Lab # 03

OBJECTIVE: To learn RMI basics: Generate the stubs and skeletons and run client and server

Task 1: Create RMI program in which server performs basic arithmetic operations.

Interface class

```
import java.rmi.*;
public interface CalculationsInterface extends Remote{
 public String calculate(int var1,int var2,char operator)
 throws RemoteException;
}
```

Calculation class

```
import java.rmi.*;
import java.rmi.server.*;
public class Calculations extends UnicastRemoteObject implements Calculations
public Calculations() throws RemoteException{
System.out.println("Object created");
}
public String calculate(int var1,int var2,char operator)
throws RemoteException{
switch (operator){
case '+':
return "Addition is : "+ (var1+var2);
case '-':
return  "Subtraction is : "+(var1-var2);
case '*':
return "Multiplication is : "+(var1*var2);
case '/':
return "Division is : "+((float)var1/var2);
default:
return "Invalid Input types";}}}
```

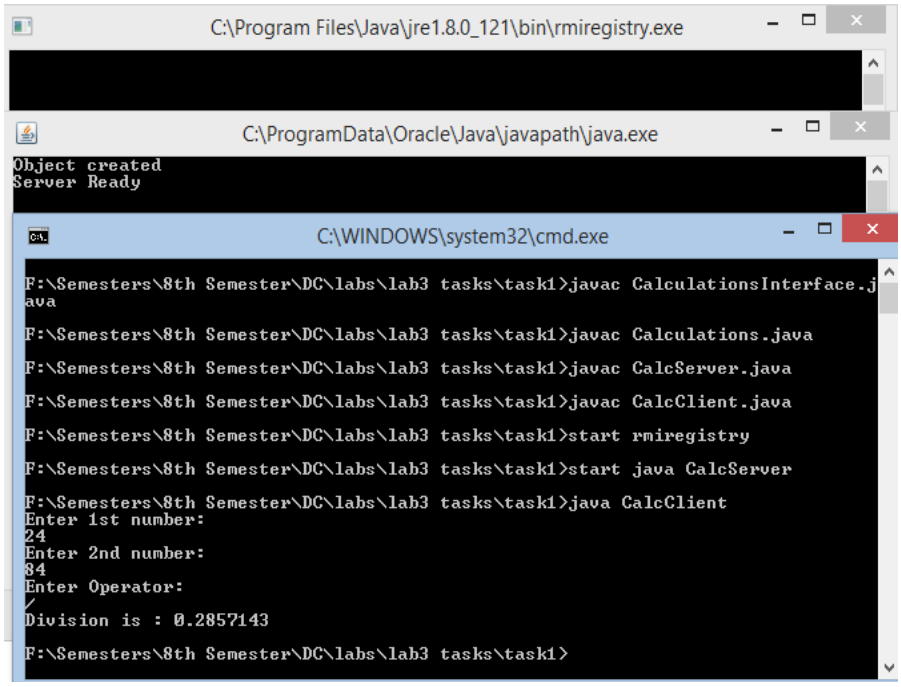
Server Class

```
import java.rmi.*;
public class CalcServer {
public static void main(String args[]){
try{
Naming.rebind("rmi://localhost/CalcServer",new Calculations());
System.out.println("Server Ready");
}catch(Exception e){
System.out.println("Server failed");
}}}
```

Client Class:

```
import java.rmi.*;
import java.util.Scanner;
class CalcClient{
public static void main(String args[]){
CalculationsInterface calculations;
String serverName="rmi://localhost/CalcServer";
Scanner input=new Scanner(System.in);
System.out.println("Enter 1st number:");
int num1=input.nextInt();
System.out.println("Enter 2nd number:");
int num2=input.nextInt();
System.out.println("Enter Operator:");
char operator=input.next().charAt(0);
try{
calculations=(CalculationsInterface)Naming.lookup(serverName);
System.out.println(calculations.calculate(num1,num2,operator));
}catch(Exception e){
System.out.println("Exception Ocurred on client side "+e);
}}}
```

Output:



Task 2: Create RMI program in which server sorts the numbers in ascending order.

