Department of Software Engineering Mehran University of Engineering and Technology, Jamshoro			
Course: SW422–Distributed Computing			
Instructor	RabeeaJaffari	**Practical/Lab No.**	03
Date		**CLOs**	CLO-3: P5 & CLO-3: P3
Signature		**Assessment Score**	

Topic	**To work with Remote Method Invocation (RMI) API**
Objectives	- Learn RMI basics and generating stubs and skeletons and running client and server

Lab Discussion: Theoretical concepts and Procedural steps

Lab Tasks

Submission Date:

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

ArithmeticInterface.java | Server.java | Client.java

```
1 import java.rmi.*;
2 public interface ArithmeticInterface extends Remote{
3     //16SW04 JAWARIA SATTAR
4     public double Addition(double a, double b) throws RemoteException;
5     public double Subtraction(double a, double b) throws RemoteException;
6     public double Multiplication(double a, double b) throws RemoteException;
7     public double Division(double a, double b) throws RemoteException;
8
9 }
```

Arithmetic.java | Server.java | Client.java

```
1  import java.rmi.*;
2  import java.rmi.server.*;
3  public class Arithmetic extends UnicastRemoteObject implements ArithmeticInterface{
4     public Arithmetic() throws RemoteException{
5         super();
6     }    //16SW04 JAWARIA SATTAR
7 public double Addition(double a, double b){
8     return(a+b);
9
10 }
11 public double Subtraction(double a, double b){
12     return(a-b);
13
14 }
15
16 public double Multiplication(double a, double b){
17     return(a*b);
18
19 }
20
21 public double Division(double a, double b){
22     return(a/b);
23
24 }
25
26
27  }
```

Arithmetic.java | Server.java | Client.java

```java
import java.rmi.*;
import java.rmi.server.*;
import java.rmi.registry.*;
public class Server extends Arithmetic {
    public Server() throws RemoteException{

    }
//16SW04 JAWARIA SATTAR
    public static void main(String args[]) throws RemoteException{

try{

    String ro="rmi://localhost:1099/Jawaria";
    Arithmetic p=new Arithmetic();
    Naming.rebind(ro,p);
    System.out.println("Server is ready to perform Arithmetic Operations!");
}

catch(Exception e)
{

System.out.println(e);

}
    }

 }

```

Arithmetic.java | Server.java | Client.java

```
1  import java.rmi.*;
2  import java.rmi.server.*;
3  import java.rmi.registry.*;
4  public class Client{
5
6      static ArithmeticInterface AriInt;
7      static String name="rmi://localhost:1099/Jawaria";
8
9
10 //16SW04 JAWARIA SATTAR
11     public static void main(String args[]) throws RemoteException{
12
13 try{
14     int a=Integer.parseInt(args[0]);
15     int b=Integer.parseInt(args[1]);
16
17     AriInt=(ArithmeticInterface)Naming.lookup(name);
18     System.out.println("First Integer: "+a);
19     System.out.println("Second Integer: "+b);
20     System.out.println("16SW04 (JAWARIA) (SATTAR)");
21     System.out.println("Addition: "+AriInt.Addition(a,b));
22     System.out.println("Subtraction: "+AriInt.Subtraction(a,b));
23     System.out.println("Multiplication: "+AriInt.Multiplication(a,b));
24     System.out.println("Division: "+AriInt.Division(a,b));
25
26 }
27
28 catch(Exception e)
29 {
30
31 System.out.println(e);
32
33 }
34     }
35
36  }
37
```

Command Prompt

```
E:\8th Semester\DC Labs\Lab 3\16SW04\Task1>javac *.java

E:\8th Semester\DC Labs\Lab 3\16SW04\Task1>rmic Arithmetic
Warning: generation and use of skeletons and static stubs for JRMP
is deprecated. Skeletons are unnecessary, and static stubs have
been superseded by dynamically generated stubs. Users are
encouraged to migrate away from using rmic to generate skeletons and static
stubs. See the documentation for java.rmi.server.UnicastRemoteObject.

E:\8th Semester\DC Labs\Lab 3\16SW04\Task1>start rmiregistry

E:\8th Semester\DC Labs\Lab 3\16SW04\Task1>start java Server

E:\8th Semester\DC Labs\Lab 3\16SW04\Task1>java Client 0 4
First Integer: 0
Second Integer: 4
16SW04 (JAWARIA) (SATTAR)
Addition: 4.0
Subtraction: -4.0
Multiplication: 0.0
Division: 0.0

E:\8th Semester\DC Labs\Lab 3\16SW04\Task1>
```

