EXPERIMENT NO. 11

1. What is GUI programming? How to do GUI programming in JAVA?

Ans: GUI stands for Graphical User Interface. It refers to an interface that allows one to interact with electronic devices like computers and tablets through graphic elements. **GUI programming** involves the use of a number of predefined components such as buttons, checkboxes, text fields, windows, menus, etc. that are part of the class hierarchy.

GUI programming in Java can be done using three sets of Java APIs for graphics programming: AWT(Abstract Windowing Toolkit), Swing and JavaFX.

Abstract Windows Toolkit (AWT). The AWT components are contained within the java.awt package. Java applications using AWT display differently on each platform.

Swing GUI components allow the designer to specify a uniform look and feel for a GUI application across all platforms, or to use each platforms custom look and feel.

Using a Graphical User Interface, when you perform an action, an event is generated. In event-driven programming, the program responds to the events.

The GUI programs that we will create in Java will contain three types of software. These are:

1. **Graphical components** that make up the GUI
2. **Listener classes/methods** that receive the events and respond to them, and
3. **Application methods** that do the work for the user.

2. What is AWT? Why AWT?

Ans: **Java AWT** (Abstract Window Toolkit) is *an API to develop GUI or window-based applications* in java. Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavyweight i.e. its components are using the resources of OS.

The AWT is part of the Java Foundation Classes (JFC) — the standard API for providing a graphical user interface (GUI) for a Java program.

The AWT hides you from the underlying details of the GUI, your application will be running on and thus is at very high level of abstraction. It takes the lowest common denominator approach to retain portability.

Abstract Window Toolkit (AWT) is a set of application program interfaces (API s) used by Java programmers to create graphical user interface (GUI) objects, such as buttons, scroll bars, and windows. AWT is part of the Java Foundation Classes (JFC) from Sun Microsystems, the company that originated Java. The JFC are a comprehensive set of GUI class libraries that make it easier to develop the user interface part of an application program.

3. Differentiate AWT and Swing?

Ans: *Model View controller (MVC)

AWT	Swing
AWT components are heavyweight components	Swing components are lightweight components
AWT doesn't support pluggable look and feel	Swing supports pluggable look and feel
AWT programs are not portable	Swing programs are portable
AWT is old framework for creating GUIs	Swing is new framework for creating GUIs
AWT components require java.awt package	Swing components require javax.swing package
AWT supports limited number of GUI controls	Swing provides advanced GUI controls like Jtable, JTabbedPane etc
More code is needed to implement AWT controls functionality	Less code is needed to implement swing controls functionality
AWT doesn't follow MVC	Swing follows MVC

4. Why swing controls are known as lightweight controls?

Ans:

Swing components depend less on the target platform and use less of the native GUI resource. Hence the **Swing** components that don't rely on native GUI are referred to as **lightweight** components.
Swing is considered lightweight because it is fully implemented in Java, without calling the native operating system for drawing the graphical user interface components.

5. What are the different swing dialog controls are available?

Ans:

In a computer application a dialog is a window which is used to talk to the application. In java Swing toolkit, we can create two kinds of dialogs: standard dialogs and custom dialogs. Standard dialogs are predefined dialogs available in the Swing toolkit, for example the JColorChooser or the JFileChooser. Custom dialogs are created by programmers. They are based on the JDialog class. These are dialogs for common programming tasks like showing text, receiving input, loading files & saving files. There are two types of dialogs: model and modeless. Message dialog is also one of the dialog control of swing.

- Message dialogs
- Custom dialogs
- JFileChooser
- JColorChooser
- Alert dialog

Above are some of the swing dialog controls available.

JDialog is a part Java swing package. The main purpose of the dialog is to add components to it. JDialog can be customized according to user need .

Constructor of the class are:
1.**JDialog()** : creates an empty dialog without any title or any specified owner
2.**JDialog(Frame o)** :creates an empty dialog with a specified frame as its owner
3.**JDialog(Frame o, String s)** : creates an empty dialog with a specified frame as its owner and a specified title
4.**JDialog(Window o)** : creates an empty dialog with a specified window as its owner
5.**JDialog(Window o, String t)** : creates an empty dialog with a specified window as its owner and specified title.
6.**JDialog(Dialog o)** :creates an empty dialog with a specified dialog as its owner
7.**JDialog(Dialog o, String s)** : creates an empty dialog with a specified dialog as its owner and specified title.

