**Lab Exercise 7**

**JAVA RMI**

**Name: Ayush Sharma**

**Reg. No: 15BCE1335**

**Faculty: Prof. Gayatri. R**

**RMI Implementation:**

1. Create the remote interface

2. Provide the implementation of the remote interface

3. Create the server application

4. Create the client application

5. Compile the Java source files (above created 4 files).

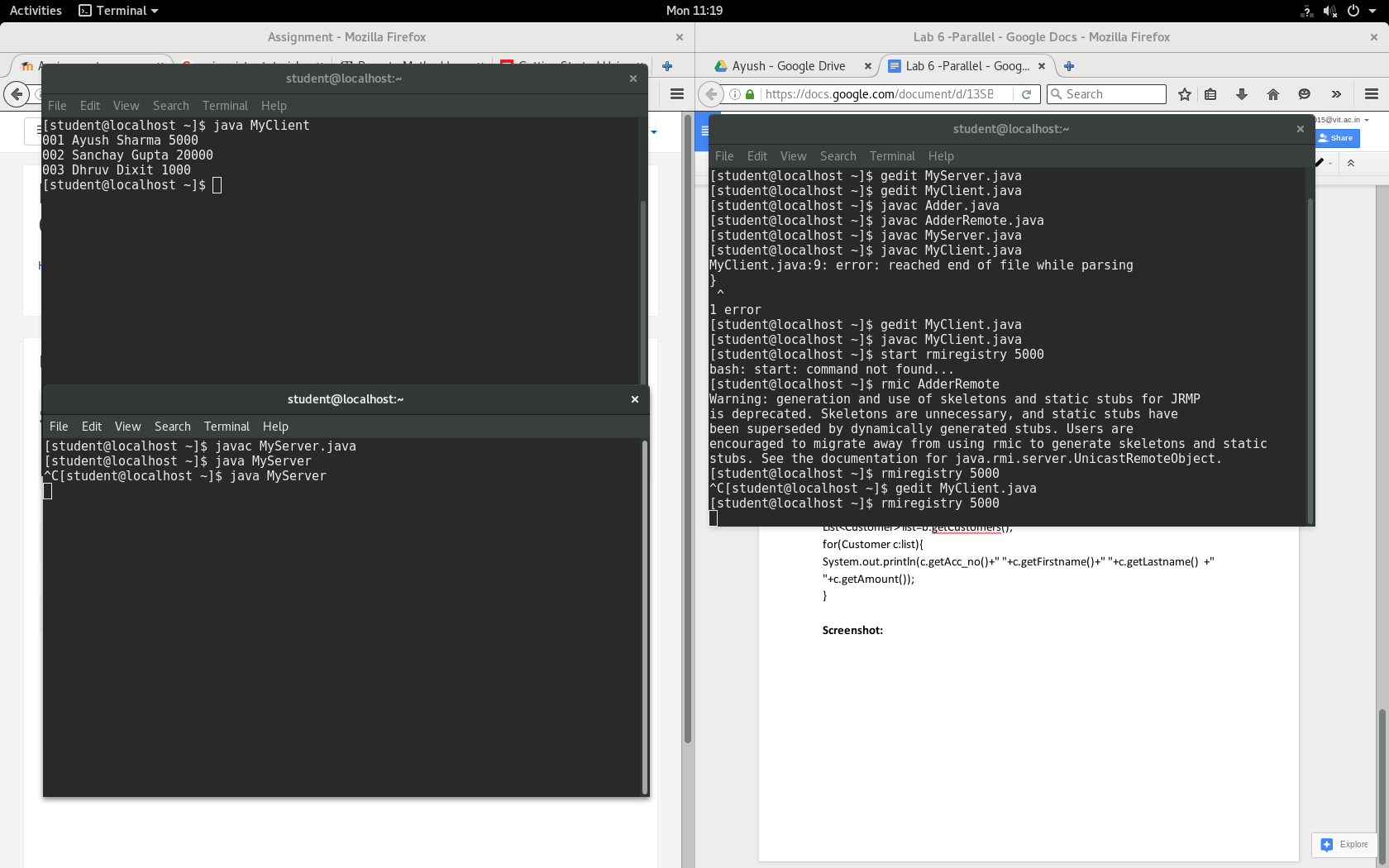
6. Generate the client stubs and server skeletons using the rmic tool.

