

## ASSIGNMENT 2

### PROBLEM STATEMENT: -

Design a distributed application using RMI for remote computation where client submits two strings to the server and server returns the concatenation of the given strings.

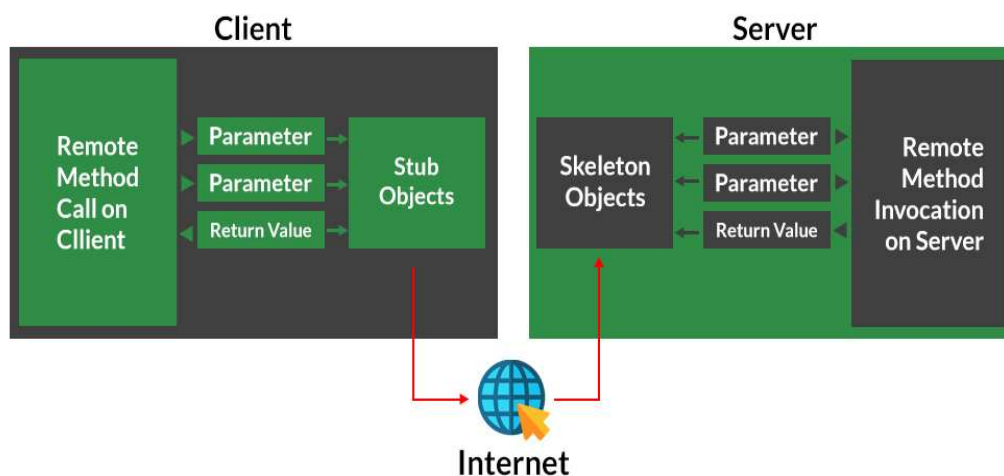
### OBJECTIVE:

1. Students should be able to Implement the server class that enables the remote interface.
2. Students should be able to start the RMI registry on the server side. This will allow clients to look up the server object by name.

### THEORY:

In this practical students need to design and develop a distributed Hotel booking application using Java RMI. A distributed hotel booking system consists of the hotel server and the client machines. The server manages hotel rooms booking information. A customer can invoke the following operations at his machine i) Book the room for the specific guest ii) Cancel the booking of a guest.

### Working of RMI




Remote Method Invocation (RMI) is a Java technology that allows a Java object running in one Java Virtual Machine (JVM) to invoke methods on an object running in another JVM. It enables distributed computing by providing a way for objects to interact across different JVMs, making it suitable for building distributed applications.

Following steps are involved in design of distributed application using RMI for remote computation where client submits two strings to the server and server returns the concatenation of the given strings.

In Python, you can use the Pyro4 library to implement a distributed application using RMI (Remote Method Invocation) for remote computation. Pyro4 is a Python Remote Objects library that simplifies the process of building distributed applications.

Before we start, we need to install the Pyro4 library. we can install it using:



```
bash
pip install Pyro4
```

### Server Implementation:

# server.py

```
import Pyro4
```

```
@Pyro4.expose
```

```
class StringConcatenationServer:
```

```
    def concatenate_strings(self, str1, str2):
```

```
        result = str1 + str2
```

```
        return result
```

```
def main():
```

```
    daemon = Pyro4.Daemon() # Create a Pyro daemon
```

```
    ns = Pyro4.locateNS() # Locate the Pyro nameserver
```

```
    # Create an instance of the server class
```

```
    server = StringConcatenationServer()
```

```
    # Register the server object with the Pyro nameserver
```

```
    uri = daemon.register(server)
```

```
    ns.register("string.concatenation", uri)
```

```
    print("Server URI:", uri)
```

```
    with open("server_uri.txt", "w") as f:
```

```
        f.write(str(uri))
```

```
daemon.requestLoop()
```

```
if __name__ == "__main__":
```

```
    main()
```

### **Client Implementation:**

**# client.py**

```
import Pyro4
```

```
def main():
```

```
    with open("server_uri.txt", "r") as f:
```

```
        uri = f.read()
```

```
    server = Pyro4.Proxy(uri) # Connect to the remote server
```

```
    str1 = input("Enter the first string: ")
```

```
    str2 = input("Enter the second string: ")
```

```
    result = server.concatenate_strings(str1, str2)
```

```
    print("Concatenated Result:", result)
```

```
if __name__ == "__main__":
```

```
    main()
```

### **Steps to Run:**

Install Pyro4 library

Then Use command Pyro4-ns

And use following steps

- 1) Save the server code in a file, e.g., server.py
- 2) Save the client code in a file, e.g., client.py
- 3) Open a terminal and run the server: `python server.py`
- 4) you will get server uri paste it in server\_uri.txt file. keep it in same folder where you have stored python files.
- 5) Open another terminal and run the client: `python client.py`
- 6) enter the values for concatenation.

### **CONCLUSION:**

In this way we have design a distributed application using RMI for remote computation where client submits two strings to the server and server returns the concatenation of the given strings.

### **ORAL QUESTION**

1. What is RMI (Remote Method Invocation) and how does it facilitate communication between distributed Java applications?
2. Explain the client-server architecture in the context of RMI-based distributed applications.
3. How would you define the role of the client and the server in an RMI-based system for remote computation?
4. Can you outline the steps involved in implementing an RMI-based solution for remote computation in Java?
5. Describe the process of registering a remote object with the RMI registry.