

TASK 4

Create a distributed system using Java RMI(Remote Method Invocation) where multiple nodes interact and share resources.

Solution

The image displays two screenshots from the NetBeans IDE, illustrating the process of creating a new Java class.

Top Screenshot: New File Dialog

- Steps:**
 1. Choose File Type
 2. ...
- Choose File Type:**
 - Project:** ChatApplication
 - Filter:** (empty)
 - Categories:**
 - Java
 - Swing GUI Forms
 - JavaBeans Objects
 - AWT GUI Forms
 - Unit Tests
 - Selenium Tests
 - Micronaut
 - Persistence
 - Groovy
 - Web Services
 - File Types:**
 - Java Class (selected)
 - Java Interface
 - Java Enum
 - Java Annotation Type
 - Java Exception
 - Java Package Info
 - Java Module Info
 - JApplet
 - Applet
 - Java Main Class
 - Java Singleton Class
 - Description:** Creates a new plain Java class. This template is useful for creating new non-visual classes.
- Navigation:** < Back, Next > (highlighted), Finish, Cancel, Help

Bottom Screenshot: New Java Class Dialog

- Steps:**
 1. Choose File Type
 2. Name and Location
- Name and Location:**
 - Class Name:** DistributedSystemRMI
 - Project:** ChatApplication
 - Location:** Source Packages
 - Package:** (empty)
 - Created File:** \NetBeansProjects\ChatApplication\src\main\java\DistributedSystemRMI.java
 - Superclass:** (empty) with a Browse... button
 - Interfaces:** (empty) with a Browse... button
- Warning:** ⚠ Warning: It is highly recommended that you do not place Java classes in the default package
- Navigation:** < Back, Next >, Finish (highlighted), Cancel, Help

New Java Interface

Steps

1. Choose File Type

2. **Name and Location**

Name and Location

Class Name: RemoteInterface

Project: ChatApplication

Location: Source Packages

Package:

Created File: >cuments\NetBeansProjects\ChatApplication\src\main\java\RemoteInterface.java

Superinterface: Browse...

Warning: It is highly recommended that you do not place Java classes in the default package.

< Back

Next >

Finish

Cancel

Help

DistributedSystemRMI.java x RemoteInterface.java x

Source History

```
1 import java.rmi.Remote;
2 import java.rmi.RemoteException;
3
4 public interface RemoteInterface extends Remote {
5     String sayHello(String name) throws RemoteException;
6 }
7
```

New Java Class

Steps

1. Choose File Type

2. **Name and Location**

Name and Location

Class Name: RemoteImplementation

Project: ChatApplication

Location: Source Packages

Package:

Created File: ts\NetBeansProjects\ChatApplication\src\main\java\RemoteImplementation.java

Superclass: Browse...

Interfaces: Browse...

Warning: It is highly recommended that you do not place Java classes in the default package.

< Back

Next >

Finish

Cancel

Help

```
DistributedSystemRMI.java x RemoteInterface.java x RemoteImplementation.java x
Source History
1 import java.rmi.RemoteException;
2 import java.rmi.server.UnicastRemoteObject;
3
4 public class RemoteImplementation extends UnicastRemoteObject implements RemoteInterface {
5     protected RemoteImplementation() throws RemoteException {
6         super();
7     }
8
9     @Override
10    public String sayHello(String name) throws RemoteException {
11        return "Hello, " + name + "!";
12    }
13 }
14
```

New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name:

Project:


Location:

Package:

Created File:

Superclass:

Interfaces:

 Warning: It is highly recommended that you do not place Java classes in the default package.

```
...va RemoteInterface.java x RemoteImplementation.java x Server.java x Client.java x
Source History
1 import java.rmi.Naming;
2 import java.rmi.registry.LocateRegistry;
3 import java.rmi.RemoteException;
4 import java.net.MalformedURLException;
5
6 public class Server {
7     public static void main(String[] args) {
8         try {
9             LocateRegistry.createRegistry(1099); // Default RMI registry port
10            RemoteImplementation obj = new RemoteImplementation();
11            Naming.rebind("//localhost/RemoteHello", obj);
12            System.out.println("Server is ready.");
13        } catch (RemoteException e) {
14            System.err.println("Server RemoteException: " + e.toString());
15        } catch (MalformedURLException e) {
16

```

```

17         System.err.println("Server MalformedURLException: " + e.toString());
18
19     } catch (Exception e) {
20         System.err.println("Server Exception: " + e.toString());
21     }
22 }
23
24 }
25
26

```

New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name:

Project:

Location:

Package:

Created File:

Superclass:

Interfaces:

Warning: It is highly recommended that you do not place Java classes in the default package.

...va
RemoteInterface.java
RemoteImplementation.java
Server.java
Client.java

Source
History

```

1 import java.rmi.Naming;
2
3 public class Client {
4     public static void main(String[] args) {
5         try {
6             RemoteInterface stub = (RemoteInterface) Naming.lookup("//localhost/RemoteHello");
7             String response = stub.sayHello("Muneeba");
8             System.out.println("Response: " + response);
9         } catch (Exception e) {
10             System.err.println("Client exception: " + e.toString());
11             e.printStackTrace();
12         }
13     }
14 }
15

```

Output x

Run Server