7. Start the RMI registry (in one console).

8. Run the Server (in another console).

9. Run the Client (from any machine / if testing in same machine, then run in separate console)

**Screenshots:**



**Ques.1** Basic Calculator Implementation using Java RMI.

**Calculator.java**

public interface Calculator extends Remote

{

public int add(int x,int y)throws RemoteException;

public int sub(int x,int y)throws RemoteException;

public int mul(int x,int y)throws RemoteException;

public int div(int x,int y)throws RemoteException;

}

**CalculatorRemote.java**

public class CalculatorRemote extends UnicastRemoteObject implements Calculator

{

CalculatorRemote()throws RemoteException{

super();

}

public int add(int x,int y){ return x+y; }

public int sub(int x,int y){ return x-y; }

public int mul(int x,int y){ return x\*y; }

public int div(int x,int y){ return x/y; }

}

**MyServer.java**

public class MyServer{

public static void main(String args[]){

try{

Calculator stub=new CalculatorRemote();

Naming.rebind("rmi://localhost:5000/Calculator",stub);

}catch(Exception e){System.out.println(e);}

}

}

**MyClient.java**

public class MyClient{

public static void main(String args[]){

try{

Calculator stub=(Calculator)Naming.lookup("rmi://localhost:5000/Calculator");

System.out.println(stub.add(34,4));

System.out.println(stub.sub(34,4));

System.out.println(stub.mul(4,4));

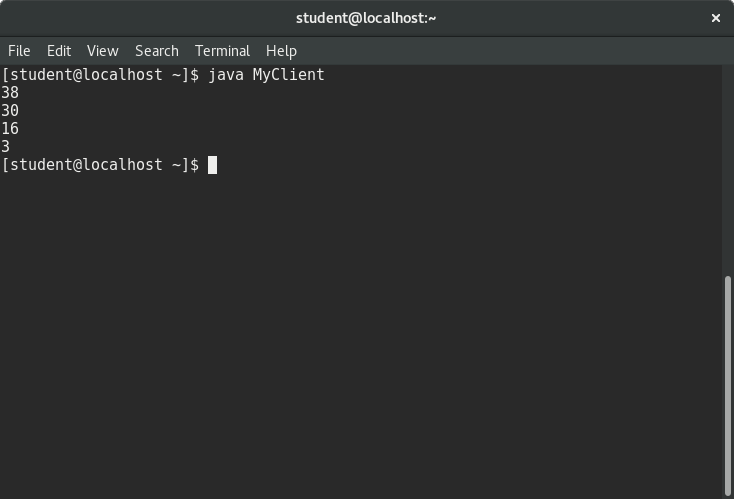
System.out.println(stub.div(6,4));

}catch(Exception e){}

}

}

**Screenshot:**



**Ques.2** Java RMI program for current bank balance

**Customers.java**

public class Customer implements java.io.Serializable{

private int acc\_no;

private String firstname,lastname;

private float amount;

}

**Bank.java**

interface Bank extends Remote{

public List<Customer> getCustomers()throws RemoteException; }

**Bank\_RMI.java**

class Bank\_RMI extends UnicastRemoteObject implements Bank{

Bank\_RMI()throws RemoteException{}

public List<Customer> getCustomers(){

List<Customer> list=new ArrayList<Customer>();

try{

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con= DriverManager.getConnection("jdbc:oracle:thin:@localhost:5000:xe","syste","oracle");

PreparedStatement ps=con.prepareStatement("select \* from customers");

ResultSet rs=ps.executeQuery();

while(rs.next()){

Customer c=new Customer();

c.setAcc\_no(rs.getInt(1));

c.setFirstname(rs.getString(2));

c.setLastname(rs.getString(3));

c.setAmount(rs.getFloat(4));

list.add(c);

}

con.close();

}catch(Exception e){System.out.println(e);}

return list;

}

}

**MyServer.java**

public class MyServer{

public static void main(String args[])throws Exception{

Remote r=new Bank\_RMI();

Naming.rebind("rmi://localhost:5000/Bank",r);

}}

**MyClient.java**

public class MyClient{

public static void main(String args[])throws Exception{

Bank b=(Bank)Naming.lookup("rmi://localhost:5000/Bank");

List<Customer> list=b.getCustomers();

for(Customer c:list){

System.out.println(c.getAcc\_no()+" "+c.getFirstname()+" "+c.getLastname() +" "+c.getAmount());

}

**Screenshot:**

