St. Francis Institute of Technology
Department of Computer Engineering

Academic Year: 2021-2022 **Semester: VIII**
Subject: Distributed Computing **Class / Branch / Division: BE/CMPN/A**
Name: Rebecca Dias Roll Number:18

Experiment No: 03

Aim: Client-Server Implementation using RPC/ RMI

Theory:

Remote Method Invocation (RMI)

Socket-based communications require you to make explicit connections between servers and clients and transmit data through those connections. The Remote Method Invocation (RMI) in Java simplifies this situation by not requiring you to make explicit connections. Objects located on remote hosts can be accessed as if they were situated locally through this framework. The underlying connection and transmission of data are handled implicitly by the JVM. These are some terms that you must be familiar with:

Server: An program that provides services to objects residing on other machines. Service contract: An interface that defines the services provided by the server.

Client: An object that uses the services provided by the server program on a remote machine. Registry: A service provided to clients to locate a server on a remote machine providing the service.

Stub: An object that resides on the same host as the client and serves as a proxy or surrogate,of the remote server.

Skeleton: An object that resides on the same host as the server, receiving requests from the stubs and dispatching the requests to the server.

The client and server classes are written by programmers. The stubs and skeletons are created by theRMI compiler from the server code. This is how the RMI process works:

1. The client invokes a method that is offered by the server. The invocation is received by thestub.
2. The stub arranges the request and data in a linear stream. This is known as marshaling andsends that information to the machine that the server resides on.
3. The skeleton receives the request, unmarshals the information, and invokes the method onthe server object.
4. The server executes the method and returns the result to the skeleton.
5. The skeleton marshals the result and sends it to the stub.

6. The stub unmarshals the result and sends the result to the client object.

In this simple example on Remote Method Invocation, the server offers the services of a calculator that can perform elementary arithmetic operations. These are the steps that you would need to perform to set up the system. The sample code is appended below.

1. Clients must know what services the Calculator Server provides. RMI: Calculator interface lists the methods available to remote clients.
2. Clients must find a Calculator object on the server. RMI: The Calculator Server registers an object with a special server, called the rmi registry so that clients can look it up by name.
3. Clients must be able to pass arguments to and invoke a method of the Calculator object
4. located on the server. RMI: A special compiler called rmic creates a stub class for use on the client, and a skeleton class for use on the server. The stub takes the request from the client, passes the arguments to the skeleton which invokes the Calculator methods, and sends the return value back to the stub which passes it to the client.

Steps:

On the server machine:

1. Compile ICalculator.java, Calculator.java, and CalculatorServer.java
2. Create the stub and skeleton classes using the command **rmic Calculator**
3. Copy the Calculator_Stub.class to the client site
4. Start the registry using the command **rmiregistry&**
5. Start CalculatorServer using **java CalculatorServer serverHostname**

On the client machine:

1. Compile ICalculator.java and CalculatorClient.java
2. Run CalculatorClient using the command **java CalculatorClient serverHostname**

Programs:

RMI Client Side Program:

```
import Pyro4

num1 = float (input("Enter The First Number: "))
num2 = float (input("Enter The Second Number: "))
print ("A List Of Available Mathematical Operations \n")
print ("1.Multiple ")
print ("2.Divide ")
print ("3.Substract ")
print ("4.Add \n")
opr = int (input("Enter Your Choice (1,2,3,4): "))
ns = Pyro4.locateNS()
uri = ns.lookup('obj')
calculator = Pyro4.Proxy(uri)
```

```
if opr==1:
   print ("Result: ")
   print (calculator.Mul_func(num1,num2))
elif opr==2:
   print ("Result: ")
   print (calculator.Div_func(num1,num2))
elif opr==3:
   print ("Result: ")
   print (calculator.Sub_func(num1,num2))
elif opr==4:
   print ("Result: ")
   print (calculator.Add_func(num1,num2))
else:
   print ("\n Error : Mathematical Operations Input doesn't match With The List!")
```

