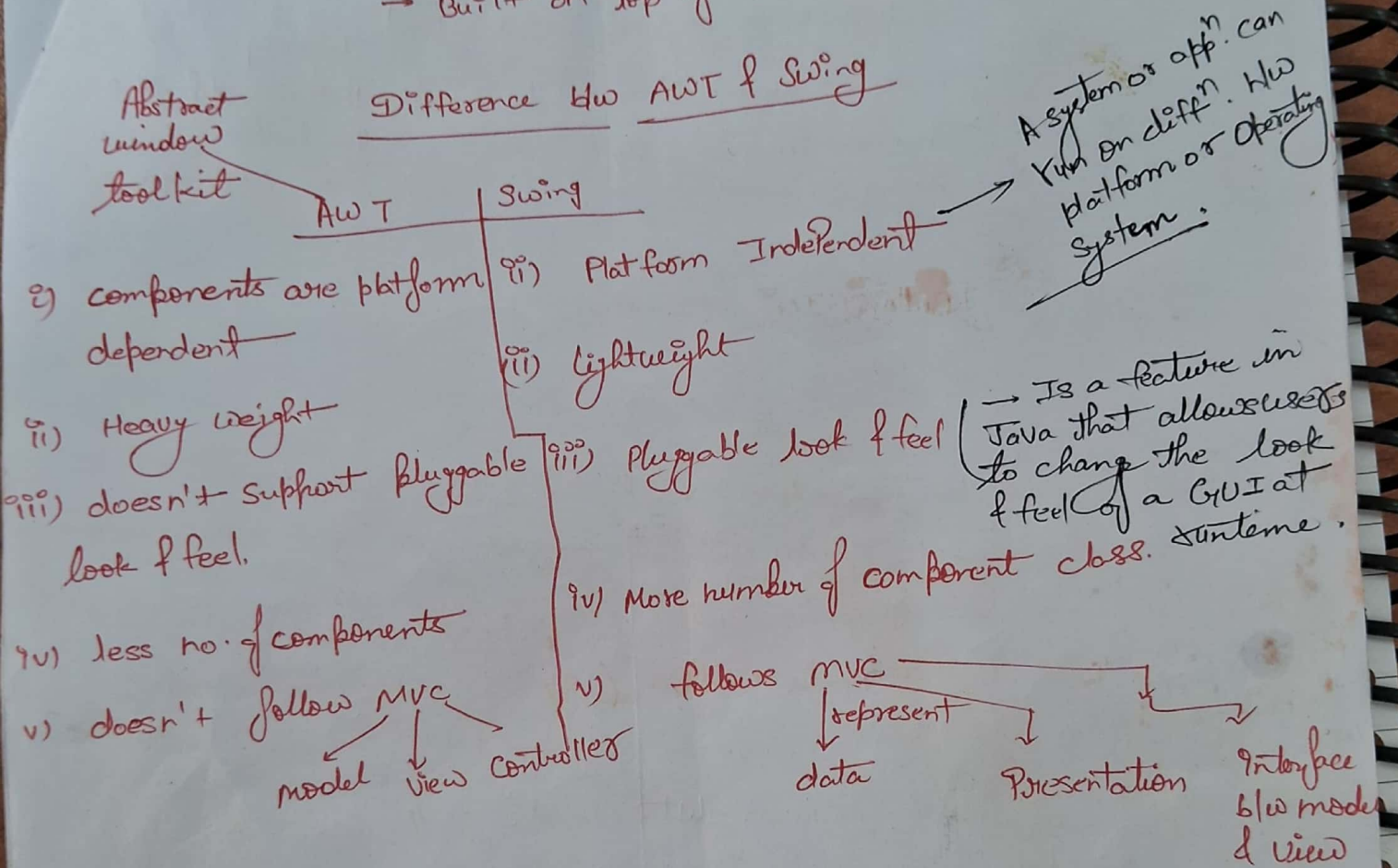


Unit - 3

Java Swing: → {window based appⁿ}

- Part of Java foundation classes (JFC)
- Built on top of AWT API.

Difference b/w AWT & Swing



Import method of component class;


Void add (component) || Add a component to container.

Void set Size (int width, int height)
↳ set size of window

Void set layout (layout manager m)
↳ flow, layout
↳ Grid, layout


Java Swing classes

(1) \rightarrow J Button class : \rightarrow used to create a button.

J Button () \rightarrow No text Icon 

J Button (String s) \rightarrow
 \rightarrow Then we have a button
with String s.

String \rightarrow that will appear on
the button. For e.g
s on अदर पास किता
click /

J Button (Icon i) \rightarrow 

(2) J Radio Button class \rightarrow J Radio Button it use to

\rightarrow J Radio Button () \rightarrow choose one / single from multiple
 \rightarrow No text option.