C:\Program Files\Java\jdk-12.0.1\bin\java.exe

```
Server is ready to perform Arithmetic Operations!
```

C:\Program Files\Java\jdk-12.0.1\bin\rmiregistry.exe

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

SortingInterface.java | Server.java | Client.java

```
1 import java.rmi.*;
2 public interface SortingInterface extends Remote{
3     //16SW04 JAWARIA SATTAR
4     public void SortNumber() throws RemoteException;
5
6
7 }
```

Sorting.java | Server.java | Client.java

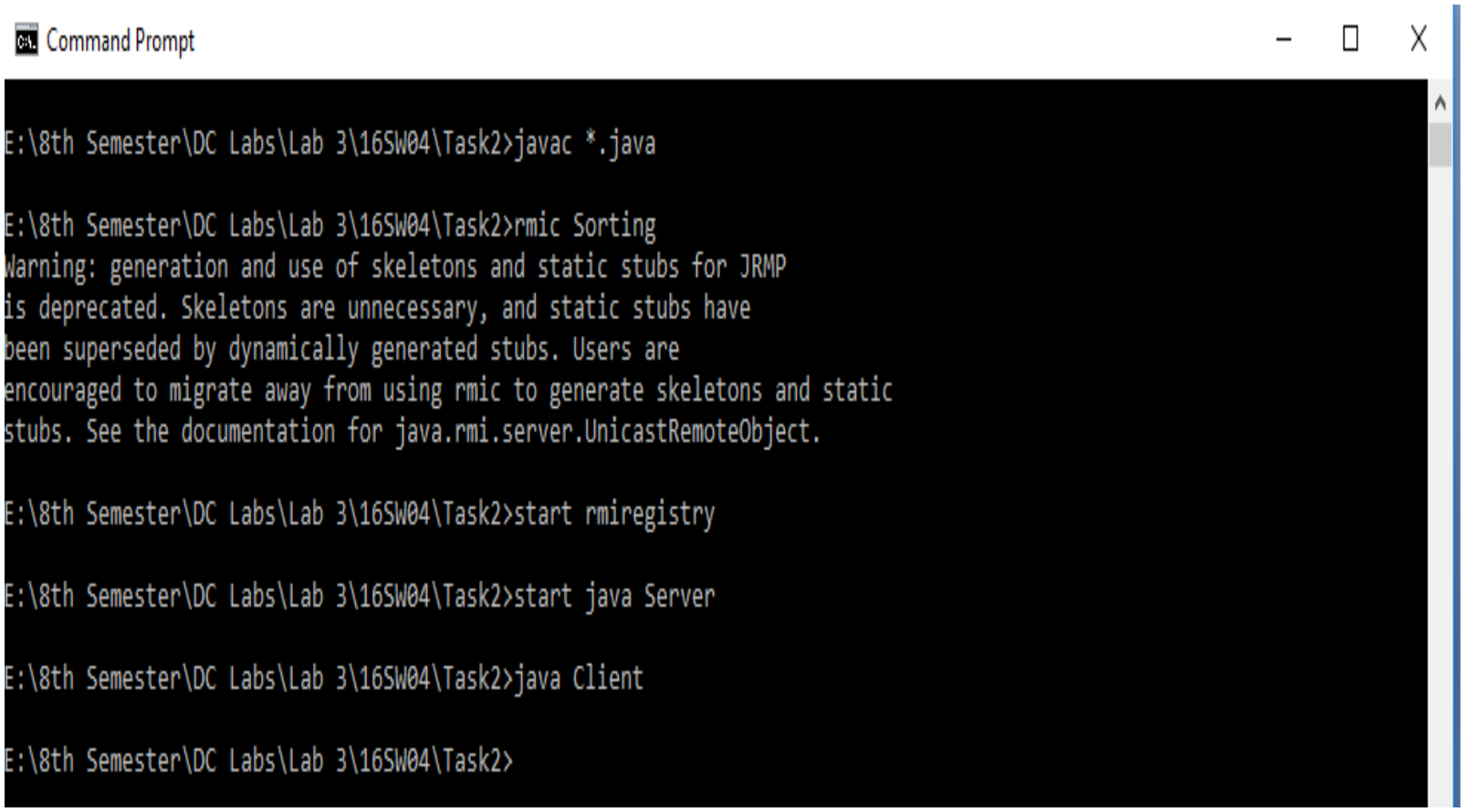
```
1  import java.rmi.*;
2  import java.rmi.server.*;
3  import java.util.Arrays;
4  public class Sorting extends UnicastRemoteObject implements SortingInterface{
5     public Sorting() throws RemoteException{
6         super();
7     }    //16SW04 JAWARIA SATTAR
8 public void  SortNumber() throws RemoteException{
9     int[] arr={26, 30, 90, 22, 04};
10    Arrays.sort(arr);
11    System.out.print("Sorted Array:");
12    System.out.print(Arrays.toString(arr));
13
14 }
15
16
17  }
```