6. **Write a only single code for different message dialog control.**

Ans:
```
//This program shows a series of dialog boxes one

//after the other

//Imports are listed in full to show what's being used

//could just import javax.swing.* and java.awt.* etc..

import javax.swing.JFrame;

import javax.swing.JOptionPane;
import javax.swing.UIManager;
import javax.swing.Icon;
import java.awt.EventQueue;
public class SimpleDialogFrame extends JFrame{
//Using a standard Java icon
private Icon optionIcon = UIManager.getIcon("FileView.computerIcon");
//Application start point
public static void main(String[] args) {
//Use the event dispatch thread for Swing components
EventQueue.invokeLater(new Runnable()
{
public void run()
{
//create GUI frame
new SimpleDialogFrame().setVisible(true);
}
});
}
public SimpleDialogFrame()
{
//make sure the program exits when the frame closes
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setTitle("Simple Dialog Box Example");
setSize(500,500);
//This will center the JFrame in the middle of the screen
setLocationRelativeTo(null);
//TO TRY: Comment out the above line and use null for the parent
```

```
//component in one of the JOptionPane calls to see the difference
//it makes to the position of the dialog box.
setVisible(true);
//Use the showMessageDialog method for a plain message dialog box
JOptionPane.showMessageDialog(this, "This is the dialog message"
,"This is the dialog title", JOptionPane.PLAIN_MESSAGE);
//Use the showMessageDialog method for a error message dialog box
JOptionPane.showMessageDialog(this, "This is the dialog message"
,"This is the dialog title", JOptionPane.ERROR_MESSAGE);
//Use the showConfirmDialog method for a warning message dialog box
//with OK, CANCEL buttons. Capture the button number with an int variable
int choice = JOptionPane.showConfirmDialog(this, "This is the dialog message"
,"This is the dialog title", JOptionPane.WARNING_MESSAGE
, JOptionPane.OK_CANCEL_OPTION);
//Use the showConfirmDialog method for an information message dialog box
//with YES, NO, CANCEL buttons. It shows the button choice of previous
//message box
JOptionPane.showConfirmDialog(this,"Last button pressed was number " + choice
, "This is the dialog title", JOptionPane.INFORMATION_MESSAGE
, JOptionPane.YES_NO_CANCEL_OPTION);
//The showOptionDialog method can be made to work as if it were the confirmDialog
//method by using null for the last three parameters. In this case the options for
//the button types (YES, NO, CANCEL) and the message type
(INFORMATION_MESSAGE)
//will be used.
JOptionPane.showOptionDialog(this, "This is the dialog message"
, "This is the dialog title", JOptionPane.YES_NO_CANCEL_OPTION,
JOptionPane.INFORMATION_MESSAGE
,null, null, null);
//Use the showOptionDialog method to make a custom box. If the options parameter
//is null the YES, NO, CANCEL buttons are used. Also notice that even though
//the message type is INFORMATION_MESSAGE the usual icon is overriden by the one
//provided.
JOptionPane.showOptionDialog(this, "This is the dialog message"
, "This is the dialog title", JOptionPane.YES_NO_CANCEL_OPTION,
JOptionPane.INFORMATION_MESSAGE
,optionIcon, null, null);
//String array to be used for the buttons
String[] buttonOptions = new String[] {"Happy Button", "Sad Button", "Confused
Button"};
//If the options parameter is not null the YES, NO, CANCEL buttons are not used
//The buttons are made with the object array - in this case a String array.
JOptionPane.showOptionDialog(this, "This is the dialog message"
, "This is the dialog title", JOptionPane.YES_NO_CANCEL_OPTION,
JOptionPane.INFORMATION_MESSAGE
,optionIcon, buttonOptions, buttonOptions[0]);
}
}
```