\rightarrow J Radio Button (String s) \rightarrow text

\rightarrow J Radio Button (String s, boolean Select)
 \rightarrow text selected status.

Use to import
the built-in
user-define
package

For e.g. import java.awt.event.*; // Event handling
import javax.swing.*; // Swing Classes

public class Swing

{
Swing()

}

JFrame f = new JFrame();

JButton b1 = new JButton ("click");

b1.set Bounds (100, 150, 130, 48);
* y width height

f.add(b1);

Button Group \leftarrow

J Radio Button
" "

J1 = new JRadio Button ("A");
J2 = " " " " ("B");

J1.set Bounds (---); J2.set Bounds (---);

Button group bg = new Button Group();
bg.add(J1); bg.add(J2);

(3) J Text Area class \rightarrow multiline area

Text area() // Blank Text Area

J Text Area (storing s)

J Text Area (int row, int col)

```

    JTextArea jT = new JTextArea(100, 100);
    f.add(jT);
    f.setLayout(new BorderLayout());
    f.setVisible(true);

```

Jcombo Box class : \rightarrow

Combo Box class : \rightarrow

\rightarrow used to create combo box (drop-down list)
be selected. 0-100 numbers

- Only one item can be selected. → There are 3 types of constructors.

(1) \rightarrow Combo Box ()

(1) L Jcombo Box () Jcombo Box with items array.
(2) L JcomboBox (object [] items)

(2) L JComboBox (vector <?> items) as vector array

Table class → used to display data on two dimensional table of cells)

L J Table() Empty cells

J Table (object [][] rows, object [] col)
↓
specified data

JColorChooser class → create color chooser dialog box.

J color chooser () { color show Dialog () }

Jcolorchooser (color Initial color)

E.g. → `color icolor = color.Black;`
`color c = JColorChooser.showDialog`
`(this, "select color", icolor);`
`if. setBackground(c);`

How to display Image in Swing :-

Use method `drawImage()` of `Graphics` class
 Object of `Image` class. → ^{co-ordinates} x, y

Syntax :-

`Boolean drawImage (Image img, int x, int y, Image`

`observer, observer)`

object of `ImageObserver` interface

E.g. → `Import java. awt;`
`Import javax.swing. JFrame;`
`Public class Image extends Canvas`

{

`public void paint (Graphics g)`

{ `Toolkit tk = Toolkit.getDefaultToolkit();`

`Image i = tk.getImage ("ai.jpg")`

`g.drawImage (i, 100, 100, this);`

}

}

Interfaces → has method for handling notification of state of Image loading.

can be used to redisplay.

- Java Swing: → Swing is the extension of the ^{abstract window toolkit} AWT which offers much improved functionality over AWT.
- It is part of Oracle's Java foundation classes.
 - Java Swing provides platform-independent & light weight components.
 - Swing is entirely written in Java.
 - Java Swing key feature of Swing supports:
 - * A pluggable look & feel (changes to GUI at runtime)
 - * lightweight (uses less no. of resources) → fast execution.
 - * AWT is base of Swing.
 - * Swing follows MVC (Model, View, Controller).

MVC Architecture

Feature of Swing class: →

Java FX implem of Swing → Java FX is the newer standard with a smaller library more consistent updates & consistent MVC support.

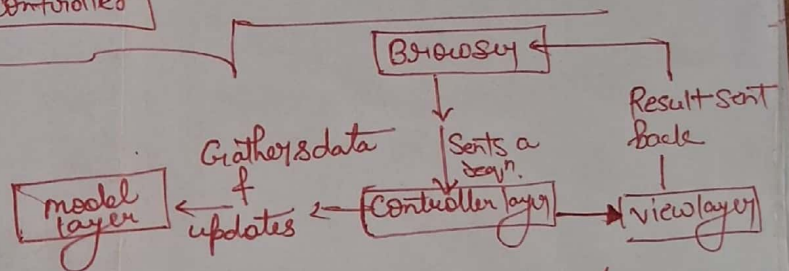
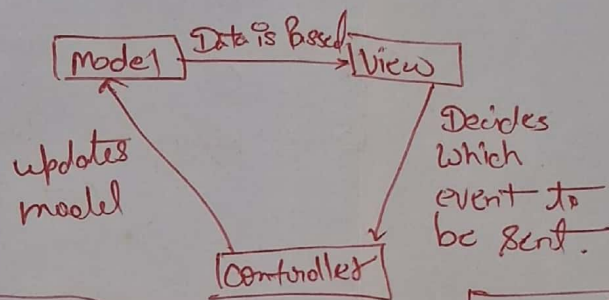
The model view controller (MVC) is a well known design pattern in the web development field.

- It is way to organise our code.
- It consist of 3-layers → ① model ② view ③ Controller

① Model: → state info. associated with component
↳ checkbox on information → out → checked & unchecked
↳ It represent business layer of applⁿ.

