

St. Francis Institute of Technology
Department of Computer Engineering

Academic Year: 2021-2022

Semester: VIII

Subject: DCL

Class/Branch/: BE/CMPNA

Name :- Nithin Menezes

Roll Number: 56

Experiment 3

Aim: Remote Procedure Call/ Remote Method Invocation Operation for Client Sever Model Implementation

Pre - requisites: Networking

Theory:

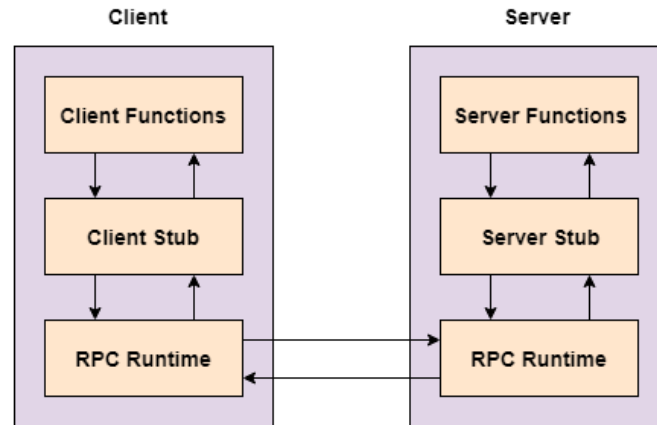
A remote procedure call is an interprocess communication technique that is used for client-server based applications. It is also known as a subroutine call or a function call.

A client has a request message that the RPC translates and sends to the server. This request may be a procedure or a function call to a remote server. When the server receives the request, it sends the required response back to the client. The client is blocked while the server is processing the call and only resumed execution after the server is finished.

The sequence of events in a remote procedure call are given as follows –

- The client stub is called by the client.
- The client stub makes a system call to send the message to the server and puts the parameters in the message.
- The message is sent from the client to the server by the client's operating system.
- The message is passed to the server stub by the server operating system.
- The parameters are removed from the message by the server stub.
- Then, the server procedure is called by the server stub.

A diagram that demonstrates this is as follows –



The RMI (Remote Method Invocation) is an API that provides a mechanism to create distributed application in java. The RMI allows an object to invoke methods on an object running in another JVM.

The RMI provides remote communication between the applications using two objects stub and skeleton.

Understanding stub and skeleton

RMI uses stub and skeleton object for communication with the remote object.

A remote object is an object whose method can be invoked from another JVM. Let's understand the stub and skeleton objects:

Stub:

The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object. When the caller invokes method on the stub object, it does the following tasks:

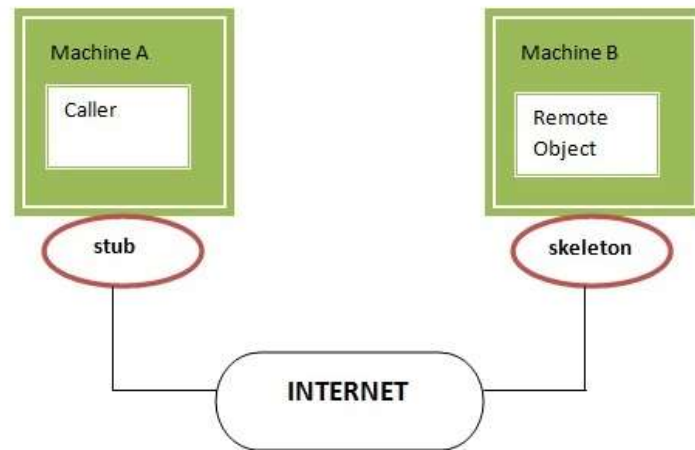
1. It initiates a connection with remote Virtual Machine (JVM),
2. It writes and transmits (marshals) the parameters to the remote Virtual Machine (JVM),
3. It waits for the result
4. It reads (unmarshals) the return value or exception, and
5. It finally, returns the value to the caller.

Skeleton:

The skeleton is an object, acts as a gateway for the server side object. All the incoming requests are routed through it. When the skeleton receives the incoming request, it does the following tasks:

1. It reads the parameter for the remote method
2. It invokes the method on the actual remote object, and
3. It writes and transmits (marshals) the result to the caller.

In the Java 2 SDK, an stub protocol was introduced that eliminates the need for skeletons.



Understanding requirements for the distributed applications

If any application performs these tasks, it can be distributed application.

1. The application need to locate the remote method
2. It need to provide the communication with the remote objects, and
3. The application need to load the class definitions for the objects.

