIs consume little resources, they are still available in modern computers with GUIs and are widely-used by professionals.



MS-DOS

Java GUI History

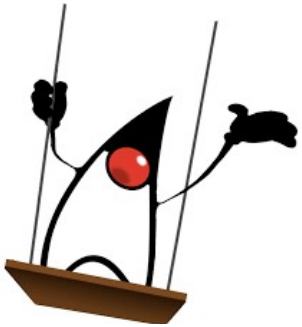
- Abstract Window Toolkit (AWT)
  - JDK 1.0 (1995)
  - Most of AWT's UI components have become obsolete
- Swing
  - JDK 1.2 (1997)
  - Enhancement of AWT
- JavaFX
  - JDK 8 (2008), replacement to Swing
  - Actively maintained and expected to grow in future

Java GUI Programming APIs

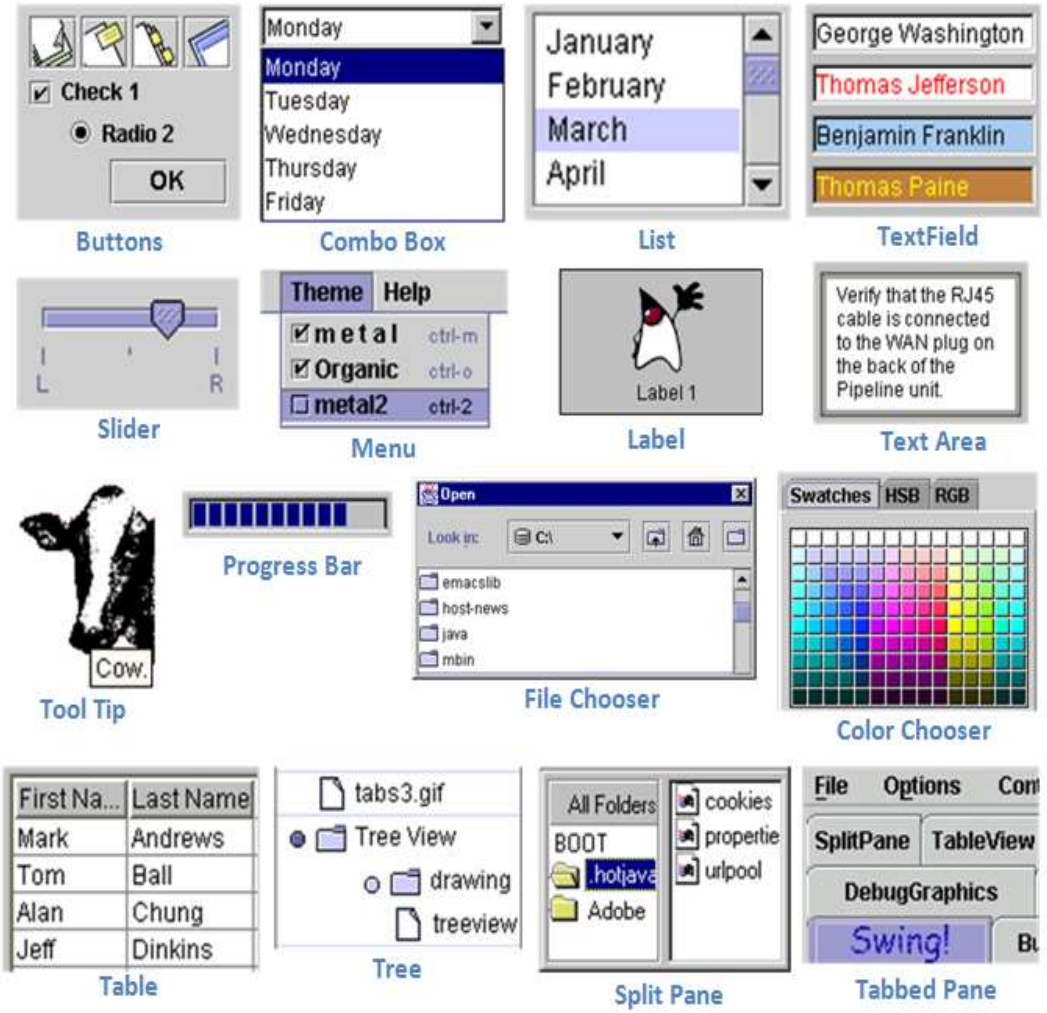
- **AWT** (Abstract Windowing Toolkit): introduced in JDK 1.0
- AWT components are **platform-dependent**. Their creation relies on the operating system's high-level user interface module.
  - For example, creating an AWT check box would cause AWT directly to call the underlying native subroutine that creates a check box.
  - This makes GUI programs written in AWT look like native applications
- AWT contains 12 packages of 370 classes (Swing and FX are more complex, 650+ classes)
  - They are developed by expert programmers with advanced design patterns.
  - Writing your own graphics classes (re-inventing the wheels) is mission impossible!

