**LAB:NO:03**

**OBJECT: To learn RMI basics: Generate the stubs and skeletons and run client and server.**

**Task1: Create RMI program in which server performs basic arithmetic operations.**

**1)Interface on both client and server side.**

**import java.rmi.\*;**

**public interface rmiInterface extends Remote{**

**public double add(double a,double b)throws RemoteException;**

**public double div(double a,double b)throws RemoteException;**

**public double sub(double a,double b)throws RemoteException;**

**public double mul(double a,double b)throws RemoteException;**

**public double mod(double a,double b)throws RemoteException;**

**}**

**2)Server Remote object class:**

**import java.rmi.\*;**

**import java.rmi.server.\*;**

**public class rmiServerImpl extends UnicastRemoteObject implements rmiInterface{**

**public rmiServerImpl() throws RemoteException{**

**super();**

**}**

**public double add(double a,double b){**

**return(a+b);**

**}**

**public double sub(double a,double b){**

**return(a-b);**

**}**

**public double div(double a,double b){**

**return(a/b);**

**}**

**public double mul(double a,double b){**

**return(a\*b);**

**}**

**public double mod(double a,double b){**

**return(a%b);**

**}**

**}**

**3)Server main class:**

**import java.rmi.\*;**

**import java.rmi.server.\*;**

**import java.rmi.registry.\*;**

**public class rmiServer extends rmiServerImpl{**

**public rmiServer() throws RemoteException{**

**}**

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

**try**

**{**

**String ro="rmi://localhost:1099/arioperation";**

**rmiServerImpl p=new rmiServerImpl();**

**Naming.rebind(ro,p);**

**System.out.println("server is ready");**

**}**

**catch(Exception e){**

**System.out.println(e);**

**}**

**}**

**}**

**4)Client main class:**

**import java.rmi.\*;**

**import java.rmi.server.\*;**

**import java.rmi.registry.\*;**

**public class rmiClient{**

**static rmiInterface ari;**

**static String name="rmi://localhost/arioperation";**

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

**try{**

**int a=Integer.parseInt(args[0]);**

**int b=Integer.parseInt(args[1]);**

**ari=(rmiInterface)Naming.lookup(name);**

**System.out.println("first number:"+a);**

**System.out.println("second number:"+b);**

**System.out.println("add:\t"+ari.add(a,b));**

**System.out.println("mul:\t"+ari.mul(a,b));**

**System.out.println("sub:\t"+ari.sub(a,b));**

**System.out.println("div:\t"+ari.div(a,b));**

**System.out.println("mod:\t"+ari.mod(a,b));**

**}**

**catch(Exception e){**

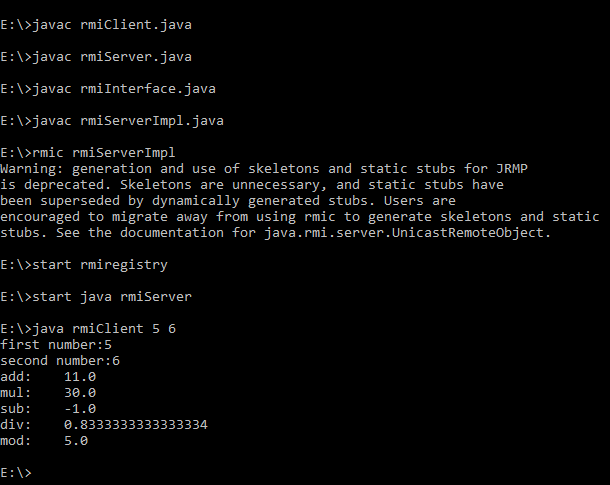
