

Task 1: RPC Implementation Using Python

A. RPC Server Program:

```
# importing from xmlrpc.server, SimpleXMLRPCServer
from xmlrpc.server import SimpleXMLRPCServer

# start the server
with SimpleXMLRPCServer(('127.0.0.1', 8000)) as server:
    print("Listening on port 8000...")

    def factorial(n):
        if (n==0 or n==1):
            return 1
        else:
            return n * factorial(n-1)

    # register the name and function (both) when calling the
    server from the client
    server.register_function(factorial, "factorial")
    server.serve_forever()
```

OUTPUT:

```
C:\Windows\System32\cmd.exe - python xmlrpc_server.py
Microsoft Windows [Version 10.0.19044.1766]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Yashuv\Desktop\Distributed System Assignment\XML-RPC in python>python xmlrpc_server.py
Listening on port 8000...
```

B. RPC Client Program:

```
# importing xmlrpc client
import xmlrpc.client

# indicate the server in .ServerProxy
with xmlrpc.client.ServerProxy('http://127.0.0.1:8000/') as proxy:

    # client performs an RPC by sending an HTTP request to a server
    while True:

        n = int(input('\tEnter number for Factorial: '))

        print("\t",n,"! = ",proxy.factorial(n), end='\n\n')
```

OUTPUT:

```
cmd Select C:\Windows\System32\cmd.exe - python xmlrpc_client.py
Microsoft Windows [Version 10.0.19044.1766]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Yashuv\Desktop\Distributed System Assignment\XML-RPC in python>python xmlrpc_client.py
Enter number for Factorial: 6
6 ! = 720

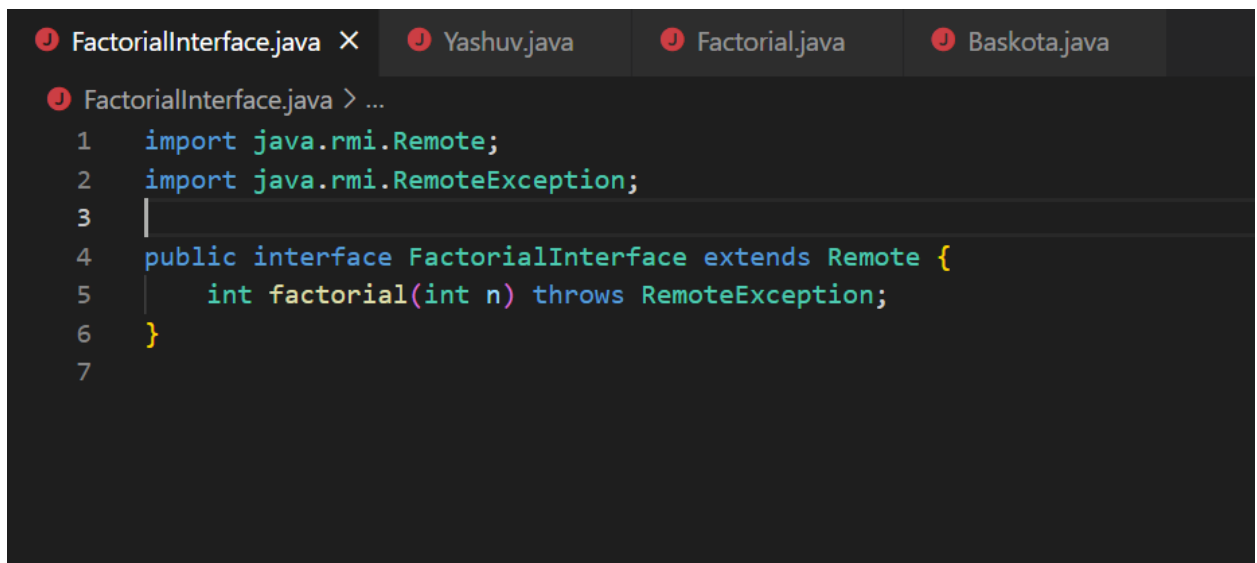
Enter number for Factorial: 5
5 ! = 120

Enter number for Factorial: 10
10 ! = 3628800

Enter number for Factorial:
```