https://www.ntu.edu.sg/home/ehchua/programming/java/J4a_GUI.html

Java GUI Programming APIs

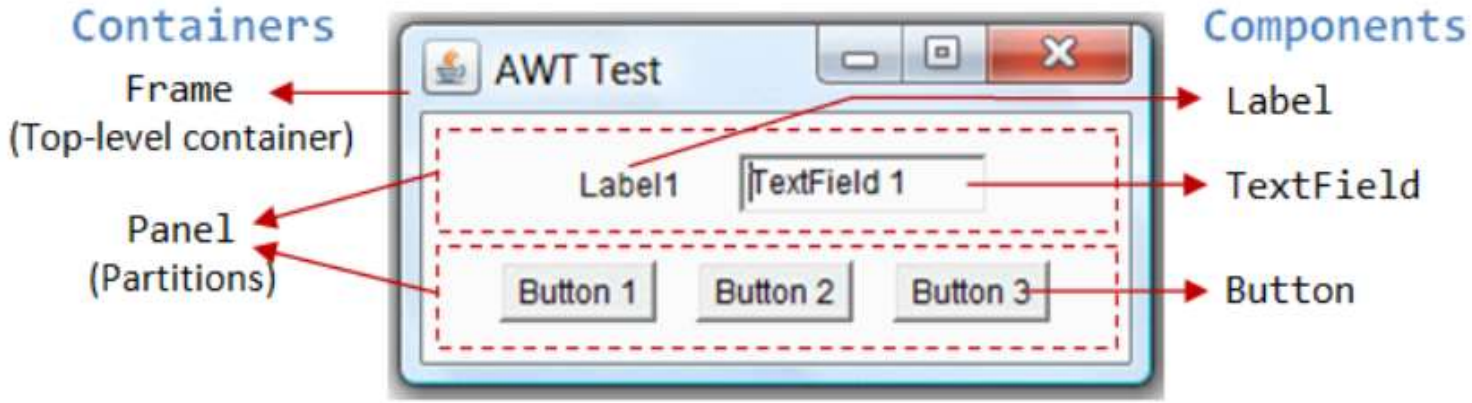


- **Swing**, introduced in 1997 after the release of JDK 1.1, provides a much more comprehensive set of UI widgets than AWT
- Unlike AWT's UI widgets, Swing's are not implemented by platform-specific code. They are written entirely in Java and **platform-independent**.
  - In Swing, user interface elements, such as buttons, menus, and so on, were painted onto blank windows.
- **Pluggable look and feel:** Swing component can have the native platform's "look and feel" or a cross-platform look and feel (the "Java Look and Feel")



