Program 3 Assignment (LO4) (LO5) - 25 points

Completion requirements

Due: Saturday, October 11, 2025, 2:59 AM

Programming Task: Write and compile a program that displays a button that can be clicked multiple times.

Submission Requirements: - Part 1: Screenshot of your program's code - Part 2: Screenshot of the compiled output

Essay Task:

After compiling your program, write a 3–5 page research-based essay analyzing: - Practical applications of Java GUI in real-world scenarios - The significance of Java GUI functions for professionals in the field - A comparison and justification of your choice between Swing or JavaFX - Support your insights with a literature review, ensuring a well-researched discussion. Include both screenshots in your essay.

Formatting Guidelines: - Follow APA 7 style - Main body: 3–5 pages - Refer to the written assignment rubric under the "Start Here" tab

Deadline: Saturday at 11:59 PM EST

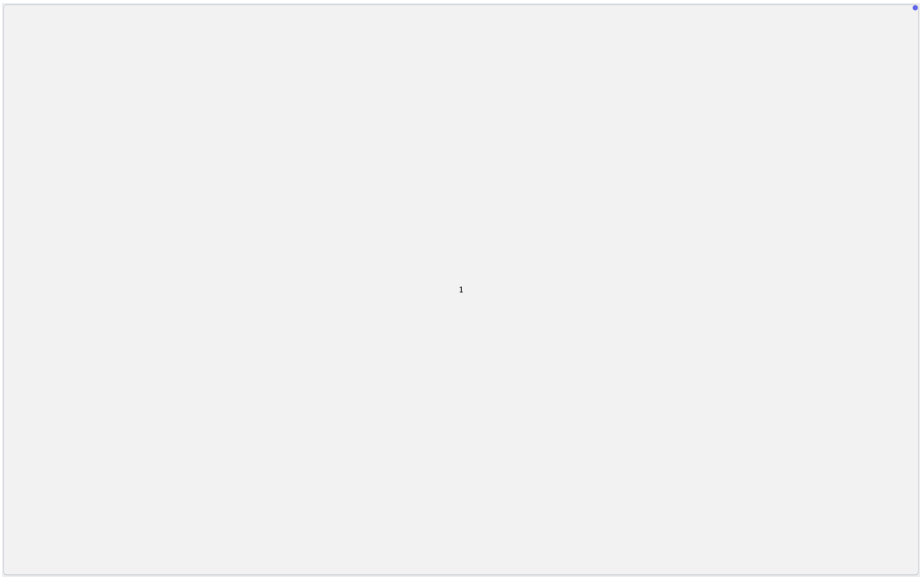
Tip: Reading Lesson 15 will help with this assignment.

Notes:

- This assignment must be formatted in APA Style 7th edition.
- Please refer to the written assignment rubric on the Start Here tab for this paper.
- This paper is due Saturday at 11:59 PM EST.

```
import java.awt.BorderLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.SwingUtilities;

public class Click extends JFrame {
    private int count;
    private final JButton button;

    public Click() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        count = 0;
        button = new JButton(Integer.toString(count));
        button.addActionListener(event -> {
            count++;
```

```
            button.setText(Integer.toString(count));
        });

        JPanel content = new JPanel(new BorderLayout());
        content.add(button);
        setContentPane(content);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            Click window = new Click();
            window.setVisible(true);
        });
    }
}
```



The Enduring Relevance of Java GUI: From Legacy Systems to Modern Applications

Introduction

Java is renowned for its "write once, run anywhere" philosophy. While much of the contemporary discourse revolves around web and mobile development, Java's capabilities in creating robust, platform-independent Graphical User Interfaces (GUIs) remain profoundly significant. Java GUI frameworks, primarily the Abstract Window Toolkit (AWT), Swing, and the modern JavaFX, have provided the foundation for a vast array of desktop applications that are integral to various industries.

Practical Applications of Java GUI in Real-World Scenarios

The versatility of Java's GUI frameworks has led to their adoption in numerous domains, from scientific research to enterprise-level software.

In the realm of **scientific and medical computing**, Java GUI is instrumental. Applications like **ImageJ**, a public domain image processing program developed at the National Institutes of Health, are built on a Java GUI and are used extensively in scientific research for analyzing and processing images. Similarly, financial trading platforms often employ Java to deliver real-time data feeds, charting, and order execution interfaces to traders. For example, **Thinkorswim by TD Ameritrade** is a popular trading platform built with Java that provides a rich, customizable desktop experience.

Significance of Java GUI Functions for Professionals

For software developers, UI/UX designers, and end-users, the functions provided by Java GUI frameworks offer distinct and significant advantages. These features go beyond mere component rendering; they encompass the entire architecture of user interaction and application design.

For **developers**, the **Model-View-Controller (MVC) architecture**, or variations thereof, is a cornerstone of both Swing and JavaFX. This separation of concerns allows for cleaner, more maintainable code. The data (Model) is decoupled from its presentation (View) and the user interaction logic (Controller). This modularity enables development teams to work on different parts of the application concurrently and simplifies debugging and testing.

For **UI/UX designers**, modern Java GUI, particularly JavaFX, offers a significant leap forward. JavaFX introduces features that align with modern design principles. It supports **Cascading Style Sheets (CSS)** for styling, allowing designers to control the look and feel of an application without altering the Java code.

For the **end-user**, the primary benefit is a stable, responsive, and feature-rich application.

A Comparative Analysis: Swing vs. JavaFX

When developing a new desktop application in Java, the primary choice is between the legacy Swing framework and the modern JavaFX.

Swing, introduced in 1997 as a successor to AWT, has been the workhorse of Java GUI development for decades. Its major advantage is its maturity and the vast amount of documentation, third-party libraries, and developer knowledge available. Swing components are "lightweight" as they are written entirely in Java, which gives them a consistent look and feel across all platforms. However, its architecture is showing its age. Customizing the look and feel of Swing applications can be cumbersome, often requiring deep dives into complex "pluggable look and feel" APIs.

JavaFX, on the other hand, was designed from the ground up to be the next-generation GUI toolkit for Java. It addresses many of Swing's shortcomings. Its key advantages include:

- **Rich Internet Application (RIA) Features**: JavaFX was built to support features expected in modern applications, including audio/video playback, animations, and web components.
- **FXML and Scene Builder**: JavaFX introduces FXML, an XML-based language for defining user interfaces. This allows for a declarative approach to UI design, separating the interface layout from the application logic. Tools like Scene Builder provide a visual drag-and-drop environment for creating FXML layouts, empowering designers and accelerating

development.

Pick swing for simple app.

Conclusion

Java's role in GUI development remains robust and critically important in a world often dominated by web interfaces.

References

- Deitel, P., & Deitel, H. (2018). *Java How to Program, Early Objects* (11th ed.). Pearson.
- JetBrains. (2023). *The State of Developer Ecosystem 2023*. Retrieved from https://www.jetbrains.com/lp/devecosystem-2023/
- Schildt, H. (2018). *Java: The Complete Reference* (11th ed.). McGraw-Hill Education.