1. **Simple hello world program to understand basics**

```
import javax.swing.*;
import java.awt.*;

public class FirstGUI{
    public static void main(String[] args) {
        Abc obj = new Abc();
    }
}

class Abc extends JFrame{
    public Abc(){
        JLabel l = new JLabel("Hello World");
        JLabel l1 = new JLabel("Hello Meet");
        add(l);
        add(l1);

        setLayout(new FlowLayout());
        setVisible(true);
        setSize(400, 400);
        setDefaultCloseOperation(3); //3 or
JFrame.EXIT_ON_CLOSE
    }
}
```

1. **setLayout(new FlowLayout());**

This line sets the layout manager for the current **JFrame** (or **JWindow**, **JDialog**, etc.) to a **FlowLayout**. A layout manager is responsible for arranging the components (such as buttons, labels, text fields, etc.) within a container.

2. `setVisible(true);`

This line makes the `JFrame` (or other window) visible on the screen. When a `JFrame` is created, it's initially invisible, so this line is necessary to display the window to the user.

3. `setDefaultCloseOperation(3); //3 or JFrame.EXIT_ON_CLOSE`

This line sets the default close operation for the `JFrame`. The default close operation determines what happens when the user closes the window (e.g., by clicking the "X" button in the top-right corner).

In this case, the value `3` is equivalent to `JFrame.EXIT_ON_CLOSE`, which means that the application will exit (i.e., terminate) when the window is closed. This is a common behavior for many GUI applications.

2. Learn about Calculator concept with ActionPerformed with some basic styles

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Calc {
    public static void main(String args[]){
        Calculator obj = new Calculator();
    }
}

class Calculator extends JFrame implements ActionListener{
    JTextField t1, t2;
    JButton b1, b2;
    JLabel l;

    public Calculator(){
        t1 = new JTextField(10);
        t2 = new JTextField(10);
        b1 = new JButton("Addition");
        b2 = new JButton("Multiply");
        l = new JLabel();

        t1.setFont(new Font("Arial", Font.BOLD, 18));
        t2.setFont(new Font("Arial", Font.BOLD, 18));
        b1.setFont(new Font("Arial", Font.BOLD, 18));
        b2.setFont(new Font("Arial", Font.BOLD, 18));
        l.setFont(new Font("Arial", Font.BOLD, 18));
```

```
        t1.setForeground(Color.BLUE);
        t2.setForeground(Color.BLUE);
        b1.setForeground(Color.RED);
        b2.setForeground(Color.RED);
        l.setForeground(Color.black);

        add(t1);
        add(t2);
        add(b1);
        add(b2);
        add(l);

        b1.addActionListener(this);
        b2.addActionListener(this);

        setLayout(new FlowLayout());
        setVisible(true);
        setSize(400, 400);
        setDefaultCloseOperation(3);
    }

    public void actionPerformed(ActionEvent ae){
        try {
            int a = Integer.parseInt(t1.getText());
        int b = Integer.parseInt(t2.getText());
        if(ae.getSource() == b1){
            int sum = a + b;
            l.setText("Addition is : " + sum);
        }
        if(ae.getSource() == b2){
            int mul = a * b;
            l.setText("Multiply is : " + mul);
        }
        } catch (Exception e) {
            l.setText("Input must be Integer");
        }
    }
}
```

3. JDBC for database connection (Data take from user)

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Scanner;
```

```
import javax.swing.JOptionPane;

public class DB_connection {
    public static void main(String args[]){
        try {
            // Load the MySQL JDBC Driver
            Class.forName("com.mysql.cj.jdbc.Driver");

            // Establish a connection to the database
            Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc", "root",
"Meet@123");

            Scanner sc = new Scanner(System.in);

            System.out.print("Enter id for student : ");
            int id = sc.nextInt();
            sc.nextLine();

            System.out.print("Enter name for student : ");
            String name = sc.nextLine();

            System.out.print("Enter mobile no for student : ");
            int phone = sc.nextInt();

            // SQL query to insert data into the database
            String sql = "INSERT INTO crud_operation VALUES (?, ?, ?)";

            // Prepare the SQL statement
            PreparedStatement stmt = connection.prepareStatement(sql);
            stmt.setInt(1, id);        // Set the first parameter (id)
            stmt.setString(2, name); // Set the second parameter (name)
            stmt.setInt(3, phone);  // Set the third parameter (phone)

            // Execute the insert statement
            int rowInserted = stmt.executeUpdate();
            if (rowInserted > 0) {
                System.out.println("Values inserted successfully");
            }

            // SQL query to select all data from the database
            String selectSql = "SELECT * FROM crud_operation";
            Statement selectStatement = connection.createStatement();
            ResultSet rs = selectStatement.executeQuery(selectSql);

            // Process the result set
            while (rs.next()) {
```

```
                System.out.println(rs.getInt(1) + " " + rs.getString(2) + " " +
rs.getInt(3));
            }

            JOptionPane.showMessageDialog(null, rs.toString(), "Database
Results", JOptionPane.INFORMATION_MESSAGE);

            // Close the statement and the connection
            stmt.close();
            connection.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

4. Understand callable Statement

- ❖ CallableStatement interface is used to call the **stored procedures and functions**.
- ❖ We can have business logic on the database by the use of stored procedures and functions that will make the performance better because these are precompiled.

The example to get the instance of CallableStatement is given below:

CallableStatement stmt=con.prepareCall("{call myprocedure(?,?)}");

Process:

To implement the stored procedure defined in your setup_database.sql file and call it from a Java application, you'll follow these steps:

1. **Create and Run the SQL File**: This step involves running your setup_database.sql file to set up the database, tables, and stored procedure.
2. **Call the Stored Procedure from Java**: This step involves writing a Java program that connects to the MySQL database and calls the stored procedure using CallableStatement.

