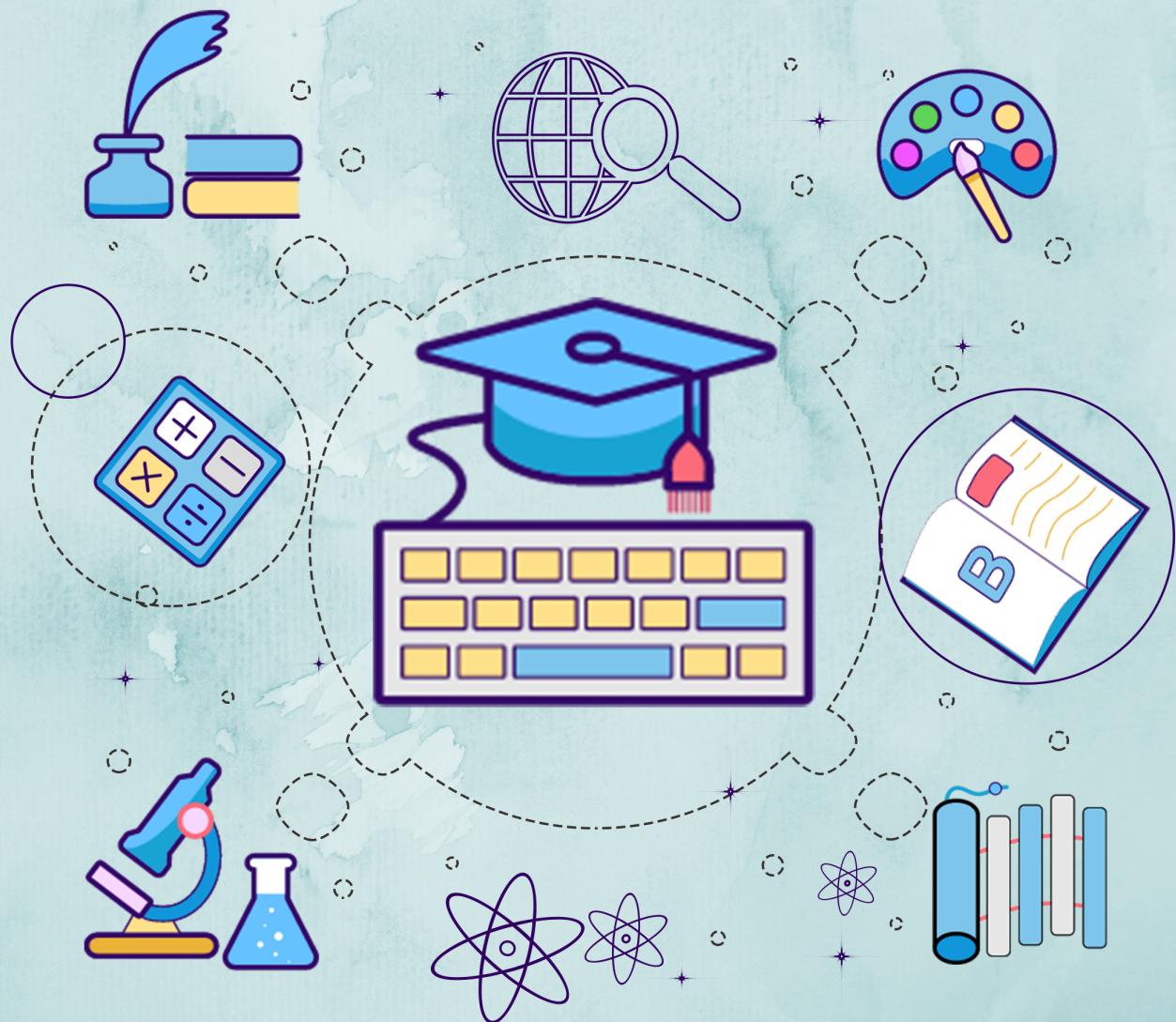


# Kerala Notes



**SYLLABUS | STUDY MATERIALS | TEXTBOOK**

**PDF | SOLVED QUESTION PAPERS**



## KTU STUDY MATERIALS

# OBJECT ORIENTED PROGRAMMING USING JAVA

CST 205

## Module 5

### Related Link :

- KTU S3 STUDY MATERIALS
- KTU S3 NOTES
- KTU S3 SYLLABUS
- KTU S3 TEXTBOOK PDF
- KTU S3 PREVIOUS YEAR  
SOLVED QUESTION PAPER

