''बेटी बचाओ, बेटी पढ़ाओ''

# JAYOTI VIDYAPEETH WOMEN'S UNIVERSITY, JAIPUR

## (Format for Preparing E Notes)

## Faculty of CSE

**Faculty Name-** **JV'n Ayushi Choudhary (Assistant Professor/ Associate Professor/ Professor)**

**Program- .....**BCA 5th Semester

**Course Name** – Advanced Java

**Academic Day starts with –**

- Greeting with saying **'Namaste'** by joining Hands together following by 2-3 Minutes Happy session, Celebrating birthday of any student of respective class and **National Anthem**.
- Lecture Starts with-
- Review of previous Session- Define AWT
- Topic to be discussed today : Working with AWT

**Working with AWT (Abstract Window Toolkit)**

Working with AWT (Abstract Window Toolkit) in Java involves creating graphical user interfaces (GUIs) for desktop applications. AWT was one of the earliest GUI libraries in Java and provides a set of classes and methods for building windows, handling events, and rendering graphics. Working with AWT in Java is as follows:

**1. Components and Containers:** AWT defines a hierarchy of components and containers for building GUIs. Components (e.g., buttons, labels) are the basic building blocks, and containers (e.g., Frame, Panel) are used to group and organize components.

**2. Creating Windows:** The **Frame** class is used to create the main application window. It represents a top-level window with a title bar and decorations. One can customize the window's appearance and behavior, such as setting the title, size, and close operation.

**3. Adding Components:** Components can be added like buttons, labels, text fields, and more to the window using container classes like **Panel**. Components are added to containers, which, in turn, are added to the window.

**4. Event Handling:** AWT components generate events in response to user interactions (e.g., button clicks, mouse movements). Event listeners are used to capture and respond to these events.

Event listeners are defined as classes that implement specific listener interfaces, such as **ActionListener** for buttons and **MouseListener** for mouse events.

**5. Layout Management:** AWT provides layout managers like **FlowLayout**, **BorderLayout**, and **GridLayout** to manage the arrangement and sizing of components within containers. These managers automatically adjust component positions based on the container's size.

**6. Graphics and Custom Drawing:** AWT allows custom drawing using the **Graphics** class. You can override the **paint(Graphics g)** method to draw shapes, images, and text directly on components.

**7. Platform Independence:** AWT is designed to be platform-independent, allowing you to create GUI applications that work on different operating systems without modification.

**8. Window Decorations:** AWT windows have standard platform-specific decorations, such as title bars, minimize, maximize, and close buttons. Customizing these decorations is limited.

**9. Threading and Event Dispatching:** AWT applications should be run on the Event Dispatch Thread (EDT) to ensure thread safety and responsiveness. You can use the **EventQueue.invokeLater()** method to post GUI-related tasks to the EDT.

**10. Limitations and Modern Alternatives:** AWT, while functional, has limitations in terms of its component set and look and feel. It has been largely superseded by Swing and JavaFX, which offer more features and a more modern appearance.

In summary, working with AWT in Java involves creating windows, adding components, handling events, managing layouts, and customizing the graphical aspects of your GUI applications. While AWT laid the foundation for Java GUI development, developers often prefer more advanced and modern libraries like Swing or JavaFX for contemporary desktop applications.