**Java RMI String Concat:**  
Interface:

package com.rmi.java;

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface StringConcatenationService extends Remote {

    boolean checkConcatenationEquality(String[] array1, String[] array2) throws RemoteException;

}

Interface Impl:

package com.rmi.java;

import java.rmi.Naming;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.server.UnicastRemoteObject;

public class StringConcatenationServiceImpl extends UnicastRemoteObject implements StringConcatenationService {

    protected StringConcatenationServiceImpl() throws RemoteException {

        super();

    }

    @Override

    public boolean checkConcatenationEquality(String[] array1, String[] array2) throws RemoteException {

        String concatenated1 = concatenateWithoutSpaces(array1);

        String concatenated2 = concatenateWithoutSpaces(array2);

        return concatenated1.equals(concatenated2);

    }

    private String concatenateWithoutSpaces(String[] array) {

        StringBuilder concatenated = new StringBuilder();

        for (String str : array) {

            concatenated.append(str.replaceAll("\\s+", ""));

        }

        return concatenated.toString();

    }

    public static void main(String[] args) {

        try {

            // Specify the port number here (e.g., 1098)

            int portNumber = 1099;

            LocateRegistry.createRegistry(portNumber);

            StringConcatenationService service = new StringConcatenationServiceImpl();

            String url = "//localhost:" + portNumber + "/StringConcatenationService";

            Naming.rebind(url, service);

            System.out.println("Server is running on port " + portNumber + "...");

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}

Client:

package com.rmi.java;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.util.Scanner;

public class Client {

    public static void main(String[] args) {

        try {

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

            StringConcatenationService service = (StringConcatenationService) registry.lookup("StringConcatenationService");

            Scanner scanner = new Scanner(System.in);

            System.out.println("Enter the first string array (comma-separated): ");

            String[] array1 = scanner.nextLine().split(",");

            for (int i = 0; i < array1.length; i++) {

                array1[i] = array1[i].trim(); // Trim the string to remove leading and trailing spaces

            }

            System.out.println("Enter the second string array (comma-separated): ");

            String[] array2 = scanner.nextLine().split(",");

            for (int i = 0; i < array2.length; i++) {

                array2[i] = array2[i].trim(); // Trim the string to remove leading and trailing spaces

            }

            boolean result = service.checkConcatenationEquality(array1, array2);

            System.out.println("Are the concatenated strings equal? " + result);

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}

**RMI Palindrome Interface**

package com.rmi.java2;

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface PalindromeCheckService extends Remote {

    boolean isPalindrome(String str) throws RemoteException;

}

Palindrome Service Impl

package com.rmi.java2;

import java.rmi.Naming;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.rmi.server.UnicastRemoteObject;

public class PalindromeCheckServiceImpl extends UnicastRemoteObject implements PalindromeCheckService {

    protected PalindromeCheckServiceImpl() throws RemoteException {

        super();

    }

    @Override

    public boolean isPalindrome(String str) throws RemoteException {

        // Implementation of palindrome check

      str = str.replaceAll("\\s+", "").toLowerCase(); // Remove spaces and convert to lowercase

          int left = 0;

          int right = str.length() - 1;

          while (left < right) {

              if (str.charAt(left) != str.charAt(right)) {

                  return false;

              }

              left++;

              right--;

          }

          return true;

    }

    public static void main(String[] args) {

        try {

            // Specify the port number here

            int portNumber = 1029; // Change this to your desired port number

            // Create RMI registry on the specified port

            Registry registry = LocateRegistry.createRegistry(portNumber);

            PalindromeCheckService service = new PalindromeCheckServiceImpl();

            // Specify the URL for binding

            String url = "//localhost:" + portNumber + "/PalindromeCheckService";

            // Use Naming.rebind() to bind the service to the specified URL

            Naming.rebind(url, service);

            System.out.println("Palindrome Check Service is running on port " + portNumber + "...");

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}

Client

package com.rmi.java2;

import java.rmi.Naming;

import java.util.Scanner;

public class Client {

    public static void main(String[] args) {

        try {

            // Specify the port number here

            int portNumber = 1029; // Change this to the port number where the RMI service is running

            // Specify the URL to look up the RMI service

            String url = "//localhost:" + portNumber + "/PalindromeCheckService";

            // Use Naming.lookup() to look up the RMI service

            PalindromeCheckService service = (PalindromeCheckService) Naming.lookup(url);

            Scanner scanner = new Scanner(System.in);

            System.out.println("Enter a string to check if it's a palindrome: ");

            String str = scanner.nextLine();

            boolean isPalindrome = service.isPalindrome(str);

            if (isPalindrome) {

                System.out.println("The string '" + str + "' is a palindrome.");

            } else {

                System.out.println("The string '" + str + "' is not a palindrome.");

            }

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}

**HTML Reader**

package htmlreader.java;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.HttpURLConnection;

import java.net.URL;

public class HtmlRetriever {

    public static void main(String[] args) {

        String urlString = "https://www.google.com"; // Specify the URL here

        try {

            // Create URL object

            URL url = new URL(urlString);

