Object Oriented Programming

Project Report

Blood Bank Management System

Group Members:

1. ***AIMA MALIK (001)***
2. ***AYESHA NADEEM (020)***
3. ***Khadija Shafqat (051)***
4. ***MAHNOOR SAJJAD (054)***
5. ***Sadia (075)***

Department: Computer Science

Batch: BCS-II (A)

Submitted To: Sir Adeel Khalid

Instructor signature: _____

In Java, both AWT (Abstract Window Toolkit) and Swing are libraries used for creating graphical user interfaces (GUIs), but they serve different purposes and have distinct features:

AWT (Abstract Window Toolkit):

1. Purpose:

- AWT is the original GUI toolkit for Java, introduced in JDK 1.0.
- It provides a set of classes for building GUI components and handling user input events.
- AWT components are lightweight wrappers around native GUI components provided by the operating system.

2. Key Features:

- **Component Classes**: AWT includes classes like Button, Label, TextField, Checkbox, etc., for creating basic GUI components.
- **Layout Managers:** AWT provides layout managers (FlowLayout, BorderLayout, GridLayout, etc.) for arranging components within containers (Panel, Frame, Window).
- **Event Handling:** AWT supports event handling through listeners (ActionListener, MouseListener, etc.) to respond to user interactions.
- **Graphics and Fonts:** AWT provides classes (Graphics, Font, Color, etc.) for drawing shapes, text, and images directly on components.
- **Platform Dependence**: AWT components rely heavily on the underlying operating system's GUI libraries, which can lead to inconsistent appearance across platforms.

Swing:

1-Purpose:

- Swing is a more advanced GUI toolkit introduced in Java 2 (JDK 1.2).
- It is built on top of AWT but provides more powerful and flexible components with consistent behavior across different platforms.
- Swing components are entirely written in Java and are not wrappers around native components.

2-Key Features:

Component Classes: Swing includes classes like JButton, JLabel, JTextField, JCheckBox, etc., which are lightweight and customizable.

Advanced Components: Swing offers additional components like JTable, JTree, JScrollPane, etc., for complex data presentation and interaction.

Pluggable Look and Feel: Swing supports pluggable look and feel (LookAndFeel) to change the appearance of components independently of the underlying platform

Double Buffering: Swing components use double buffering for smoother and flicker-free rendering of GUIs.

Event Handling: Swing uses event listener interfaces (ActionListener, MouseListener, etc.) similar to AWT for event handling.

Graphical User Interfaces (GUIs) utilize several key concepts and components to create interactive and user-friendly screens.

Here are some fundamental concepts used in GUI screens:

1-Widgets/Controls:

These are the building blocks of GUIs, such as buttons, text fields/text areas. Widgets allow users to interact with the application by clicking, typing, or selecting options.

- ## Buttons:

Buttons are standard widgets in a GUI. They come with the default Tkinter module and you can place them in your window. A Python function or method can be associated with a button. This function or method is named the callback function.

- ## Text fields/Areas:

The TextField class in Java AWT is a GUI component in the 'java. awt' package. It allows the user to enter a single line of text as an input. It is a simple way to collect text-based information from the user.A TextArea object is a multi-line region that displays text. It can be set to allow editing or to be read-only.

2-Layouts:

. GUIs organize widgets spatially using layouts such as grids, stacks, columns, or rows. Layout managers handle how widgets are arranged and resized on the screen

3-Event Handling:

GUIs respond to user actions (like mouse clicks or keyboard input) through event handling mechanisms. Events trigger actions or updates within the application, such as clicking a button to submit a form.

4-Graphics and Images:

GUIs often incorporate graphical elements such as icons, logos, or images to enhance visual appeal and provide visual cues.

5-Text and Fonts:

Text is crucial for displaying information, labels, instructions, and user input. GUIs support different fonts, sizes, and styles to improve readability and convey meaning.

6-JTable:

JTable is a component in Java Swing that provides a way to display and edit tabular data in a graphical user interface. It is a versatile component commonly used in applications where data needs to be presented in a grid-like format, similar to spreadsheets.

7-Frames:

In the context of graphical user interfaces (GUIs), a "frame" typically refers to a top-level window or container that serves as a primary component for displaying an application's graphical interface. In Java Swing, frames are represented by the JFrame class, which provides essential functionalities for creating, managing, and displaying GUI windows.

8-Panel:

In Java Swing, a JPanel is a lightweight container that is used to group and organize other components (such as buttons, labels, text fields, etc.) within a graphical user interface. Panels are commonly used to structure the layout of a frame or another container, enabling developers to create more complex and organized GUIs.

Inheritance:

Inheritance is a fundamental concept in object-oriented programming (OOP) that is also utilized in GUI programming to promote code reuse, simplify maintenance, and enhance modularity. In the context of GUI screens, inheritance is primarily used in the following ways:

Inheriting from Swing Components:

Java Swing provides a rich hierarchy of classes for GUI components. Inheritance allows developers to create custom components by extending existing Swing classes, such as JFrame, JPanel, JButton, JLabel,

etc. This approach facilitates the creation of specialized components with additional behaviors or custom appearance while leveraging the functionality provided by the base class.

Benefits of Using Inheritance in GUI Screens:

Code Reuse: Inheritance allows developers to reuse existing functionality from base classes, reducing redundant code and improving productivity.

Modularity: GUI components can be structured hierarchically, facilitating easier maintenance and updates.

Customization: Custom components can be tailored to specific application requirements while maintaining consistency and adherence to design patterns.

Polymorphism: Inheritance supports polymorphism, enabling different components to be treated uniformly through their common superclass or interface.

In conclusion, inheritance in GUI programming with Java Swing enhances flexibility, promotes efficient code organization, and supports the creation of customized, reusable components for building complex graphical interfaces.