# MODULE 5

## Graphical User Interface & Database support of Java

### CHAPTER 1

#### SWING

Prepared By Mr. EBIN PM, AP, IESCE

1

#### SWING FUNDAMENTALS

- Java Swing is a **GUI Framework** that contains a set of classes to provide more powerful and flexible GUI components than AWT.
- Swing provides the look and feel of modern Java GUI.
- Swing library is an official Java GUI tool kit released by Sun Microsystems.
- It is used to create graphical user interface with Java.
- Swing classes are defined in **javax.swing** package and its sub-packages.
- Java Swing provides **platform-independent** and **lightweight** components.

Prepared By Mr.EBIN PM, AP, IESCE

EDULINE

2

**Java Swing** is a part of Java Foundation Classes (JFC) that is *used to create window-based applications*.

It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java

## JFC

- The Java Foundation Classes (JFC) are a set of GUI components which simplify the development of desktop applications.
- The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

## ❖ Features of Swing

### Platform Independent:

- It is platform independent, the swing components that are used to build the program are not platform specific.
- It can be used at **any platform** and **anywhere**.

### Lightweight:

- Swing components are lightweight which helps in creating the UI lighter.
- Swings component allows it to plug into the operating system user interface framework that includes the mappings for screens or device and other user interactions like key press and mouse movements.

### Plugging:

- It has a powerful component that can be extended to provide the support for the user interface that helps in good look and feel to the application.
- It refers to the highly **modular-based architecture** that allows it to plug into other customized implementations and framework for user interfaces.

**Manageable:** It is easy to manage and configure. Its mechanism and composition pattern allows changing the settings at run time as well. The uniform changes can be provided to the user interface without doing any changes to application code.

### MVC:

- They mainly follows the concept of MVC that is **Model View Controller**.
- With the help of this, we can do the changes in one component without impacting or touching other components.
- It is known as loosely coupled architecture as well.

### Customizable:

- Swing controls can be easily customized. It can be changed and the visual appearance of the swing component application is independent of its internal representation.

### Rich Controls :

- Swing provides a rich set of advanced controls like Tree, TabbedPane, slider, colorpicker, and table controls.

## ❖Difference between AWT and Swing

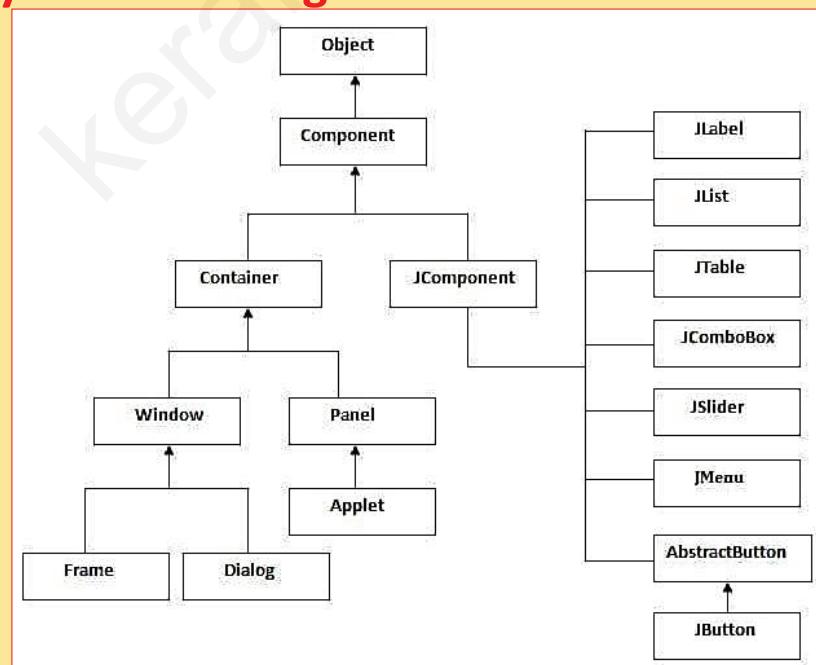
No.	Java AWT	Java Swing
1)	AWT components are <b>platform-dependent</b> .	Java swing components are <b>platform-independent</b> .
2)	AWT components are <b>heavyweight</b> .	Swing components are <b>lightweight</b> .
3)	AWT <b>doesn't support pluggable look and feel</b> .	Swing <b>supports pluggable look and feel</b> .
4)	AWT provides <b>less components</b> than Swing.	Swing provides <b>more powerful components</b> such as tables, lists, scrollpanes, colorchooser, tabbedpane etc.
5)	AWT <b>doesn't follows MVC</b> (Model View Controller) where model represents data, view represents presentation and controller acts as an interface between model and view.	Swing <b>follows MVC</b> .