            // Create HttpURLConnection object

            HttpURLConnection connection = (HttpURLConnection) url.openConnection();

            // Set request method

            connection.setRequestMethod("GET");

            // Get input stream from connection

            BufferedReader reader = new BufferedReader(new InputStreamReader(connection.getInputStream()));

            // Read HTML content

            StringBuilder htmlContent = new StringBuilder();

            String line;

            while ((line = reader.readLine()) != null) {

                htmlContent.append(line);

            }

            // Close reader

            reader.close();

            // Display HTML content on console

            System.out.println("HTML Content:");

            System.out.println(htmlContent.toString());

            // Save HTML content to a file

            String fileName = "output.html";

            BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));

            writer.write(htmlContent.toString());

            writer.close();

            System.out.println("HTML content saved to file: " + fileName);

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}

**RandomGeneratorGUI**

package com.java.question4;

import javax.swing.\*;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.util.Random;

public class RandomGeneratorGUI {

    private JTextField outputField;

    public static void main(String[] args) {

        SwingUtilities.invokeLater(() -> {

            new RandomGeneratorGUI().createAndShowGUI();

        });

    }

    private void createAndShowGUI() {

        // Create the main frame

        JFrame frame = new JFrame("Random Number Generator");

        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        frame.setSize(300, 150);

        frame.setLayout(new BoxLayout(frame.getContentPane(), BoxLayout.Y\_AXIS));

        // Create text field for output

        outputField = new JTextField(20);

        outputField.setEditable(false);

        // Add mouse listener to the text field

        outputField.addMouseListener(new MouseAdapter() {

            @Override

            public void mousePressed(MouseEvent e) {

                generateRandomInteger();

            }

            @Override

            public void mouseReleased(MouseEvent e) {

                generateRandomDouble();

            }

        });

        // Add components to the frame

        frame.add(new JLabel("Click and hold the mouse to generate random integer"));

        frame.add(new JLabel("Release the mouse to generate random double"));

        frame.add(outputField);

        // Display the frame

        frame.setVisible(true);

    }

    private void generateRandomInteger() {

        Random random = new Random();

        int randomInt = random.nextInt(100); // Adjust the range as needed

        outputField.setText("Random Integer: " + randomInt);

    }

    private void generateRandomDouble() {

        Random random = new Random();

        double randomDouble = random.nextDouble();

        outputField.setText("Random Double: " + randomDouble);

    }

}

**JDBC**

package com.question2.java;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.sql.Statement;

import javax.sql.rowset.JdbcRowSet;

import javax.sql.rowset.CachedRowSet;

import javax.sql.rowset.\*;

public class EmployeeDataManagement {

    public static void main(String[] args) {

        String dbURL = "jdbc:mysql://localhost:3306/ayush\_java";

        String username = "root";

        String password = "mysqlroot778$";

        try {

            // Connect to the database using JdbcRowSet (Connected RowSet)

            JdbcRowSet jdbcRowSet = RowSetProvider.newFactory().createJdbcRowSet();

            jdbcRowSet.setUrl(dbURL);

            jdbcRowSet.setUsername(username);

            jdbcRowSet.setPassword(password);

            jdbcRowSet.setCommand("SELECT \* FROM employee");

            jdbcRowSet.execute();

            // Fetch and display the list of all employees

            while (jdbcRowSet.next()) {

                System.out.println("Employee ID: " + jdbcRowSet.getInt("EmployeeID"));

                System.out.println("Name: " + jdbcRowSet.getString("Name"));

                System.out.println("Salary: " + jdbcRowSet.getInt("Salary"));

                System.out.println("------------------");

            }

            // Update an employee's salary using JdbcRowSet

            jdbcRowSet.beforeFirst();

            while (jdbcRowSet.next()) {

                if (jdbcRowSet.getInt("EmployeeID") == 5) {

                    jdbcRowSet.updateInt("Salary", 50000);

                    jdbcRowSet.updateRow();

                }

            }

            // Create a CachedRowSet (Disconnected RowSet) for new employee insertion

            CachedRowSet cachedRowSet =  RowSetProvider.newFactory().createCachedRowSet();

            cachedRowSet.setUrl(dbURL);

            cachedRowSet.setUsername(username);

            cachedRowSet.setPassword(password);

            cachedRowSet.setCommand("SELECT \* FROM employee");

            cachedRowSet.execute();

            cachedRowSet.moveToInsertRow();

            cachedRowSet.updateInt("EmployeeID", 5);

            cachedRowSet.updateString("Name", "New Employee");

            cachedRowSet.updateInt("Salary", 60000);

            cachedRowSet.insertRow();

            cachedRowSet.moveToCurrentRow();

            cachedRowSet.acceptChanges();

            ((Connection) cachedRowSet).setAutoCommit(false);

            System.out.println("Employee added successfully.");

            // Clean up resources

            jdbcRowSet.close();

            cachedRowSet.close();

        } catch (SQLException e) {

            e.printStackTrace();

        }

    }

}