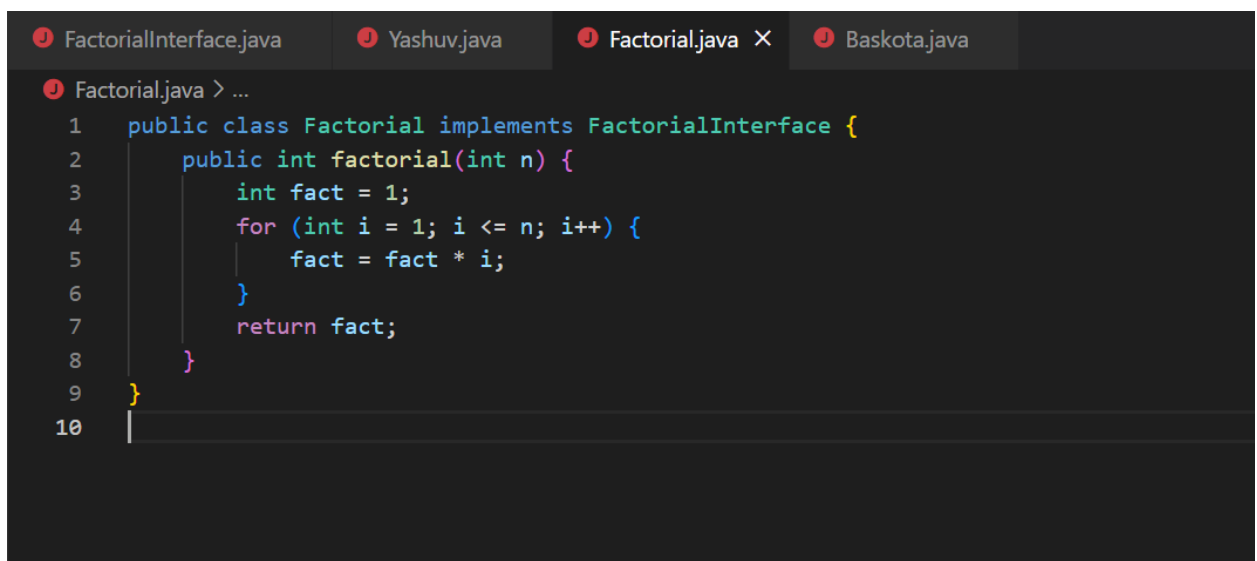
Task 2: RMI Implementation Using Java

A. Program for Remote Interface in *FactorialInterface.java* :

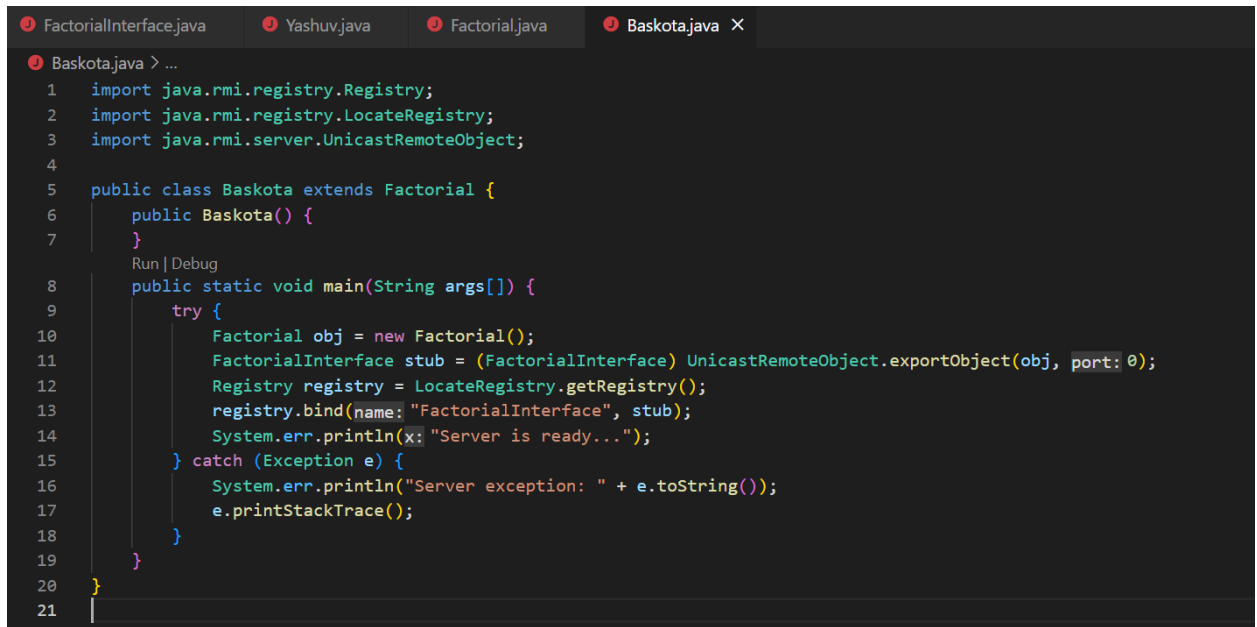


```
FactorialInterface.java X Yashuv.java Factorial.java Baskota.java
FactorialInterface.java > ...
1  import java.rmi.Remote;
2  import java.rmi.RemoteException;
3
4  public interface FactorialInterface extends Remote {
5      int factorial(int n) throws RemoteException;