Java GUI Core Concepts

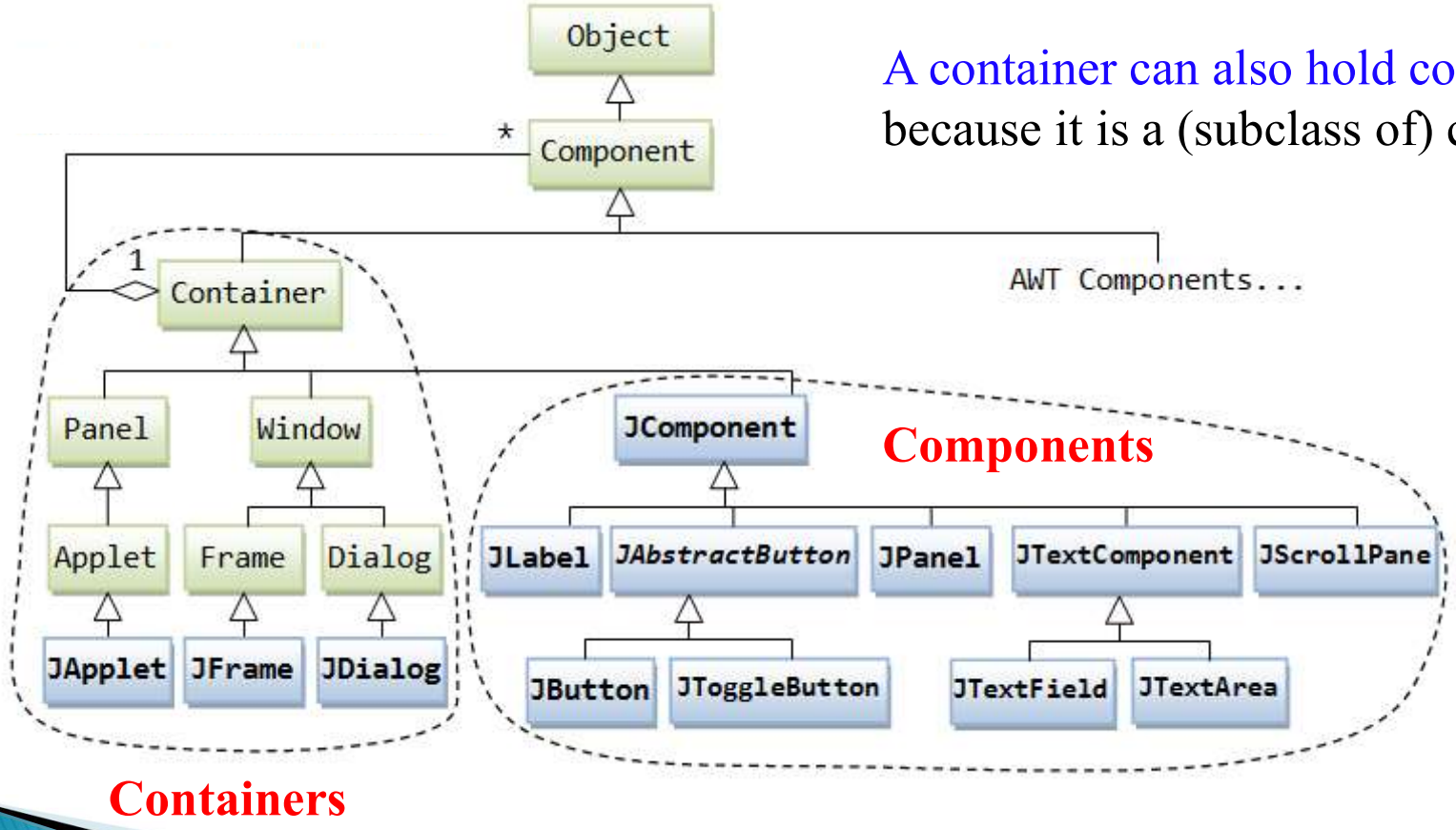
- **Component** (组件): Components are elementary GUI entities, such as Button, Label, and TextField.
- **Container** (容器): used to hold components in a specific layout
- **Event handling** (事件处理): decides what should happen if an event occurs (e.g., a button is clicked)



https://www3.ntu.edu.sg/home/ehchua/programming/java/j4a_gui.html

Java GUI Class Hierarchy

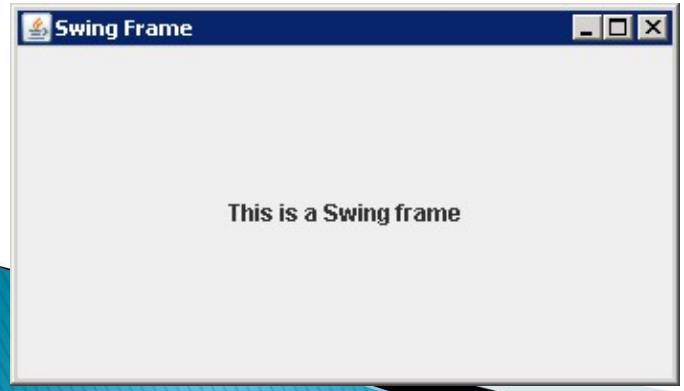
- There are two groups of classes (in package `javax.swing`): **containers** and **components**. A container is used to hold components.