Program:

Calculator:

```

public interface Calculator extends java.rmi.Remote {
    public long add(long a, long b) throws java.rmi.RemoteException;

    public long sub(long a, long b) throws java.rmi.RemoteException;

    public long mul(long a, long b) throws java.rmi.RemoteException;

    public long div(long a, long b) throws java.rmi.RemoteException;
}
  
```

CalculatorImpl:

```

public class CalculatorImpl extends
    java.rmi.server.UnicastRemoteObject implements Calculator {
  
```

// Implementations must have an

```

//explicit constructor
// in order to declare the
//RemoteException exception

public CalculatorImpl()
throws java.rmi.RemoteException { super();
}

public long add(long a, long b)
throws java.rmi.RemoteException { System.out.println ("Doing addition");
return a + b;
}

public long sub(long a, long b)
throws java.rmi.RemoteException { System.out.println ("Doing subtraction");
return a - b;
}

public long mul(long a, long b) throws java.rmi.RemoteException {
System.out.println ("Doing multiplication");
return a * b;
}

public long div(long a, long b)
throws java.rmi.RemoteException { System.out.println ("Doing division"); return a / b;
}
}

```

CalculatorClient:

```

import java.rmi.Naming;
import java.rmi.RemoteException;
import java.net.MalformedURLException; import java.rmi.NotBoundException;

public class CalculatorClient {

public static void main(String[] args) { try {
Calculator c = (Calculator)
Naming.lookup( "rmi://localhost/CalculatorService");

```

```

System.out.println( c.sub(10, 3) ); System.out.println( c.add(6, 9) ); System.out.println(
c.mul(4, 5) ); System.out.println( c.div(144, 12) );
}
catch (MalformedURLException murle) { System.out.println(); System.out.println(
"MalformedURLException"); System.out.println(murle);
}
catch (RemoteException re) { System.out.println(); System.out.println(
"RemoteException"); System.out.println(re);
}
catch (NotBoundException nbe) { System.out.println(); System.out.println(
"NotBoundException"); System.out.println(nbe);
}
catch (
java.lang.ArithmeticException
ae) { System.out.println(); System.out.println( "java.lang.ArithmeticException");
System.out.println(ae);
}
}
}
}

```

CalculatorServer:

```

import java.rmi.Naming; public class CalculatorServer {
public CalculatorServer() { try {
Calculator c = new CalculatorImpl();
Naming.rebind("rmi://localhost:1099/CalculatorService", c);
} catch (Exception e) { System.out.println("Trouble: " + e);
}
}

public static void main(String args[]) { new CalculatorServer();
}
}

```

Output:

Server:

```
Command Prompt - java CalculatorServer
Microsoft Windows [Version 10.0.19043.1466]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Nithin>cd C:\Users\Nithin\OneDrive\Desktop\Exp_3_DC

C:\Users\Nithin\OneDrive\Desktop\Exp_3_DC>javac CalculatorServer.java

C:\Users\Nithin\OneDrive\Desktop\Exp_3_DC>javac CalculatorImpl.java

C:\Users\Nithin\OneDrive\Desktop\Exp_3_DC>java CalculatorServer
Doing subtraction
Doing addition
Doing multiplication
Doing division
```

Client:

```
Select Command Prompt

C:\Users\Nithin>cd C:\Users\Nithin\OneDrive\Desktop\Exp_3_DC

C:\Users\Nithin\OneDrive\Desktop\Exp_3_DC>javac CalculatorClient.java

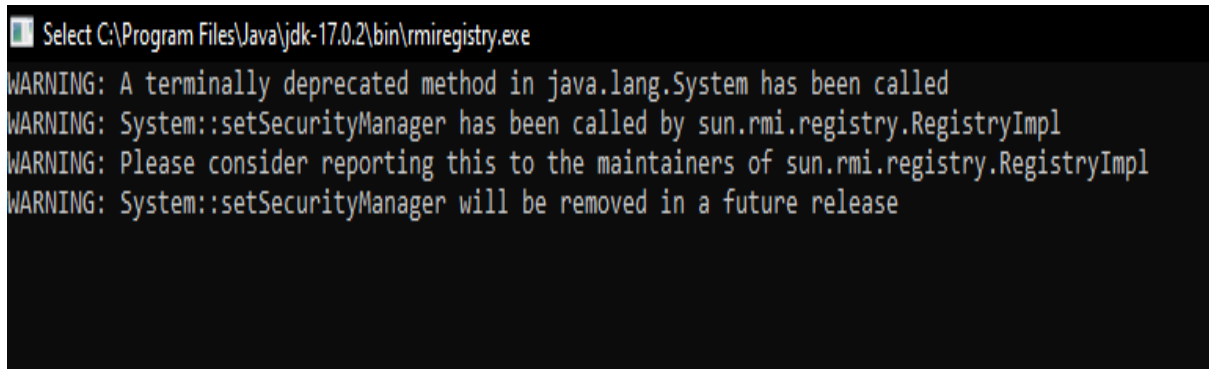
C:\Users\Nithin\OneDrive\Desktop\Exp_3_DC>javac Calculator.java

C:\Users\Nithin\OneDrive\Desktop\Exp_3_DC>start rmiregistry

C:\Users\Nithin\OneDrive\Desktop\Exp_3_DC>java CalculatorClient
7
15
20
12

C:\Users\Nithin\OneDrive\Desktop\Exp_3_DC>
```

rmiregistry:



```
Select C:\Program Files\Java\jdk-17.0.2\bin\rmiregistry.exe
WARNING: A terminally deprecated method in java.lang.System has been called
WARNING: System::setSecurityManager has been called by sun.rmi.registry.RegistryImpl
WARNING: Please consider reporting this to the maintainers of sun.rmi.registry.RegistryImpl
WARNING: System::setSecurityManager will be removed in a future release
```

Conclusion:

Thus, remote procedure call/ remote method invocation operation for client server model is implemented successfully.