Prepared By Mr.EBIN PM, AP, IESCE

**EDULINE**

7

## ❖Hierarchy of Java Swing classes



Prepared By Mr.EBIN PM, AP, IESCE

**EDULINE**

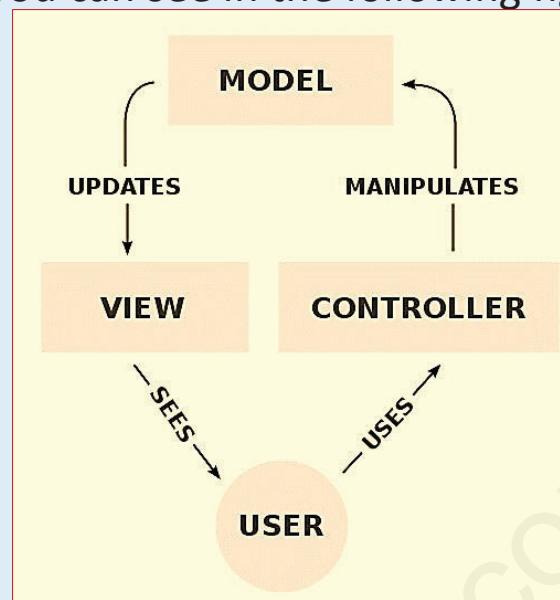
8

## The Model-View-Controller Architecture

- Swing uses the model-view-controller architecture (MVC) as the fundamental design behind each of its components
- Essentially, MVC breaks GUI components into three elements. Each of these elements plays a crucial role in how the component behaves.
- The Model-View-Controller is a well known **software architectural pattern** ideal to implement user interfaces on computers by dividing an application into three interconnected parts

- **Main goal** of Model-View-Controller, also known as MVC, is to separate internal representations of an application from the ways information are presented to the user.
- Initially, MVC was designed for desktop GUI applications but it's quickly become an extremely popular pattern for designing web applications too.
- MVC pattern has the **three components** :
  - Model** that manages data, logic and rules of the application
  - View** that is used to present data to user
  - Controller** that accepts input from the user and converts it to commands for the Model or View.

- the MVC pattern defines the interactions between these three components like you can see in the following figure :



- The Model receives commands and data from the Controller. It stores these data and updates the View.
- The View lets to present data provided by the Model to the user.
- The Controller accepts inputs from the user and converts it to commands for the Model or the View.

## COMPONENTS & CONTAINERS

- A **component** is an independent visual control, such as a push button or slider.
- A **container** holds a group of components. Thus, a container is a special type of component that is designed to hold other components.
- Swing components inherit from the **javax.Swing.JComponent** class, which is the root of the Swing component hierarchy.

### ❖ COMPONENTS

- Swing components are derived from the **JComponent** class.
- JComponent provides the functionality that is common to all components. For example, JComponent supports the pluggable look and feel.
- JComponent inherits the AWT classes Container and Component. Thus, a Swing component is built on and compatible with an AWT component.
- All of Swing's components are represented by classes defined within the package **javax.swing**.
- The following table shows the class names for Swing components

- JApplet
- JColorChooser
- JDialog
- JFrame
- JLayeredPane
- JMenuItem
- JPopupMenu
- JRootPane
- JSlider
- JTable
- JTogglegbutton
- JViewport
- JButton
- JComboBox
- JEditorPane
- JInternalFrame
- JList
- JOptionPane
- JProgressBar

Prepared By Mr.EBIN PM, AP, IESCE

EDULINE

15

- Notice that all component classes begin with the letter J.
- For example, the class for a label is JLabel; the class for a push button is JButton; and the class for a scroll bar is JScrollPane