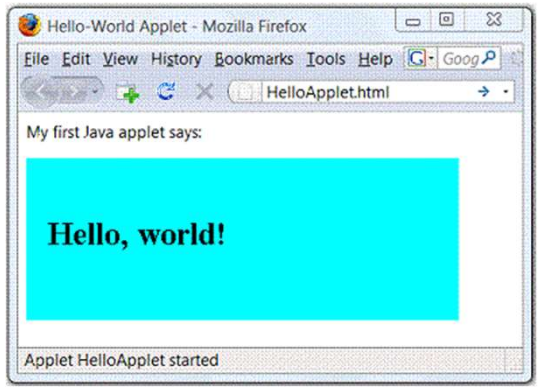


A container can also hold containers because it is a (subclass of) component

Containers: top level container

- A Swing application requires a **top-level container** (a window that is not contained inside another window)
- There are three top-level containers in Swing:
  - **`JFrame (主窗体)`**: used for the application's main window (with an icon, a title, minimize/maximize/close buttons, an optional menu-bar, and a content-pane)
  - **`JDialog (对话框)`**: used for secondary pop-up window (with a title, a close button, and a content-pane).
  - **`JApplet`**: used for the applet's display-area (content-pane) inside a browser's window.





Containers: top level container

- A Swing application requires a **top-level container** (a window that is not contained inside another window)
- There are three top-level containers in Swing:
  - **`JFrame (主窗体)`**: used for the application's main window (with an icon, a title, minimize/maximize/close buttons, an optional menu-bar, and a content-pane)
  - **`JDialog (对话框)`**: used for secondary pop-up window (with a title, a close button, and a content-pane).
  - **`JApplet`**: used for the applet's display-area (content-pane) inside a browser's window.
- There are secondary containers (such as `JPanel面板`) which can be used to group and layout relevant components (布局).

Secondary containers are placed inside a top-level container or another secondary container

Building Our First Swing Program

```
import javax.swing.JFrame;

public class HelloWorld extends JFrame {
    public HelloWorld() {
        super("Our first Swing program");
    }

    public static void main(String[] args) {
        HelloWorld gui = new HelloWorld();
        gui.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
        gui.setSize(800, 600);
        gui.setVisible(true);
    }
}
```

Select a top-level container (mostly JFrame)

Creates a new, initially invisible Frame with the specified title.

Exit the application (process) when the close button is clicked.
Default value HIDE_ON_CLOSE hides the JFrame, but keeps the application running.

Building Our First Swing Program

```
import javax.swing.JFrame;

public class HelloWorld extends JFrame {
    public HelloWorld() {
        super("Our first Swing program");
    }

    public static void main(String[] args) {
        HelloWorld gui = new HelloWorld();
        gui.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
        gui.setSize(800, 600);
        gui.setVisible(true);
    }
}
```

Select a top-level container (mostly JFrame)

Creates a new, initially invisible Frame with the specified title.

By default, a frame has a rather useless size of 0 × 0 pixels, which need to be resized properly

