**Q1: Answer the Following Question. [Marks: 03]**

1. setPreferredSize()
2. setHorizontalAlignment()
3. setIcon()
4. setDefaultCloseOperation()
5. setModel()
6. setFont()
7. setSelectionBackground()
8. setVisible()
9. setSelected()
10. toggles() and determinate mode (specific progress value)

**Q2: Answer the Following Question**   **[Marks: 05]**

1. What is Swing in Java and how does it differ from AWT?

Swing is a Java GUI toolkit that provides a richer set of components than AWT and is built on top of AWT but is platform-independent. AWT relies on native code for components, while Swing components are lightweight and written entirely in Java.

1. Explain the concept of "lightweight" and "heavyweight" components in Swing.

Lightweight components in Swing are drawn using Java code and don’t rely on native OS resources, while heavyweight components in AWT depend on the OS for rendering.

1. Describe the Model-View-Controller (MVC) architecture as applied in Swing.

In Swing's MVC architecture, the Model represents data, the View displays the data, and the Controller handles interactions, keeping concerns separate.

1. What is awt in java?

AWT (Abstract Window Toolkit) in Java is a platform-dependent, native code-based GUI toolkit used for creating simple graphical user interfaces.

1. Explain the concept of Look and Feel in Swing and how it can be customized.

Look and Feel in Swing refers to the appearance and behavior of GUI components, and it can be customized using the UIManager to match different themes or operating systems.

**Q3: Answer the Following Question.**  **[Marks: 03]**

1. Analyze the differences between windowStateChanged and windowClosing events in a Swing application.

**windowStateChanged** event occurs when the window's state (like maximized, minimized) changes, while **windowClosing** is triggered when the window is about to close, offering a chance to handle the close operation.

1. Compare and contrast JPopupMenu and JMenu. How do their use cases and interactions with the user differ, and when would you use one over the other?

**JPopupMenu** is a context menu that appears on right-click, typically floating over other components, while **JMenu** is part of a menu bar with items that drop down when clicked. Use **JPopupMenu** for contextual actions, and **JMenu** for top-level navigation.

1. What is the purpose of a JSeparator within a JMenu versus a JPopupMenu? How does its placement affect the organization of menu items in each context?

**JSeparator** in both **JMenu** and **JPopupMenu** visually divides menu items, organizing them into groups. In **JMenu**, it organizes items in a structured menu bar, while in **JPopupMenu**, it separates context-specific actions.

***CLO 02: Implement software design patterns as a part of software construction activity***

**Q4: Compare the following Java collections in terms of:**  **[Marks: 4]**