## ❖CONTAINERS

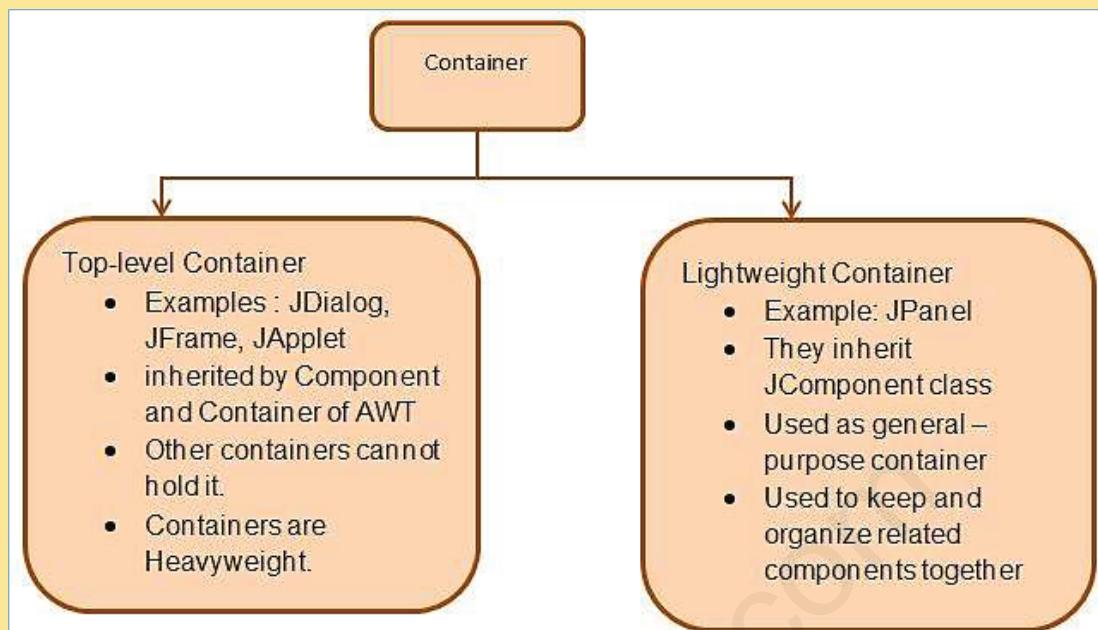
- Swing defines two types of containers. The first are top-level containers: **JFrame**, **JApplet**, **JWindow**, and **JDialog**. These containers do not inherit **JComponent**. They inherit the AWT classes **Component** and **Container**.
- The second type container are lightweight and the top-level containers are heavyweight. This makes the top-level containers a special case in the Swing component library.

Prepared By Mr.EBIN PM, AP, IESCE

EDULINE

16

In Java, Containers are divided into two types as shown below:



Prepared By Mr.EBIN PM, AP, IESCE

**EDULINE**

17

Following is the list of commonly used containers while designed GUI using SWING.

Sr.No.	Container & Description
1	<b>Panel</b> ↗ JPanel is the simplest container. It provides space in which any other component can be placed, including other panels.
2	<b>Frame</b> ↗ A JFrame is a top-level window with a title and a border.
3	<b>Window</b> ↗ A JWindow object is a top-level window with no borders and no menubar.

Prepared By Mr.EBIN PM, AP, IESCE

**EDULINE**

18

## Swing Example : A window on the screen.

```

import javax.swing.JFrame;
import javax.swing.SwingUtilities;

public class Example extends JFrame {

    public Example() {
        setTitle("Simple example");
        setSize(300, 200);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    public static void main(String[] args) {
        Example ex = new Example();
        ex.setVisible(true);
    }
}

```

### Output



Prepared By Mr.EBIN PM, AP, IESCE

**EDULINE**

19

## EVENT HANDLING IN SWINGS

- The functionality of Event Handling is what is the further step if an action performed.
- Java foundation introduced “Delegation Event Model” i.e describes how to generate and control the events.
- The key elements of the Delegation Event Model are as **source** and **listeners**.
- The listener should have registered on source for the purpose of alert notifications.
- All GUI applications are event-driven

Prepared By Mr.EBIN PM, AP, IESCE

**EDULINE**

20

## ❖ Java Swing event object

- When something happens in the application, an **event object** is created.
- For example, when we click on the button or select an item from a list.
- There are several types of events, including ActionEvent, TextEvent, FocusEvent, and ComponentEvent.
- Each of them is created under specific conditions.
- An event object holds information about an event that has occurred.