Interface class:

```
import java.rmi.*;
public interface SortingInterface extends Remote{
public String sorting(int[] numbers)
 throws RemoteException;
}
```

Sorting class:

```
public class Sorting extends UnicastRemoteObject implements SortingInterface{
public Sorting() throws Exception{
System.out.println("Server side object created");
}
public String sorting(int[] numbers)
throws RemoteException{
    int size=numbers.length;
    int hold=0;
    String str="";
    for(int i=1;i<size;i++){
        for(int j=0;j<size-i;j++){
            if(numbers[j]>numbers[j+1]){
                hold=numbers[j];
                numbers[j]=numbers[j+1];
                numbers[j+1]=hold;
                }}
            str+="\n"+numbers[i-1];
            }            return str; }}
```

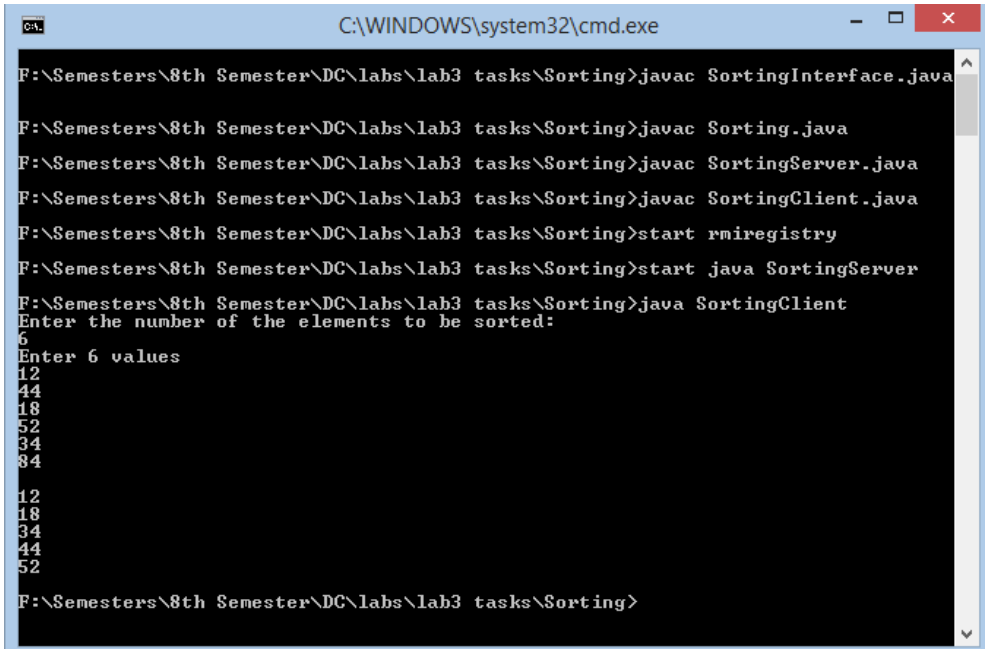
Server class:

```
import java.rmi.*;
public class SortingServer{
public static void main(String args[]){
try{
Naming.rebind("rmi://localhost/SortingServer",new Sorting());
System.out.println("Server ready");
}catch(Exception e){
System.out.println("Server Failed");
}}}
```

Client Class:

```
import java.rmi.*;
import java.util.Scanner;
class SortingClient{
public static void main(String args[]){
SortingInterface sorting;
String serverName="rmi://localhost/SortingServer";
Scanner input=new Scanner(System.in);
System.out.println("Enter the number of the elements to be sorted: ");
int length=input.nextInt();
int array[]=new int[length];
System.out.println("Enter "+length+" values");
for(int i=0;i<length;i++){
array[i]=input.nextInt();
}
try{
sorting=(SortingInterface)Naming.lookup(serverName);
System.out.println(sorting.sorting(array));
}catch(Exception e){
System.out.println("Exception Occured"+e);
}}}
```

Output:



Task 3: 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.

Interface class:

```
import java.rmi.*;
public interface WordCountInterface extends Remote{
public int countingWords(String sentence) throws RemoteException;
}
```

Wordcount class:

```
import java.rmi.*;
import java.rmi.server.*;
public class WordCount extends UnicastRemoteObject implements WordCountInte
int words=0;
char ch=' ';
public WordCount() throws RemoteException
{
}
public int countingWords(String sentence) throws RemoteException{
for(int i=0;i<sentence.length();i++){
ch=sentence.charAt(i);
if(ch==' ')
words++;
System.out.println(""+ch);}
return (words+1);
}}
```

Server class:

```
import java.rmi.*;
public class WordCountServer{
public static void main(String args[]){
try{
Naming.rebind("rmi://localhost/WordCountServer",new WordCount());
System.out.println("Server ready");
}catch(Exception e){
System.out.println("Server failed "+e);
}}}
```

Client class:

```
import java.rmi.*;
import java.util.Scanner;
class WordCountClient{
public static void main(String args[]){
WordCountInterface wordcount;
String serverName="rmi://localhost/WordCountServer";
Scanner input=new Scanner(System.in);
System.out.println("Enter your sentnce here: ");
String sentence=input.nextLine();
try{
wordcount=(WordCountInterface)Naming.lookup(serverName);
System.out.println(""+wordcount.countingWords(sentence));
}catch(Exception e){
System.out.println("Exception occured in WordCountClient "+e);
}}}
```

Output: