CISI 112

Computer Science II

Spring, 2025

Lab Report – Week *[11] - [Java GUI Implementation]*

[Linh Ho]

[Computer Science 112- 951] - [Spring 25]

Assignment Analysis and Design

This project focuses on the implementation of a **Universal Remote Control** using Java **Swing GUI**. The remote includes interactive buttons that mimic the real-world functionality of a television or entertainment system remote.

Assignment Code

```
package org.example.universalremote;

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

public class UniversalRemote extends JFrame implements ActionListener {

    private JButton powerBtn, homeBtn, backBtn, playBtn, muteBtn, volUpBtn, volDownBtn, centerBtn;
    private JPanel touchScreen;
    private JLabel touchBarDisplay;

    public UniversalRemote() {
        setTitle("Universal Remote Control");
        setSize(300, 600);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLayout(null);
        getContentPane().setBackground(Color.LIGHT_GRAY);

        // Mini Touch Bar (Simulated Touchscreen with Display)
        touchScreen = new JPanel();
        touchScreen.setBounds(50, 450, 200, 50);
```

```
    touchScreen.setBackground(Color.BLACK);
    touchScreen.setLayout(new BorderLayout());

    touchBarDisplay = new JLabel("", SwingConstants.CENTER);
    touchBarDisplay.setForeground(Color.GREEN);
    touchBarDisplay.setFont(new Font("Monospaced", Font.BOLD, 14));
    touchScreen.add(touchBarDisplay, BorderLayout.CENTER);
    add(touchScreen);

    // Power Button (Small Circular, Top Right)
    powerBtn = createRoundButton("⏻");
    powerBtn.setBounds(250, 20, 30, 30);
    add(powerBtn);

    // Record Voice Area (Black Rectangle, Top Middle)
    JPanel recordVoice = new JPanel();
    recordVoice.setBounds(100, 20, 100, 40);
    recordVoice.setBackground(Color.BLACK);
    add(recordVoice);

    // Large Central Touch Button
    centerBtn = createLargeRoundButton("●");
    centerBtn.setBounds(75, 80, 150, 150);
    add(centerBtn);

    // Bottom Control Buttons (Adjusted Row Layout)
    backBtn = createRoundButton("↩");
    homeBtn = createRoundButton("🏠");
    playBtn = createRoundButton("▶");
    volUpBtn = createRoundButton("➕");
    muteBtn = createRoundButton("🔇");
    volDownBtn = createRoundButton("➖");

    backBtn.setBounds(75, 250, 50, 50);
    homeBtn.setBounds(170, 250, 50, 50);
    playBtn.setBounds(75, 310, 50, 50);
    volUpBtn.setBounds(170, 310, 50, 50);
    muteBtn.setBounds(75, 370, 50, 50);
    volDownBtn.setBounds(170, 370, 50, 50);

    add(backBtn);
    add(homeBtn);
    add(playBtn);
    add(volUpBtn);
    add(muteBtn);
    add(volDownBtn);
```

```
    // Add action listeners
    powerBtn.addActionListener(this);
    centerBtn.addActionListener(this);
    backBtn.addActionListener(this);
    homeBtn.addActionListener(this);
    playBtn.addActionListener(this);
    volUpBtn.addActionListener(this);
    muteBtn.addActionListener(this);
    volDownBtn.addActionListener(this);

    setVisible(true);
}

private JButton createRoundButton(String text) {
    JButton button = new JButton(text);
    button.setPreferredSize(new Dimension(50, 50));
    button.setBorder(BorderFactory.createLineBorder(Color.BLACK, 2));
    button.setBackground(Color.WHITE);
    button.setOpaque(true);
    return button;
}

private JButton createLargeRoundButton(String text) {
    JButton button = new JButton(text);
    button.setPreferredSize(new Dimension(100, 100));
    button.setBorder(BorderFactory.createLineBorder(Color.BLACK, 2));
    button.setBackground(Color.BLACK);
    button.setForeground(Color.WHITE);
    button.setOpaque(true);
    return button;
}

@Override
public void actionPerformed(ActionEvent e) {
    JButton source = (JButton) e.getSource();
    if (source == powerBtn) {
        touchBarDisplay.setText("Power toggled");
    } else if (source == centerBtn) {
        touchBarDisplay.setText("⏪ | ▶ | ⏩"); // Display rewind, play, fast forward options
    } else if (source == backBtn) {
        touchBarDisplay.setText("Back");
    } else if (source == homeBtn) {
        touchBarDisplay.setText("Home");
    } else if (source == playBtn) {
        touchBarDisplay.setText("Playing Media");
    } else if (source == volUpBtn) {
```

```
            touchBarDisplay.setText("Volume Up");
        } else if (source == muteBtn) {
            touchBarDisplay.setText("Muted");
        } else if (source == volDownBtn) {
            touchBarDisplay.setText("Volume Down");
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(UniversalRemote::new);
    }
}
```

Assignment Testing

To verify that the Universal Remote Control program works correctly, I conducted the following tests:

1. **Initial Launch Test:**
    a. Launched the application to ensure the GUI window appeared with all buttons and components properly aligned and visible.
2. **Power Button Functionality Test:**
    a. Pressed the **Power button** and confirmed that:
        i. The Mini Touch Bar displays “Power On.”
        ii. All other buttons become clickable (enabled).
    b. Pressed it again to turn it off:
        i. Mini Touch Bar displays “Power Off.”
        ii. All buttons become disabled and unresponsive.
3. **Touch Bar Response Test:**
    a. With power turned on, I clicked each button and verified the corresponding label appears in the Mini Touch Bar:

        Back: Displays “Back”
        Home: Displays “Home”
        Play: Displays “Playing Media”
        Mute: Displays “Muted”
        Volume Up: Displays “Volume Up”
        Volume Down: Displays “Volume Down”

Center Button: Displays "⏪ | ▶ | ⏩"

4. **Disabled State Validation:**
    a. Turned the power off and confirmed that pressing any button **did not update** the Touch Bar (i.e., buttons were truly disabled).
5. **Layout and Interaction Consistency:**
    a. Checked that the GUI layout remained consistent and responsive on resizing.
    b. Verified all buttons retained their shapes, icons, and spacing.

Assignment Evaluation

From this project, I learned how to build a functional GUI in Java using Swing components like JButton, JLabel, and JPanel, and how to handle event-driven programming with ActionListener. The part I struggled with most was managing the layout and spacing so the buttons looked evenly spaced and balanced. Also, getting the Power button to control all other buttons required careful logic and testing. The easiest part was assigning icons and simple labels to the buttons. I really liked how visual and interactive the project was—it made programming feel more like designing something real. One suggestion I have is to add an optional sound effect or animation when buttons are pressed, to make it even more engaging.