## SWING LAYOUT MANAGERS

- Layout refers to the arrangement of components within the container.
- Layout is placing the components at a particular position within the container. The task of laying out the controls is done automatically by the Layout Manager.
- The **layout manager** automatically positions all the components **within the container**.
- Even if you do not use the layout manager, the components are still positioned by the default layout manager. It is possible to lay out the controls by hand, however, it becomes very difficult

- Java provides various layout managers to position the controls. Properties like size, shape, and arrangement varies from one layout manager to the other.
- There are following classes that represents the **layout managers**:
  - java.awt.BorderLayout
  - java.awt.FlowLayout
  - java.awt.GridLayout
  - java.awt.CardLayout
  - java.awt.GridBagLayout
  - javax.swing.BoxLayout
  - javax.swing.GroupLayout
  - javax.swing.ScrollPaneLayout
  - javax.swing.SpringLayout etc.

Prepared By Mr.EBIN PM, AP, IESCE

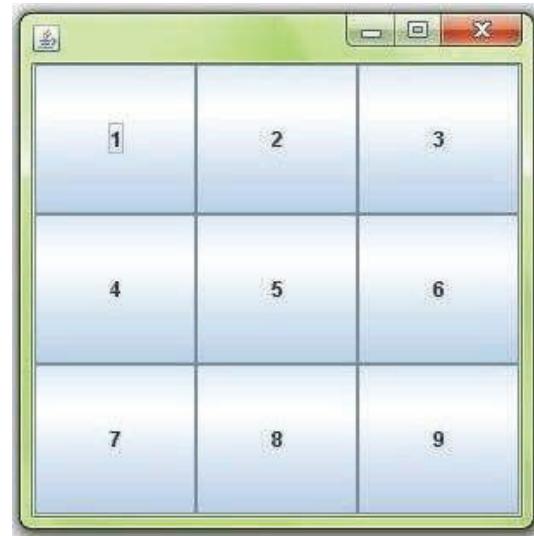
**EDULINE**

23

**BorderLayout**

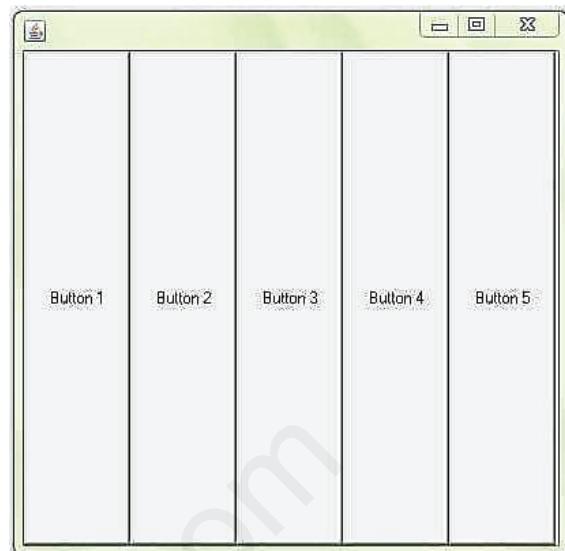


**GridLayout**



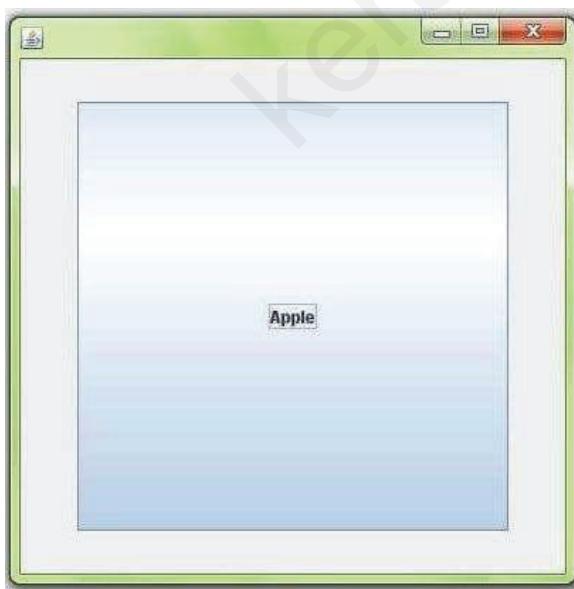
Prepared By Mr.EBIN PM, AP, IESCE

24

**FlowLayout****BoxLayout**

Prepared By Mr.EBIN PM, AP, IESCE

25

**CardLayout****GroupLayout**

Prepared By Mr.EBIN PM, AP, IESCE

26

## Example of JButton

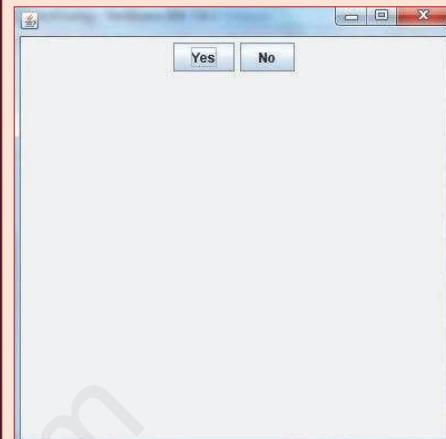
```

import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
public class testswing extends JFrame
{

    testswing()
    {
        JButton bt1 = new JButton("Yes");           //Creating a Yes Button.
        JButton bt2 = new JButton("No");            //Creating a No Button.
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE) //setting close operation.