**System.out.println(e);**

**}**

**}**

**}**

**Output:**

****

**Task2:** **Create RMI program in which server sorts the numbers in ascending order.**

**1)Interface on both client and server side.**

**import java.rmi.\*;**

**import java.rmi.server.\*;**

**interface interfaceSort extends Remote {**

**int[] sort( int[] array ) throws RemoteException;**

**}**

**2)Server Remote object class:**

**import java.rmi.\*;**

**import java.rmi.server.\*;**

**class SortServer extends UnicastRemoteObject implements interfaceSort{**

**public SortServer() throws RemoteException { }**

**public int[] sort( int[] array ){**

**System.out.print( "SortServer.sort() - " );**

**for (int i=0; i < array.length; i++)**

**System.out.print( array[i] + " " );**

**System.out.println();**

**for (int i = 0; i < array.length; i++){**

**for (int j = i + 1; j < array.length; j++) {**

**if (array[i] > array[j]) {**

**int temp = array[i];**

**array[i] = array[j];**

**array[j] = temp;**

**}**

**}**

**}**

**return array;**

**}**

**}**

**3)Server main class:**

**import java.rmi.\*;**

**import java.rmi.server.\*;**

**public class RmiSortServer{**

**public static void main( String args[] ){**

**try {**

**Naming.rebind( "rmi://localhost/SortServer", new SortServer() );**

**}**

**catch(java.net.MalformedURLException ex) { ex.printStackTrace();}**

**catch (RemoteException ex) { ex.printStackTrace();}**

**System.out.println( "SortServer bound in rmiregistry" );**

**}**

**}**

**4)Client main class:**

**import java.rmi.\*;**

**public class SortClient{**

**public static void main( String args[] ){**

**int[] array = new int[15];**

**java.util.Random rn = new java.util.Random();**

**for (int i=0; i < array.length; i++) array[i] = rn.nextInt(20) + 1;**

**for (int i=0; i < array.length; i++) System.out.print( array[i] + " " );**

**System.out.println();**

**try {**

**interfaceSort sorter = (interfaceSort) Naming.lookup( "rmi://localhost/SortServer" );**

**array = sorter.sort( array );**

**} catch (NotBoundException ex) { ex.printStackTrace();}**

**catch (java.net.MalformedURLException ex) { ex.printStackTrace();}**

**catch (RemoteException ex) { ex.printStackTrace(); }**

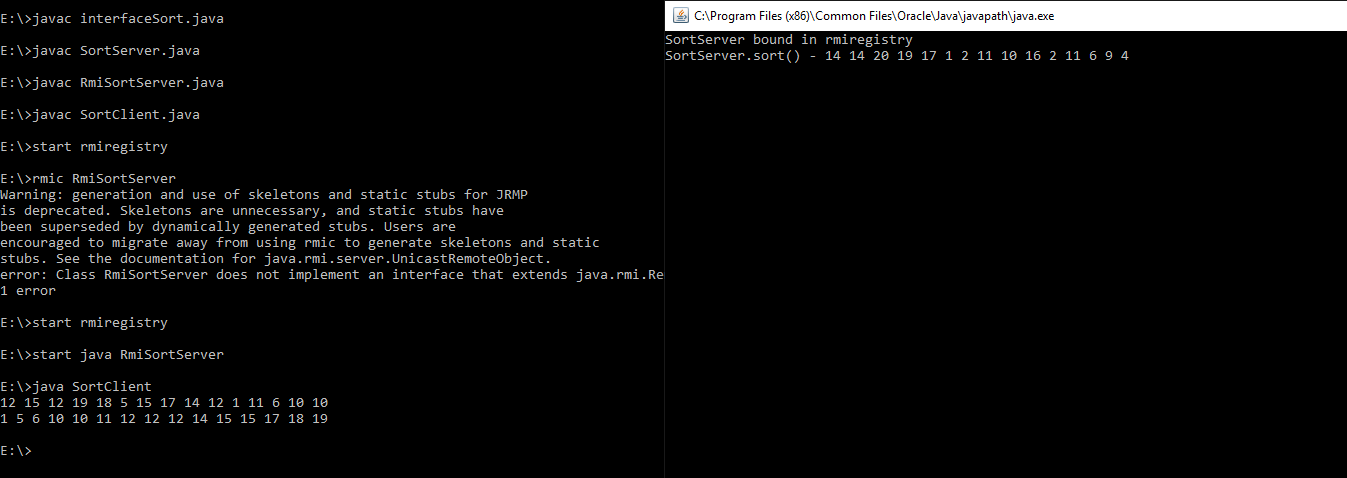
**for (int i=0; i < array.length; i++) System.out.print( array[i] + " " );**

**System.out.println();**

**}**

**}**

**Output:**

****

**Task3:(Optional Task) Write an RMI program to implement the Map/Reduce concept of Hadoop on the server program. In other words the client program asks user to provide multiple sentences and send to server, server sends back the word counts.**