Display the JFrame

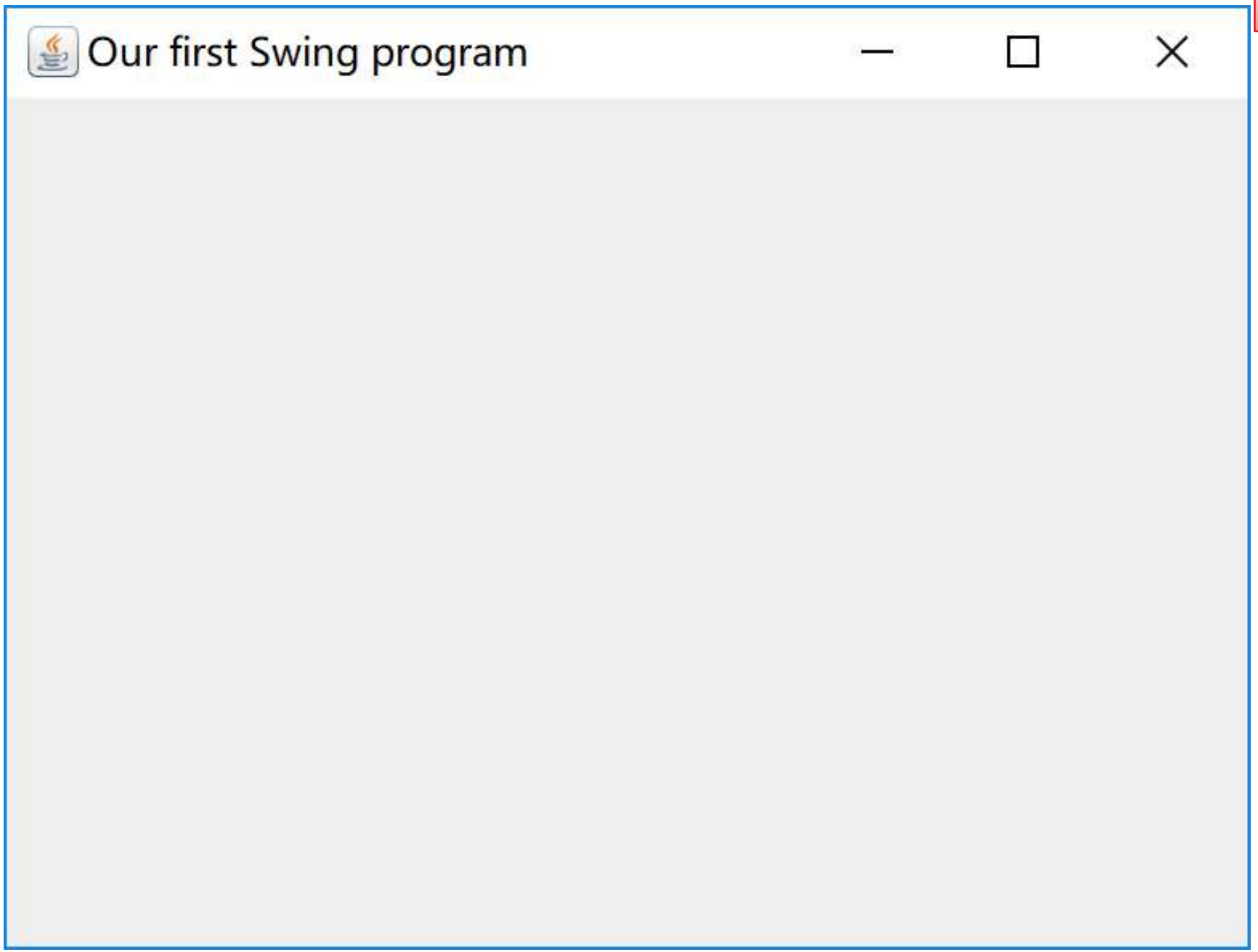
Building Our First Swing Program

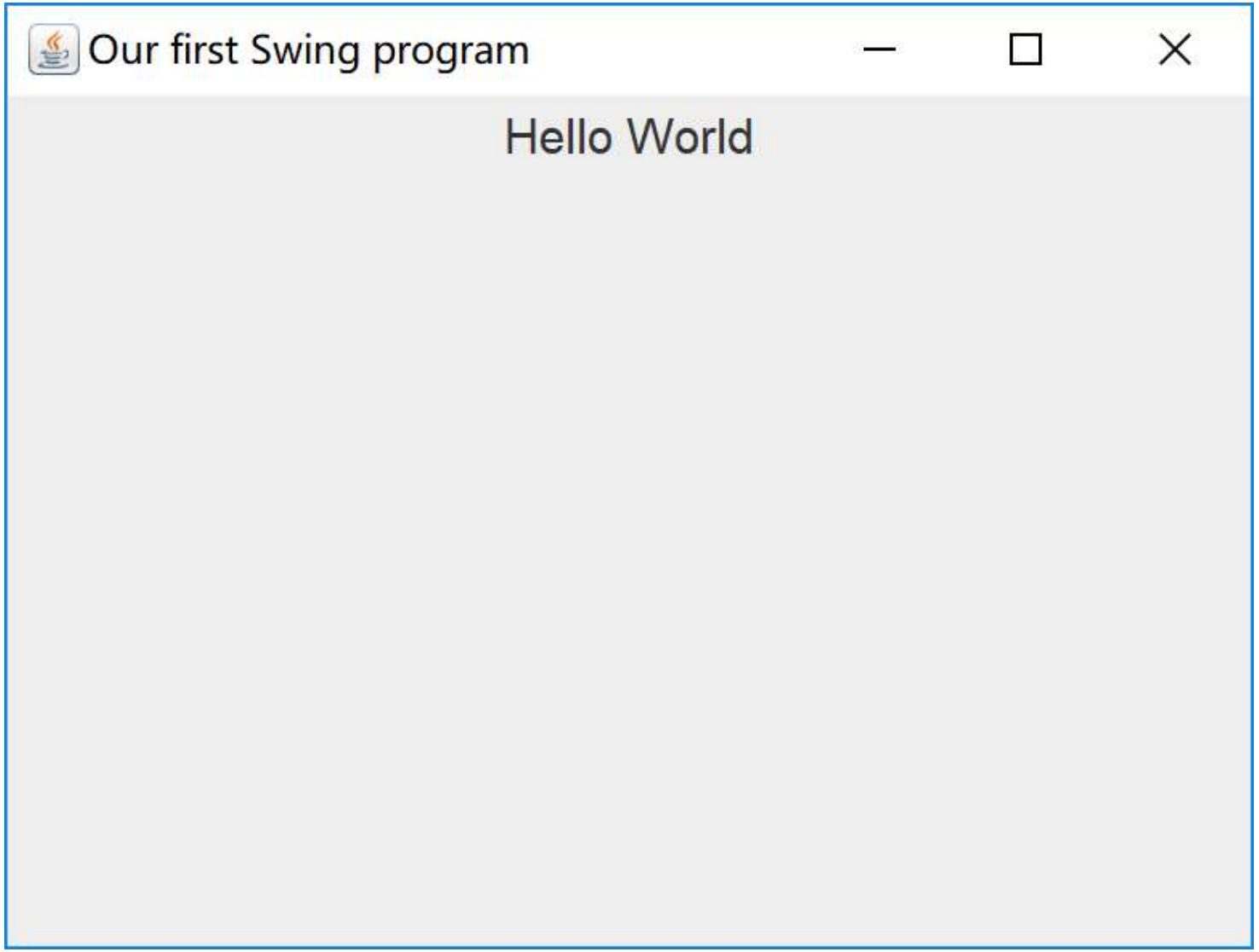
```
import javax.swing.JFrame;

public class HelloWorld extends JFrame {
    public HelloWorld() {
        super("Our first Swing program");
    }

    public static void main(String[] args) {
        HelloWorld gui = new HelloWorld();
        gui.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
        gui.setSize(800, 600);
        gui.setVisible(true);
    }
}
```

These methods are inherited from the superclass





How to add the component?

Building Our First Swing Program

```
public class HelloWorld extends JFrame {

    private JLabel label;

    public HelloWorld() {
        super("Our first Swing program");
        setLayout(new FlowLayout());
        label = new JLabel("Hello World");
        label.setFont(new Font("San Serif", Font.PLAIN, 30));
        add(label);
    }

    public static void main(String[] args) { // same as earlier }
}
```

Declaring GUI components as fields makes it easier to interact with the corresponding objects (An example of Composition)

Specifying layout (how to position GUI components)

Creating GUI component (a label here) and add it to the JFrame (actually its content pane)