       .setLayout(new FlowLayout());               //setting layout using FlowLayout object
        setSize(400, 400);                      //setting size of Jframe
        add(bt1);                                //adding Yes button to frame.
        add(bt2);                                //adding No button to frame.

        setVisible(true);
    }
    public static void main(String[] args)
    {
        new testswing();
    }
}

```



Prepared By Mr.EBIN PM, AP, IESCE

EDULINE

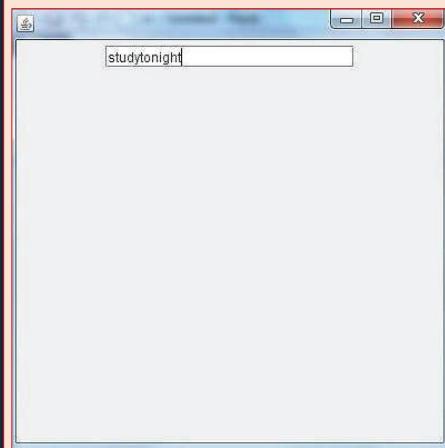
27

## Example of JTextField

```

import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
public class MyTextField extends JFrame
{
    public MyTextField()
    {
        JTextField jtf = new JTextField(20); //creating JTextField.
        add(jtf);                         //adding JTextField to frame.
        setLayout(new FlowLayout());
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400, 400);
        setVisible(true);
    }
    public static void main(String[] args)
    {
        new MyTextField();
    }
}

```



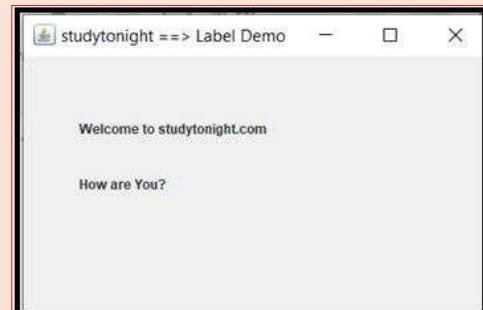
Prepared By Mr.EBIN PM, AP, IESCE

EDULINE

28

## Example of JLabel - It is used for placing text in a box

```
import javax.swing.*;
class SLabelDemo1
{
    public static void main(String args[])
    {
        JFrame label_f= new JFrame("studytonight ==> Label Demo");
        JLabel label_11,label_12;
        label_11=new JLabel("Welcome to studytonight.com");
        label_11.setBounds(50,50, 200,30);
        label_12=new JLabel("How are You?");
        label_12.setBounds(50,100, 200,30);
        label_f.add(label_11);
        label_f.add(label_12);
        label_f.setSize(300,300);
        label_f.setLayout(null);
        label_f.setVisible(true);
    }
}
```



