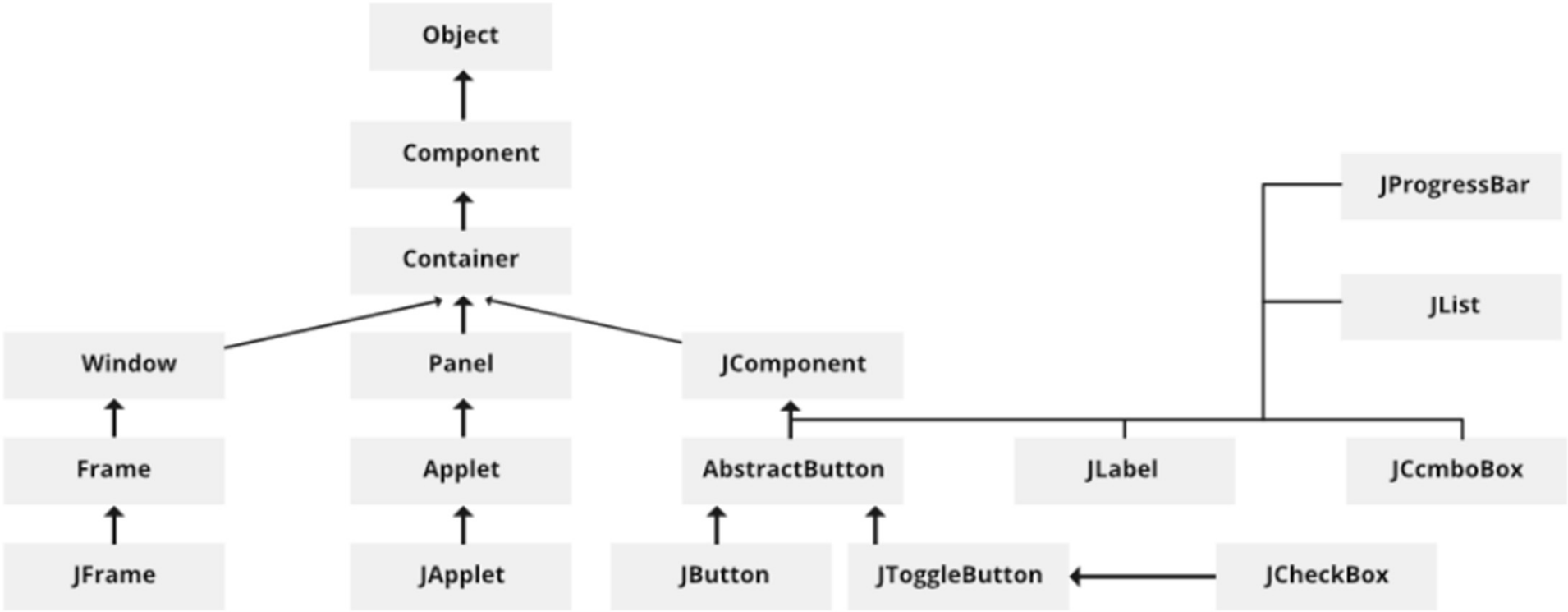
# 20251117 Monday

- Last class: frame, label, checkbox
- Objects, object inside another object
- Questions from last class:
  1 closed but still running?
  2 layoutManager?
- Today, continue, the C in MVC



- frameObject.setDefaultCloseOperation(**JFrame.**EXIT_ON_CLOSE**);**
- Concept about foreground and background
- ~~In Operating System, the parent process waits child process's "finish/return".~~

- Java Swing uses layout managers to control the positioning and sizing of components within a container.

## A Visual Guide to Layout Managers

Several AWT and Swing classes provide layout managers for general use:

- `BorderLayout`
- `BoxLayout`
- `CardLayout`
- `FlowLayout`
- `GridBagLayout`
- `GridLayout`
- `GroupLayout`
- `SpringLayout`

- A Visual Guide to Layout Managers (The Java™ Tutorials > Creating a GUI With Swing > Laying Out Components Within a Container)

# First Practice

- Panel
- Layout manager: north/south/west/east
- Input in “text area”?

```
import javax.swing.*;
import java.awt.BorderLayout;

public class FirstPractice{
    FirstPractice(){
        JFrame frame = new JFrame("Thanksgiving is coming");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JPanel panel = new JPanel();
        panel.setLayout( new BorderLayout() );

        JButton button = new JButton("Button");
        JLabel label = new JLabel("Label");
        JCheckBox checkBox = new JCheckBox("Checkbox");
        JTextArea textArea = new JTextArea("TextArea");

        panel.add(button, BorderLayout.NORTH);
        panel.add(label, BorderLayout.EAST);
        panel.add(checkBox, BorderLayout.WEST);
        panel.add(textArea, BorderLayout.SOUTH);

        frame.add(panel);

        frame.setSize(1000, 1000);
        frame.setLocation(500,500);
        frame.setVisible(true);
    }

    public static void main(String[] args){ new FirstPractice(); }
}
```

# Frame vs. Panel

- JFrame is a top-level container, while JPanel is a general-purpose container that is typically nested within a JFrame (or another JPanel).
- JFrame provides the windowing functionality (title bar, controls), while JPanel focuses on organizing and grouping components within a window.
- JFrame defines the application window, whereas JPanel is used to structure the content within that window.

# Event Listeners (Second Practice)

- Interface: “ActionListener”.

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

public class MyFirstActionListener implements ActionListener{

    private int count = 0;

    public void actionPerformed(ActionEvent event){
        count++;
        //reference/point to the original source(button)
        JButton clickedButton = (JButton) event.getSource();
        clickedButton.setText("Clicks: " + count);
    }
}
```

```java
import javax.swing.JFrame;
import javax.swing.JButton;

public class SecondPractice{

    public static void main(String[] args){
        JFrame frame = new JFrame("CSC1850-20251117");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JButton button = new JButton("Click:");
        frame.add(button);

        MyFirstActionListener listener = new MyFirstActionListener();
        button.addActionListener(listener);

        frame.setSize(500, 500);
        frame.setVisible(true);
    }
}
```

- The listener can only control the source object through "event.getSource()"

- How to create a listener that can control another object?

- The listener can only control the source object through "event.getSource()"

- How to create a listener that can control another object?
  Pass the object to new listener through CONSTRUCTOR!

# Third Practice

- Constructor!

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

public class MySecondActionListener implements ActionListener{

    private int count = 0;
    private JTextArea ta;

    public MySecondActionListener(JTextArea ta){
        this.ta = ta;
    }

    public void actionPerformed(ActionEvent event){
        count++;
        ta.setText("Clicks: " + count);
    }
}
```

```java
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JButton;
import javax.swing.JTextArea;

public class ThirdPractice{

    public static void main(String[] args){
        JFrame frame = new JFrame("CSC1850-20251117");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JPanel panel = new JPanel();

        JButton button = new JButton("Click here");
        panel.add(button);

        JTextArea ta = new JTextArea("Clicks: 0");
        panel.add(ta);

        MySecondActionListener listener = new MySecondActionListener(ta);
        button.addActionListener(listener);

        frame.add(panel);

        frame.setSize(500, 500);
        frame.setVisible(true);
    }
}
```

- From here, explore “Swing” and “JavaFX” yourself.

- New HW:
  GUI to accept TWO numbers and show the sum of them.