6  }
7
```

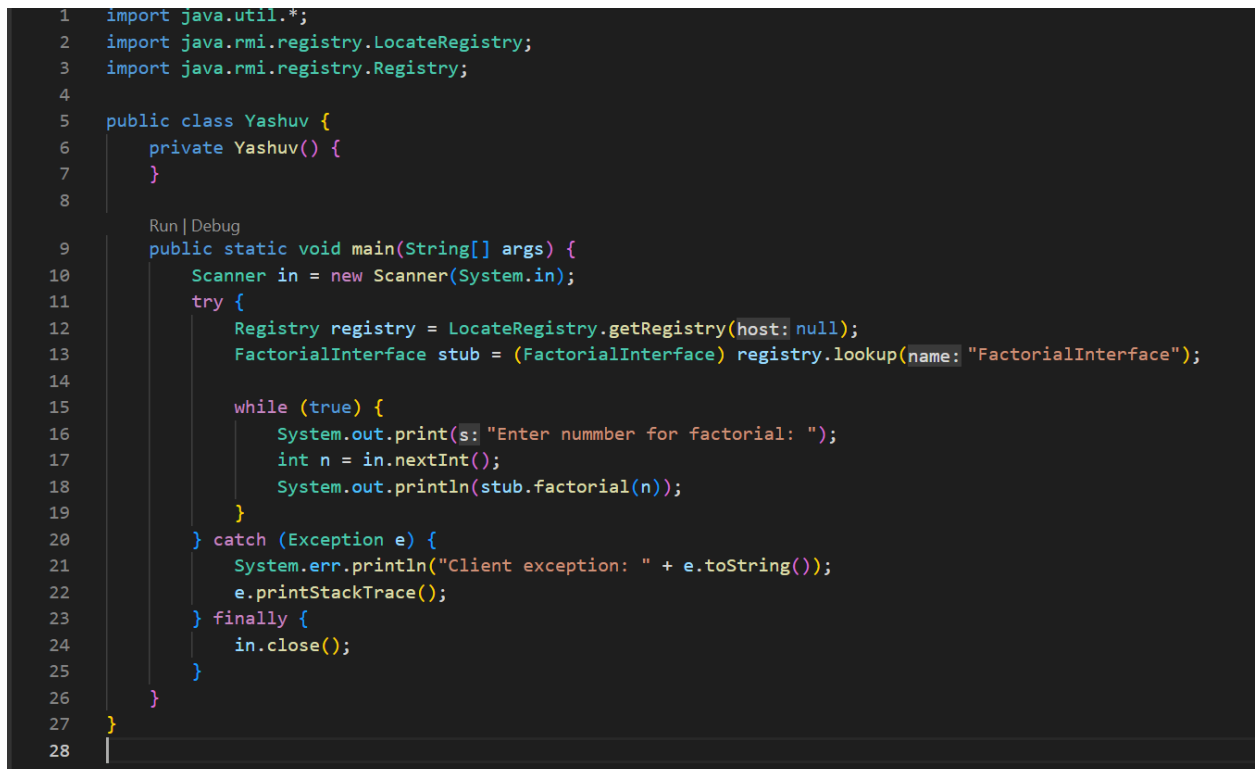
B. Program for Implementation Class in *Factorial.java* :



```
FactorialInterface.java Yashuv.java Factorial.java X Baskota.java
Factorial.java > ...
1  public class Factorial implements FactorialInterface {
2      public int factorial(int n) {
3          int fact = 1;
4          for (int i = 1; i <= n; i++) {
5              fact = fact * i;
6          }
7          return fact;
8      }
9  }
10
```