Prepared By Mr.EBIN PM, AP, IESCE

EDULINE

29

# MODULE 5

## CHAPTER 2

### JDBC



Prepared By Mr. EBIN PM, AP, IESCE

1

### Java DataBase Connectivity (JDBC)

- JDBC stands for Java Database Connectivity, which is a standard **Java API** for database-independent connectivity between the Java programming language and a wide range of databases.
- The JDBC library includes APIs for each of the tasks mentioned below that are commonly associated with database usage.
- Making a connection to a database
- Creating SQL or MySQL statements
- Executing SQL or MySQL queries in the database
- Viewing & Modifying the resulting records

## ❖ JDBC Architecture

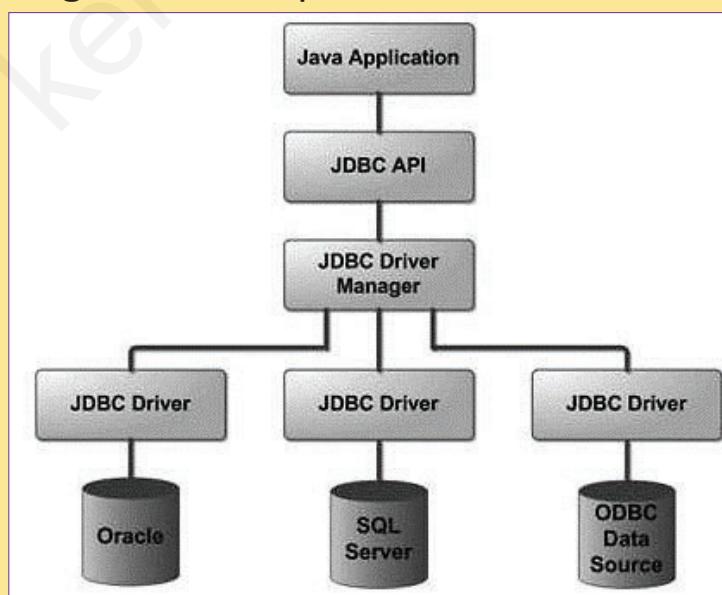
- JDBC Architecture consists of **two layers**
- **JDBC API:** This provides the application-to-JDBC Manager connection.
- **JDBC Driver API:** This supports the JDBC Manager-to-Driver Connection.
- The JDBC API uses a **driver manager** and **database-specific drivers** to provide transparent connectivity to heterogeneous databases.
- The JDBC driver manager ensures that the correct driver is used to access each data source.

Prepared By Mr.EBIN PM, AP, IESCE

**EDULINE**

3

- Following is the **architectural diagram**, which shows the location of the driver manager with respect to the JDBC drivers and the Java application



Prepared By Mr.EBIN PM, AP, IESCE

**EDULINE**

4

## Java Database Connectivity with 5 Steps

➤ There are 5 steps to connect any java application with the database using JDBC. These steps are as follows:

1. Register the Driver class
2. Create connection
3. Create statement
4. Execute queries
5. Close connection

- The `forName()` method is used to register the driver class.
- The `getConnection()` method of DriverManager class is used to establish connection with the database
- The `createStatement()` method of Connection interface is used to create statement. The object of statement is responsible to execute queries with the database.
- The `executeQuery()` method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of a table.
- By closing connection object statement and ResultSet will be closed automatically. The `close()` method of Connection interface is used to close the connection.

## Java Database Connectivity with MySQL

- To connect Java application with the MySQL database, we need to follow 5 following steps.
- In this example we are using MySQL as the database. So we need to know following informations for the mysql database:
  - 1. Driver class:** The driver class for the mysql database is com.mysql.jdbc.Driver.
  - 2. Connection URL:** The connection URL for the mysql database is jdbc:mysql://localhost:3306/sonoo where jdbc is the API, mysql is the database, localhost is the server name on which mysql is running, we may also use IP address, 3306 is the port number and sonoo is the database name. We may use any database, in such case, we need to replace the sonoo with our database name.

Prepared By Mr.EBIN PM, AP, IESCE

EDULINE

7

**3. Username:** The default username for the mysql database is root.

**4. Password:** It is the password given by the user at the time of installing the mysql database. In this example, we are going to use root as the password.

