**1.WAP to implement connection between client and server using socket programming.**

MyClient.java:

import java.rmi.Naming;

public class MyClient {

    public static void main(String[] args){

        try{

            MyRemote remoteObject = (MyRemote) Naming.lookup("rmi://localhost:1099/MyRemote");

            String response = remoteObject.sayHello();

            System.out.println("Response from server :" +response);

        } catch (Exception e) {

            e.printStackTrace();

 }

    }

}

MyRemote.java:

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface MyRemote extends Remote {

    String sayHello() throws RemoteException;

}

MyRemoteImpl.java:

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class MyRemoteImpl extends UnicastRemoteObject implements MyRemote {

    protected MyRemoteImpl() throws RemoteException {

        super();

    }

    @Override

    public String sayHello() throws RemoteException {

        return"Hello, world!!";

 }

}

MyServer.java:

import java.rmi.Naming;

import java.rmi.registry.LocateRegistry;

public class MyServer {

    public static void main(String[] args) {

        try {

            LocateRegistry.createRegistry(1099); //start RMI registry on port

            MyRemoteImpl remoteObject = new MyRemoteImpl();

            Naming.rebind( "rmi://localhost:1099/MyRemote", remoteObject);

            System.out.println( "Server is ready...");

        } catch(Exception e){

            e.printStackTrace();

 }

}

}

Output:

Server is ready...

Response from server :Hello, world!!

**2.WAP to implement RMI.**

1. Create the Remote Interface:

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface Calculator extends Remote {

int add(int a, int b) throws RemoteException;

int subtract(int a, int b) throws RemoteException;

}

1. Implement the Remote Interface:

import java.rmi.server.UnicastRemoteObject;

import java.rmi.RemoteException;

public class CalculatorImpl extends UnicastRemoteObject implements Calculator {

protected CalculatorImpl() throws RemoteException {

super();

}

@Override

public int add(int a, int b) throws RemoteException {

return a + b;

}

@Override

public int subtract(int a, int b) throws RemoteException {

return a - b;

}

}

1. Create the Server Program:

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

public class RMIServer {

public static void main(String[] args) {

try {

// Create an instance of the CalculatorImpl class

CalculatorImpl calculator = new CalculatorImpl();

// Create and get reference to registry

Registry registry = LocateRegistry.createRegistry(1099);

// Bind the remote object (stub) in the registry

registry.rebind("CalculatorService", calculator);

System.out.println("Server started...");

} catch (Exception e) {

e.printStackTrace();

}

}

}

1. Create the Client Program:

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

public class RMIClient {

public static void main(String[] args) {

try {

// Get registry

Registry registry = LocateRegistry.getRegistry("localhost", 1099);

// Lookup the remote object

Calculator calculator = (Calculator) registry.lookup("CalculatorService");

// Call the remote methods

int sum = calculator.add(5, 3);

int difference = calculator.subtract(10, 7);

System.out.println("Sum: " + sum);

System.out.println("Difference: " + difference);

} catch (Exception e) {

e.printStackTrace();

}

}

}

**Steps to Run the Application**

1. **Compile the Java files**:

bash

javac Calculator.java CalculatorImpl.java RMIServer.java RMIClient.java

1. **Start the RMI registry**:

bash

rmiregistry

1. **Start the Server**:

bash

java RMIServer

1. **Run the Client**:

bash

java RMIClient

**3.WAP to implement 2D object Using JAVA FX**

import javafx.application.Application;

import javafx.scene.Scene;

import javafx.scene.layout.Pane;

import javafx.scene.paint.Color;

import javafx.scene.shape.Circle;

import javafx.scene.shape.Rectangle;

import javafx.stage.Stage;

public class Simple2DShapes extends Application {

@Override

public void start(Stage primaryStage) {

// Create a pane to hold the shapes

Pane pane = new Pane();

// Create a rectangle

Rectangle rectangle = new Rectangle(50, 50, 200, 100);

rectangle.setFill(Color.LIGHTBLUE); // Set the fill color of the rectangle

rectangle.setStroke(Color.BLACK); // Set the stroke color of the rectangle

// Create a circle

Circle circle = new Circle(300, 150, 50);

circle.setFill(Color.YELLOW); // Set the fill color of the circle

circle.setStroke(Color.BLACK); // Set the stroke color of the circle

// Add the shapes to the pane

pane.getChildren().addAll(rectangle, circle);

// Create a scene with the pane

Scene scene = new Scene(pane, 400, 300);

// Set up the stage

primaryStage.setTitle("Simple 2D Shapes in JavaFX");

primaryStage.setScene(scene);

primaryStage.show();

}

public static void main(String[] args) {

launch(args); // Launch the JavaFX application

}

}

**4.WAP to implement 3D object Using JAVA FX.**

import javafx.application.Application;

import javafx.scene.Group;

import javafx.scene.PerspectiveCamera;

import javafx.scene.Scene;

import javafx.scene.paint.Color;

import javafx.scene.paint.PhongMaterial;

import javafx.scene.shape.Box;

import javafx.scene.transform.Rotate;

import javafx.scene.transform.Translate;

import javafx.stage.Stage;

public class Simple3DBox extends Application {

@Override

public void start(Stage primaryStage) {

// Create a Box (a 3D shape)

Box box = new Box(200, 200, 200);

// Set the material for the box (color, etc.)

PhongMaterial material = new PhongMaterial();

material.setDiffuseColor(Color.DODGERBLUE);

material.setSpecularColor(Color.LIGHTBLUE);

box.setMaterial(material);

// Rotate the box to make it more visible in 3D space

box.getTransforms().addAll(

new Rotate(45, Rotate.Y\_AXIS),

new Rotate(20, Rotate.X\_AXIS)

);

// Create a group to hold the box

Group group = new Group();

group.getChildren().add(box);

// Create a camera to view the 3D scene

PerspectiveCamera camera = new PerspectiveCamera(true);

camera.getTransforms().addAll(

new Rotate(-20, Rotate.Y\_AXIS),

new Translate(0, 0, -600) // move the camera back to view the box

);

// Create a scene with the group and set the camera

Scene scene = new Scene(group, 800, 600, true);

scene.setFill(Color.LIGHTGRAY); // Background color

scene.setCamera(camera);

// Set up the stage

primaryStage.setTitle("Simple 3D Box in JavaFX");

primaryStage.setScene(scene);

primaryStage.show();

}

public static void main(String[] args) {

launch(args); // Launch the JavaFX application

}

}