```
-- Create a new database
CREATE DATABASE IF NOT EXISTS School;
```

```sql
-- Use the new database
USE School;

-- Create a new table
CREATE TABLE IF NOT EXISTS Students (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    phone INT NOT NULL
);

-- Insert some data into the table
INSERT INTO Students (name, phone) VALUES
('Alice', 1234567890),
('Bob', 9876543210),
('Charlie', 5551234567);

-- Create a stored procedure
DELIMITER //

CREATE PROCEDURE insertAndSelectStudent(
    IN student_id INT,
    IN student_name VARCHAR(100),
    IN student_phone INT
)
BEGIN
    -- Insert a new student record
    INSERT INTO Students (id, name, phone) VALUES (student_id, student_name, student_phone);

    -- Select all student records
    SELECT * FROM Students;
END //

DELIMITER ;
```

Steps:

using MySQL Workbench:

1. Open MySQL Workbench.
2. Open a new SQL tab.
3. Go to **File > Open SQL** Script and select your **setup_database.sql** file.
4. Click the lightning bolt icon to execute the script.

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.CallableStatement;
import java.sql.ResultSet;
import java.util.Scanner;

import javax.swing.JOptionPane;

public class Callable {
    public static void main(String[] args) {
        Connection connection = null;
        CallableStatement callableStmt = null;
        ResultSet rs = null;
```

```
        Scanner sc = null;

        try {
            // Load the MySQL JDBC Driver
            Class.forName("com.mysql.cj.jdbc.Driver");

            // Establish a connection to the database
            connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/School", "root",
"Meet@123");

            sc = new Scanner(System.in);

            System.out.print("Enter id for student: ");
            int id = sc.nextInt();
            sc.nextLine(); // Consume newline

            System.out.print("Enter name for student: ");
            String name = sc.nextLine();

            System.out.print("Enter mobile no for student: ");
            int phone = sc.nextInt();

            // Prepare the call to the stored procedure
            String sql = "{CALL insertAndSelectStudent(?, ?, ?)}";
            callableStmt = connection.prepareCall(sql);

            // Set the parameters for the stored procedure
            callableStmt.setInt(1, id);
            callableStmt.setString(2, name);
            callableStmt.setInt(3, phone);

            // Execute the stored procedure
            boolean hasResults = callableStmt.execute();

            // Process the result set(s)
            StringBuilder result = new StringBuilder();
            if (hasResults) {
                rs = callableStmt.getResultSet();
                while (rs.next()) {
                    result.append(rs.getInt("id")).append(" ")
                          .append(rs.getString("name")).append(" ")
                          .append(rs.getInt("phone")).append("\n");
                }

                System.out.println(result);
            } else {
                JOptionPane.showMessageDialog(null, "No data found", "Database Results",
JOptionPane.INFORMATION_MESSAGE);
            }

        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                // Close the ResultSet
                if (rs != null) rs.close();
                // Close the CallableStatement
                if (callableStmt != null) callableStmt.close();
                // Close the Connection
                if (connection != null) connection.close();
                // Close the Scanner
                if (sc != null) sc.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
```

```
82.     }
83. }
84.
```