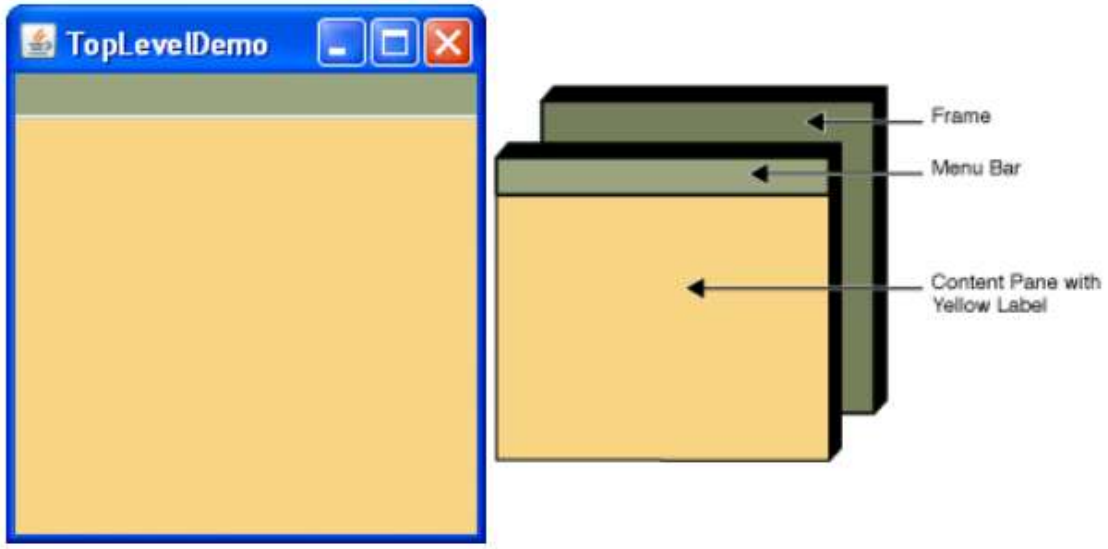
setLayout() and add() are inherited

Building Our First Swing Program

Each GUI component can be contained only once. If a component is already in a container and you try to add it to another container, the component will be removed from the first container and then added to the second.

Content Pane

- `JComponents` shall not be added onto the top-level container (e.g., `JFrame, JApplet`) directly
- `JComponents` must be added onto the so-called content pane (`java.awt.Container`) of the top-level container



You can optionally add a menu bar to a top-level container. The menu bar is by convention positioned within the top-level container, but outside the content pane.

Content Pane

- `JComponents` shall not be added onto the top-level container (e.g., `JFrame, JApplet`) directly
- `JComponents` must be added onto the so-called content pane (`java.awt.Container`) of the top-level container
- If a component is added "directly" into a `JFrame`, it is actually added into the content-pane of `JFrame` instead

```
// Suppose that "this" is a JFrame
add(new JLabel("add to JFrame directly"));
// is executed as
getContentPane().add(new JLabel("add to JFrame directly"));
```

https://www3.ntu.edu.sg/home/ehchua/programming/java/j4a_gui.html#zz-8.

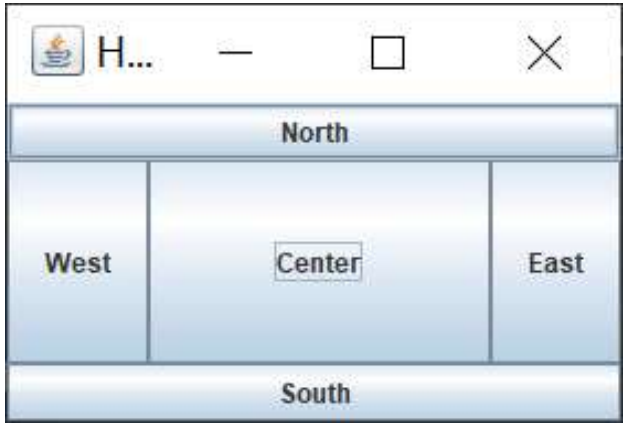
JPanel

JPanel is a container that can store a group of components and organize components in various layouts

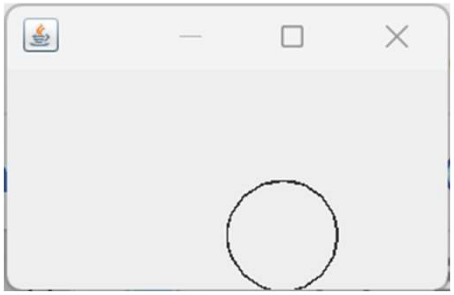
```
public class JPanelTest {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Hello World");

        //Create a panel and add components to it.
        JPanel panel = new JPanel(new BorderLayout());
        panel.add(new JButton("North"), BorderLayout.NORTH);
        panel.add(new JButton("South"), BorderLayout.SOUTH);
        panel.add(new JButton("West"), BorderLayout.WEST);
        panel.add(new JButton("East"), BorderLayout.EAST);
        panel.add(new JButton("Center"), BorderLayout.CENTER);

        frame.setContentPane(panel);
        frame.setSize(300,200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```



Draw a Component



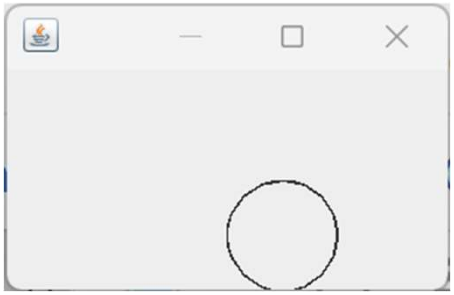
- To draw on a component, you define a class that extends `JComponent` and override the `paintComponent` method in that class.
- The `paintComponent` method is like a canvas where you draw things on your window.
- It takes one parameter of type `Graphics g`, which is like your paintbrush and gives you drawing tools.