RMI Server Side Program:

```
import Pyro4

@Pyro4.expose
class Calculator(object):
   def Mul_func(self,num1,num2):
      return (num1*num2)
   def Sub_func(self,num1,num2):
      if(num1<num2):
         a = num2-num1
      else:
         a = num1-num2
      return a
   def Add_func(self,num1,num2):
      return (num1+num2)
   def Div_func(self,num1,num2):
      return (num1/num2)

daemon = Pyro4.Daemon()
uri = daemon.register(Calculator)
ns = Pyro4.locateNS()
ns.register("obj", uri)
print("Ready.")
daemon.requestLoop()
```

OUTPUT:

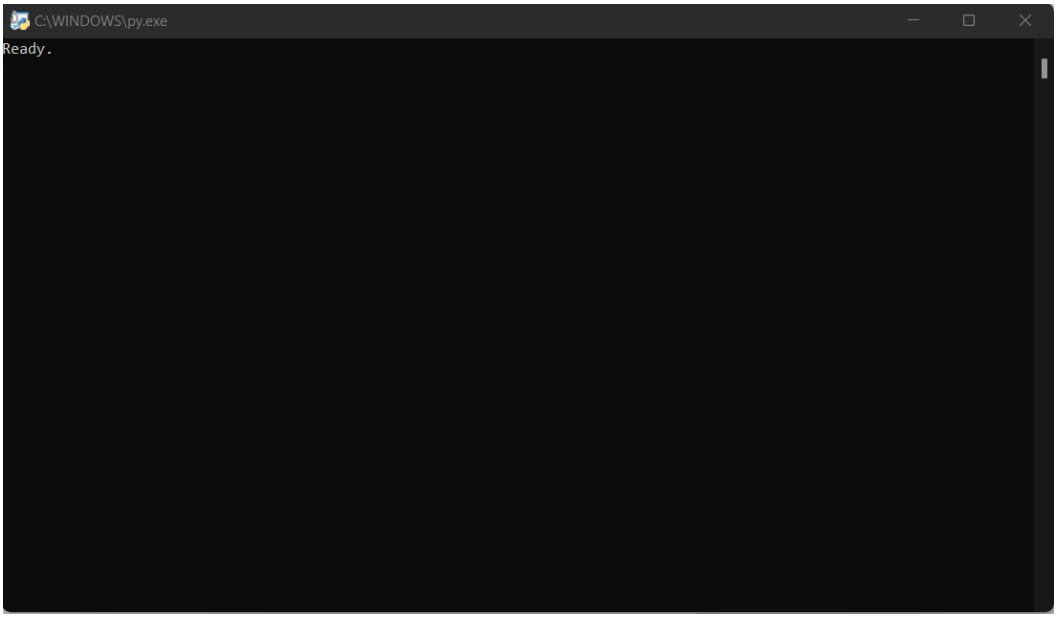
Client side:

```
  File "C:\Users\HP\AppData\Local\Programs\Python\Python37\lib\site-packages\Pyro4\core.py", line 615, in _pyroGetMetadata
Enter The First Number: 10
Enter The Second Number: 20
A List Of Available Mathematical Operations

1.Multiple
2.Divide
3.Substract
4.Add

Enter Your Choice (1,2,3,4): 1
Result:
200.0
PS F:\sem8\DC\Experiments>
```

RMI (invoke pyro)

```
C:\Users\HP>pyro4-ns
Not starting broadcast server for localhost.
NS running on localhost:9090 (127.0.0.1)
Warning: HMAC key not set. Anyone can connect to this server!
URI = PYRO:Pyro.NameServer@localhost:9090
```

Sever side:



Conclusion: In this experiment, we successfully implemented a remote procedure call.

Reference:

https://stackoverflow.com/questions/34499331/pyro4-failed-to-locate-the-nameserver/34500263
https://github.com/hcastillaq/Pyro4-Ejemplo
https://www.youtube.com/watch?v=BK0YWqj6r4
https://www.programcreek.com/python/example/102403/Pyro4.Daemon
https://pyro4.readthedocs.io/en/stable/servercode.html