② View → How the component is displayed on screen.
↳ Visualization, presentation.

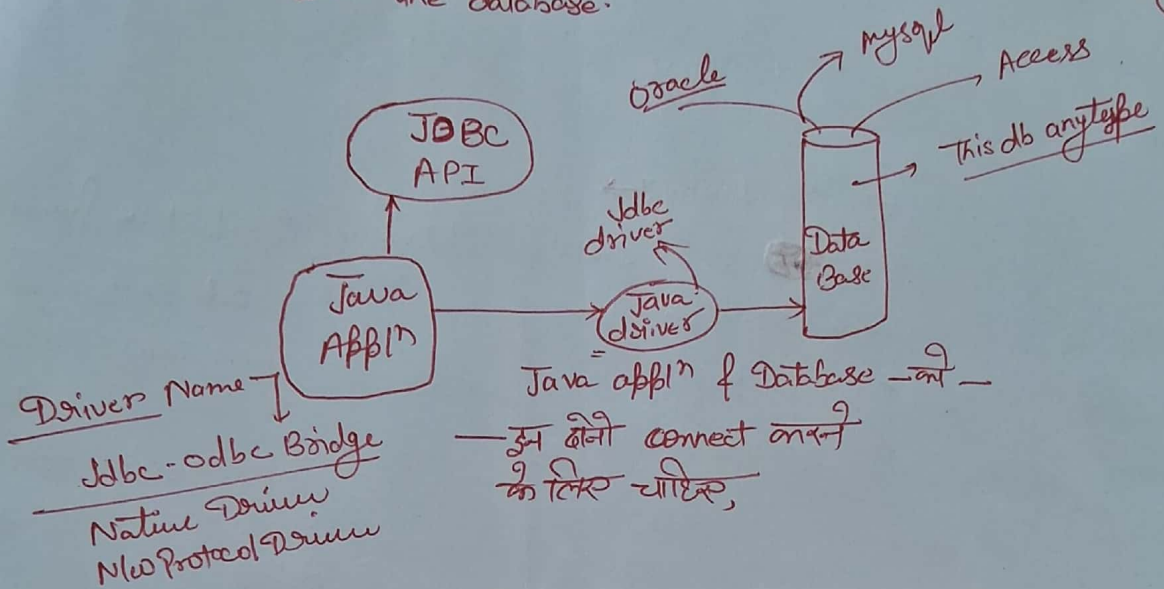
③ Controller → How the component react to user.
↳ checkbox → checked → (Hello) → display on screen
↳ (unchecked)



View → It represent the presentation layer of applⁿ.
It is used to visualize the data that the model contain.

Java JDBC lectures: → (Java database connectivity)

Java JDBC is a Java API to connect & execute query with the database.



Steps to connect to database in Java: →

- ① Registered Driver class: →
↳ for Name() method.
e.g. → Oracle, Driver Registration करना है, या load करना है।

Driver class की registration के लिए method use होता है for Name() method.

e.g. → class.forName("oracle.jdbc.driver.OracleDriver");

- ② create connection object: →

जिस class में throw होता है
Driver manager

get connection con =
class object

Driver manager.get connection
("Path", "Username", "Pwd");

mysql → [Emp / root] → Pool → Blank
↓
Path user name

(3) create statement object :-

↳ Exe. Queries with db.

Statement stat = (con.createStatement());
 class object

(4) Exe the Query :-

Result rs = stat.executeQuery("Select * from Emp");

while (rs.next())

rs.getInt();

rs.getString();

SQL statement ko
 den krke, kisi db
 me aur? Emp Table pe
 kisi result dega usko
 store krke dega rs
 object se ✓

(5) close the connection object ->

con.close();

Reg. of Driver class
 creating conn
 statement
 Execute Query
 closing conn

How to connect to Oracle Database :-

Driver class = oracle.jdbc.driver.OracleDriver

connection url = jdbc:oracle:thin@localhost:

1521:xe

Import java.sql.*; // Package

try

class for Name (M);

connection con = DriverManager.getConnection(u);

stmt

Oracle Service Name

sid	Service Name
1	A
2	B


```

Result set rs = stmt.executeQuery(Select * from
Emp);
While (rs.next())
{
    S.O.P(rs.getInt(1) + rs.getString(2));
    ↓
    System.out.println
    }
    con.close();
}

```

connecte

How to connect with mysql Database :-

Driver class = com.mysql.jdbc.Driver

Connection url = jdbc:mysql://localhost:3306/ Emp

Username = root

Pwd = —

url

Name of db

How to connect with Access Database :-

① With DSN → Data Source Name

dsn₁ → Emp]db

String url = "jdbc:odbc:dsn₁";

Class.forName("Sun.jdbc.odbc.JdbcOdbcDriver");