➤ Let's first create a table in the mysql database, but before creating table, we need to create database first.

```
create database sonoo;
```

```
use sonoo;
```

```
create table emp(id int(10),name varchar(40),age int(3));
```

Prepared By Mr.EBIN PM, AP, IESCE

EDULINE

8

## ❖ Example to Connect Java Application with mysql database

```

import java.sql.*;
class MySqlCon{
public static void main(String args[]){
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/sonoo","root","root");
//here sonoo is database name, root is username and password
Statement stmt=con.createStatement();
ResultSet rs=stmt.executeQuery("select * from emp");
while(rs.next())
System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));
con.close();
}catch(Exception e){ System.out.println(e);}
}
}

```

This example will fetch all the records of emp table.

Prepared By Mr.EBIN PM, AP, IESCE

**EDULINE**

9

## Creating a sample MySQL database

```

create database SampleDB;
use SampleDB;
CREATE TABLE `users` (
  `user_id` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(45) NOT NULL,
  `password` varchar(45) NOT NULL,
  `fullname` varchar(45) NOT NULL,
  `email` varchar(45) NOT NULL,
  PRIMARY KEY (`user_id`)
);

```

Prepared By Mr.EBIN PM, AP, IESCE

**EDULINE**

10

## Connecting to the database

```
String dbURL = "jdbc:mysql://localhost:3306/sampledb";
String username = "root";
String password = "secret";
try {
    Connection conn = DriverManager.getConnection(dbURL, username,
password);
    if (conn != null) {
        System.out.println("Connected");
    }
} catch (SQLException ex)
{
    ex.printStackTrace();
}
```

Prepared By Mr.EBIN PM, AP, IESCE

EDULINE

11

- Once the connection was established, we have a **Connection** object which can be used to create statements in order to execute SQL queries.
- In the above code, we have to close the connection explicitly after finish working with the database:

**conn.close();**

### ❖**INSERT Statement Example**

Let's write code to insert a new record into the table **Users** with following details:

username: bill

password: secretpass

fullname: Bill Gates

email: bill.gates@microsoft.com

Prepared By Mr.EBIN PM, AP, IESCE

EDULINE

12

```

String sql = "INSERT INTO Users (username, password, fullname, email) VALUES (?, ?, ?, ?)";

PreparedStatement statement = conn.prepareStatement(sql);
statement.setString(1, "bill");
statement.setString(2, "secretpass");
statement.setString(3, "Bill Gates");
statement.setString(4, "bill.gates@microsoft.com");
int rowsInserted = statement.executeUpdate();
if (rowsInserted > 0) {
    System.out.println("A new user was inserted successfully!");
}

```

Prepared By Mr.EBIN PM, AP, IESCE

**EDULINE**

13

## ❖SELECT Statement Example

```

String sql = "SELECT * FROM Users";

Statement statement = conn.createStatement();
ResultSet result = statement.executeQuery(sql);

int count = 0;

while (result.next()){
    String name = result.getString(2);
    String pass = result.getString(3);
    String fullname = result.getString("fullname");
    String email = result.getString("email");

    String output = "User #%-d: %s - %s - %s - %s";
    System.out.println(String.format(output, ++count, name, pass, fullname, email));
}

```

## Output

User #1: bill - secretpass - Bill Gates - bill.gates@microsoft.com

Prepared By Mr.EBIN PM, AP, IESCE

**EDULINE**

14

## ❖ UPDATE Statement Example

```
String sql = "UPDATE Users SET password=?, fullname=?, email=? WHERE username=?";  
  
PreparedStatement statement = conn.prepareStatement(sql);  
statement.setString(1, "123456789");  
statement.setString(2, "William Henry Bill Gates");  
statement.setString(3, "bill.gates@microsoft.com");  
statement.setString(4, "bill");  
  
int rowsUpdated = statement.executeUpdate();  
if (rowsUpdated > 0) {  
    System.out.println("An existing user was updated successfully!");  
}
```

## ❖ DELETE Statement Example

- The following code snippet will delete a record whose username field contains “bill”

```
String sql = "DELETE FROM Users WHERE username=?";  
  
PreparedStatement statement = conn.prepareStatement(sql);  
statement.setString(1, "bill");  
  
int rowsDeleted = statement.executeUpdate();  
if (rowsDeleted > 0) {  
    System.out.println("A user was deleted successfully!");  
}
```