C. Program for Server in *Baskota.java* :

```
1  import java.rmi.registry.Registry;
2  import java.rmi.registry.LocateRegistry;
3  import java.rmi.server.UnicastRemoteObject;
4
5  public class Baskota extends Factorial {
6      public Baskota() {
7      }
8      public static void main(String args[]) {
9          try {
10             Factorial obj = new Factorial();
11             FactorialInterface stub = (FactorialInterface) UnicastRemoteObject.exportObject(obj, port: 0);
12             Registry registry = LocateRegistry.getRegistry();
13             registry.bind(name: "FactorialInterface", stub);
14             System.err.println(x: "Server is ready...");
15         } catch (Exception e) {
16             System.err.println("Server exception: " + e.toString());
17             e.printStackTrace();
18         }
19     }
20 }
21
```

D. Program for Client in *Yashuv.java* :

```
1  import java.util.*;
2  import java.rmi.registry.LocateRegistry;
3  import java.rmi.registry.Registry;
4
5  public class Yashuv {
6      private Yashuv() {
7      }
8
9      public static void main(String[] args) {
10         Scanner in = new Scanner(System.in);
11         try {
12             Registry registry = LocateRegistry.getRegistry(host: null);
13             FactorialInterface stub = (FactorialInterface) registry.lookup(name: "FactorialInterface");
14
15             while (true) {
16                 System.out.print(s: "Enter nummber for factorial: ");
17                 int n = in.nextInt();
18                 System.out.println(stub.factorial(n));
19             }
20         } catch (Exception e) {
21             System.err.println("Client exception: " + e.toString());
22             e.printStackTrace();
23         } finally {
24             in.close();
25         }
26     }
27 }
28
```

OUTPUT FROM SERVER:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

Microsoft Windows [Version 10.0.19044.1766]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Yashuv\Desktop\Distributed System Assignment\RMI Java>javac *.java

C:\Users\Yashuv\Desktop\Distributed System Assignment\RMI Java>start rmiregistry

C:\Users\Yashuv\Desktop\Distributed System Assignment\RMI Java>java Baskota
Server is ready...
█
```

OUTPUT FROM THE CLIENT INTERFACE:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

Microsoft Windows [Version 10.0.19044.1766]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Yashuv\Desktop\Distributed System Assignment\RMI Java>java Yashuv
Enter nummber for factorial: 5
120
Enter nummber for factorial: 6
720
Enter nummber for factorial: 7
5040
Enter nummber for factorial: 10
3628800
Enter nummber for factorial: █
```