Connection c = DriverManager.getConnection(url);

stmt

Ex of statement Interface: \rightarrow

Provides method to execute queries with the database

Methods :-

- (1) Execute Query (String sql) → Result Set
- (2) Execute update (String sql) → boolean
- (3) Execute (String sql) → Insert, update
- (4) Execute Batch() → Insert 1
Insert 2

```
import java.sql.*; // Package
class sql
```

```

    }
    public static void main (String args[]) throws Exception
    {
        Class.forName (" oracle.jdbc.driver
                        oracle Driver ");
    }
}

```

connection con = DriverManager.getConnection
();

Statement stmt = con. create statement();
 String sql = "select * from emp";
 Result rs = stmt. executeQuery (sql);

int i = stmt.
execute update ←
("delete from emp
where id = 4") ;

```

while (rs.next())
{
    System.out.println(rs.getInt(1));
    //    //    //    (rs.getString(2));
}

rs.close();
con.close();
}

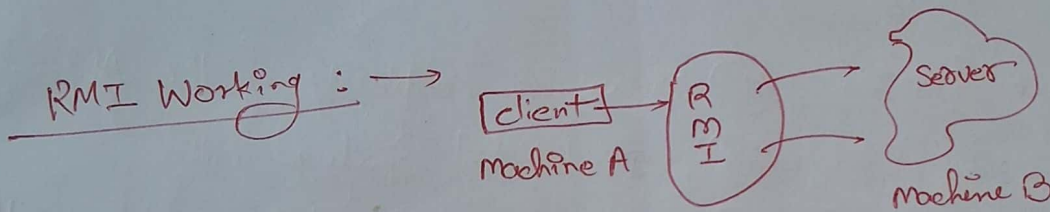
```


RMI

Java RMI : → (Java Remote Method Invocation.)

↳ It is a API that allows an object to invoke another method that exists in separate address space.

local m/c → ये यहाँ Process run कर रहे हैं।
server m/c → ये कोइ ठीक Program run कर रहे हैं।

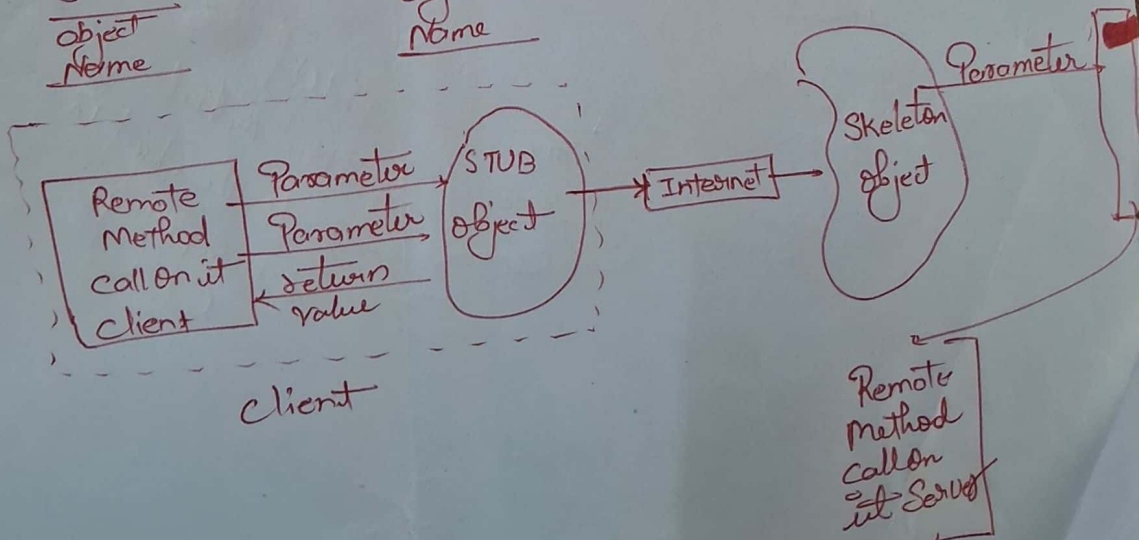


→ Machine A can invoke method of machine B.

→ Two intermediate objects are used for the communication between client and server

“STUB”
object
Name

“SKELETON”
object
Name



STUB: → It creates Info. block ^{Sends} → Server

- ↳ ID of remote object
- ↳ Method Name to Invoke
- ↳ Parameter to Pass.

SKELETON: →

- ↳ Calls Desired Method.
- ↳ Forwards the Parameter
- ↳ return the value to STUB object.

Goals of RMI: →

- (1) Minimizes complexity of the Appⁿ.
- (2) Distributed Garbage collⁿ.
- (3) Minimizes diff. b/w working with local & remote objects.

RMI Registry: → It is a namespace on which all server object are placed.