**1)Interface on both client and server side.**

**import java.rmi.\*;**

**public interface wordCountInterface extends Remote {**

**public String wordCount(String data) throws RemoteException;**

**}**

**2)Server Remote object class:**

**import java.rmi.\*;**

**import java.rmi.server.\*;**

**public class wordCountImpl extends UnicastRemoteObject implements wordCountInterface {**

**public wordCountImpl() throws RemoteException {**

**super();**

**}**

**public String wordCount(String data) {**

**String print="";**

**String[] keys = data.split(" ");**

**String[] uniqueKeys;**

**int count = 0;**

**System.out.println(data);**

**uniqueKeys = getUniqueKeys(keys);**

**for(String key: uniqueKeys)**

**{**

**if(null == key) {**

**break;**

**}**

**for(String s : keys){**

**if(key.equals(s)) {**

**count++;**

**}**

**}**

**System.out.println("Count of ["+key+"] is : "+count);**

**print+="Count of ["+key+"] is : ["+count+"]\n";**

**count=0;**

**}**

**return print;**

**}**

**public static String[] getUniqueKeys(String[] keys){**

**String[] uniqueKeys = new String[keys.length];**

**uniqueKeys[0] = keys[0];**

**int uniqueKeyIndex = 1;**

**boolean keyAlreadyExists = false;**

**for(int i=1; i<keys.length ; i++){**

**for(int j=0; j<=uniqueKeyIndex; j++) {**

**if(keys[i].equals(uniqueKeys[j])){**

**keyAlreadyExists = true;**

**}**

**} if(! keyAlreadyExists) {**

**uniqueKeys[uniqueKeyIndex] = keys[i];**

**uniqueKeyIndex++;**

**}keyAlreadyExists = false;**

**} return uniqueKeys;**

**}**

**}**

**3)Server main class:**

**import java.rmi.\*;**

**import java.rmi.server.\*;**

**import java.rmi.registry.\*;**

**public class wordCountServer extends wordCountImpl{**

**public wordCountServer() throws RemoteException{}**

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

**try{**

**String ro = "rmi://localhost:1099/wordCount";**

**wordCountImpl p = new wordCountImpl();**

**Naming.rebind(ro,p);**

**System.out.println("Server is ready!");**

**}catch ( Exception e){**

**System.out.println(e);}**

**}**

**}**

**4)Client main class:**

**import java.rmi.\*;**

**import java.rmi.server.\*;**

**public class wordCountClient{**

**static wordCountInterface wc;**

**private static String data;**

**static String name="rmi://localhost/wordCount";**

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

**try{**

**if (args.length > 0){**

**System.out.println("Send some sentence to server:");**

**for (String value:args)**

**data += value;**

**System.out.println(data);**

**}**

**wc=(wordCountInterface)Naming.lookup(name);**

**String result = wc.wordCount(data);**

**System.out.println("Count:" + result);**

**}**

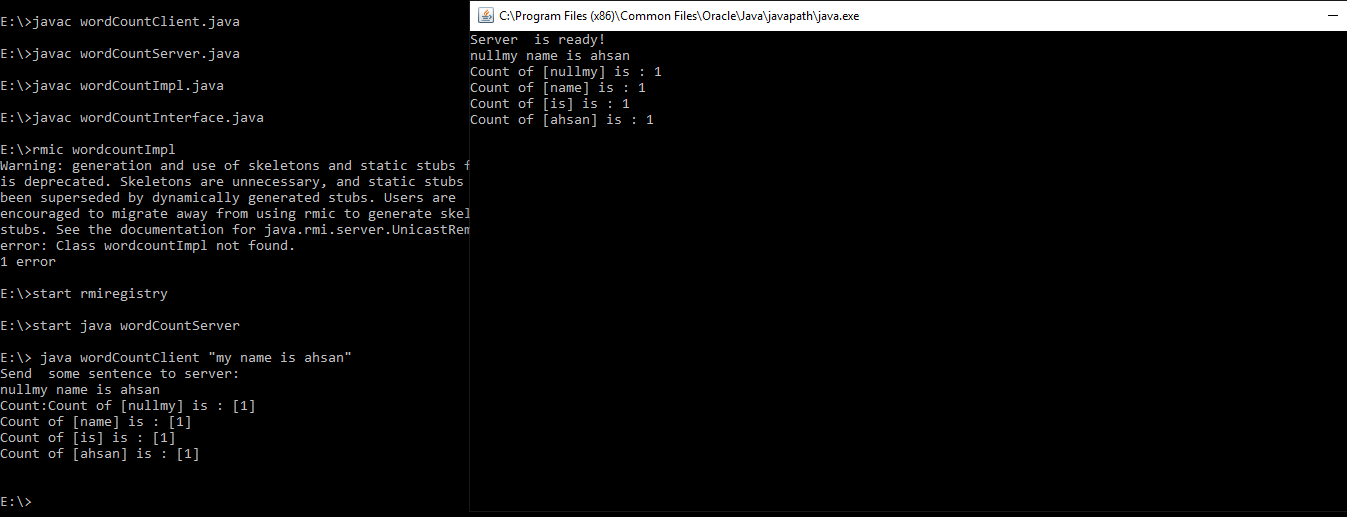
**catch ( Exception e){**

**System.out.println(e);}**

**}**

**}**

**Output:**

****