```
class MyCircle extends JComponent{
    int X = 100;
    int Y = 50;

    @Override
    public void paintComponent(Graphics g){
        super.paintComponent(g);
        g.drawOval(X,Y,50,50);
    }

    @Override
    public Dimension getPreferredSize(){
        return new Dimension(200, 100);
    }
}
```

Draw a Component



- Never call the `paintComponent` method yourself. It is called automatically whenever a part of your application needs to be redrawn, and you should not interfere with this automatic process.

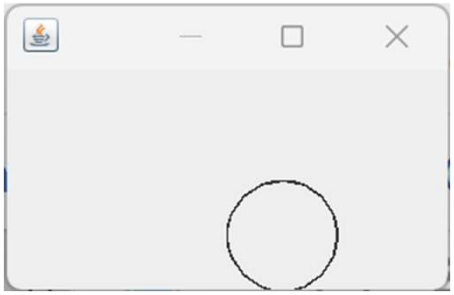
- Java automatically calls `paintComponent` method when:
  - The window first opens
  - The window is resized
  - The window's state is changed (e.g., minimize then restored)

```
class MyCircle extends JComponent{
    int X = 100;
    int Y = 50;

    @Override
    public void paintComponent(Graphics g){
        super.paintComponent(g);
        g.drawOval(X,Y,50,50);
    }

    @Override
    public Dimension getPreferredSize(){
        return new Dimension(200, 100);
    }
}
```

Draw a Component



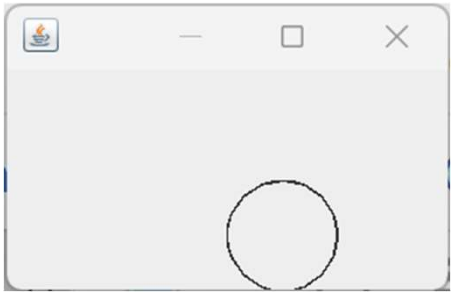
- **Visual changes**: if you need to force repainting of the screen, call the `repaint()` method, which calls paintComponent for all components
- **Layout changes**: if you need to recalculate the layout, call the `revalidate()` method.
- Sometimes we can call both `revalidate()` and `repaint()` to recalculate and redraw everything

```
class MyCircle extends JComponent{
    int X = 100;
    int Y = 50;

    @Override
    public void paintComponent(Graphics g){
        super.paintComponent(g);
        g.drawOval(X,Y,50,50);
    }

    @Override
    public Dimension getPreferredSize(){
        return new Dimension(200, 100);
    }
}
```

Draw a Component



- (0,0) is a the top-left corner
- `super.paintComponent(g)` clears the panel and prepares it for new drawing
- Recommended; otherwise, new drawings are drawn on top of old ones and may cause weird visual effects

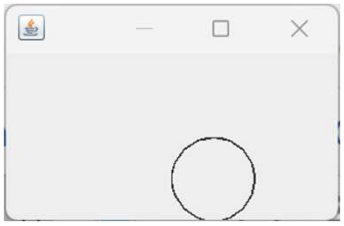
```
class MyCircle extends JComponent{
    int X = 100;
    int Y = 50;

    @Override
    public void paintComponent(Graphics g){
        super.paintComponent(g);
        g.drawOval(X,Y,50,50);
    }

    @Override
    public Dimension getPreferredSize(){
        return new Dimension(200, 100);
    }
}
```

Draw a Component

- A component should tell its users how big it would like to be. Override the `getPreferredSize` method and return an object of the `Dimension` class with the preferred width and height
- When you fill a frame with one or more components, and you simply want to use their preferred size, call the `pack()` method instead of the setSize method



```
public class GraphicsDemo {
    public static void main(String[] args) {
        JFrame frame = new JFrame();
        MyCircle circle = new MyCircle();
        frame.add(circle);
        frame.pack();
        frame.setVisible(true);
    }
}

class MyCircle extends JComponent{
    int X = 100;
    int Y = 50;

    @Override
    public void paintComponent(Graphics g){
        super.paintComponent(g);
        g.drawOval(X,Y,50,50);
    }

    @Override
    public Dimension getPreferredSize(){
        return new Dimension(200, 100);
    }
}
```