Sorting.java | Server.java | Client.

```
1  import java.rmi.*;
2  import java.rmi.server.*;
3  import java.rmi.registry.*;
4  public class Server  {
5      
6  //16SW04 JAWARIA SATTAR
7      public static void main(String args[]) {
8  
9  try{
10 
11     String ro="rmi://localhost:1099/Jawaria";
12     Sorting p=new Sorting();
13     Naming.rebind(ro,p);
14     System.out.println("Server is ready to Sort Numbers!");
15 }
16 
17 catch(Exception e)
18 {
19 
20 System.out.println(e);
21 
22 }
23     }
24 
25  }
26 
```

Sorting.java | Client.java | Server.java

```
1  import java.rmi.*;
2  import java.rmi.server.*;
3  import java.rmi.registry.*;
4  public class Client{
5
6     static SortingInterface SortNum;
7     static String name="rmi://localhost:1099/Jawaria";
8
9
10 //16SW04 JAWARIA SATTAR
11    public static void main(String args[]) throws RemoteException{
12
13 try{
14
15
16    SortNum=(SortingInterface)Naming.lookup(name);
17    SortNum.SortNumber();
18 }
19
20 catch(Exception e)
21 {
22
23 System.out.println(e);
24
25 }
26    }
27
28  }
29
```


Command Prompt
E:\8th Semester\DC Labs\Lab 3\16SW04\Task2>javac *.java
E:\8th Semester\DC Labs\Lab 3\16SW04\Task2>rmic Sorting
Warning: generation and use of skeletons and static stubs for JRMP
is deprecated. Skeletons are unnecessary, and static stubs have
been superseded by dynamically generated stubs. Users are
encouraged to migrate away from using rmic to generate skeletons and static
stubs. See the documentation for java.rmi.server.UnicastRemoteObject.
E:\8th Semester\DC Labs\Lab 3\16SW04\Task2>start rmiregistry
E:\8th Semester\DC Labs\Lab 3\16SW04\Task2>start java Server
E:\8th Semester\DC Labs\Lab 3\16SW04\Task2>java Client
E:\8th Semester\DC Labs\Lab 3\16SW04\Task2>


C:\Program Files\Java\jdk-12.0.1\bin\java.exe
Server is ready to Sort Numbers!
Sorted Array:[4, 22, 26, 30, 90]


C:\Program Files\Java\jdk-12.0.1\bin\rmiregistry.exe

3. **(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.

wordCountImpl.java | wordCountInterface.java | wordCountServer.java

```
1  import java.rmi.*;
2
3   public interface wordCountInterface extends Remote
4   {      //Jawaria Sattar 16SW04
5        public String wordCount(String data) throws RemoteException;
6
7   }
```

wordCountImpl.java | wordCountInterface.java | wordCountServer.java

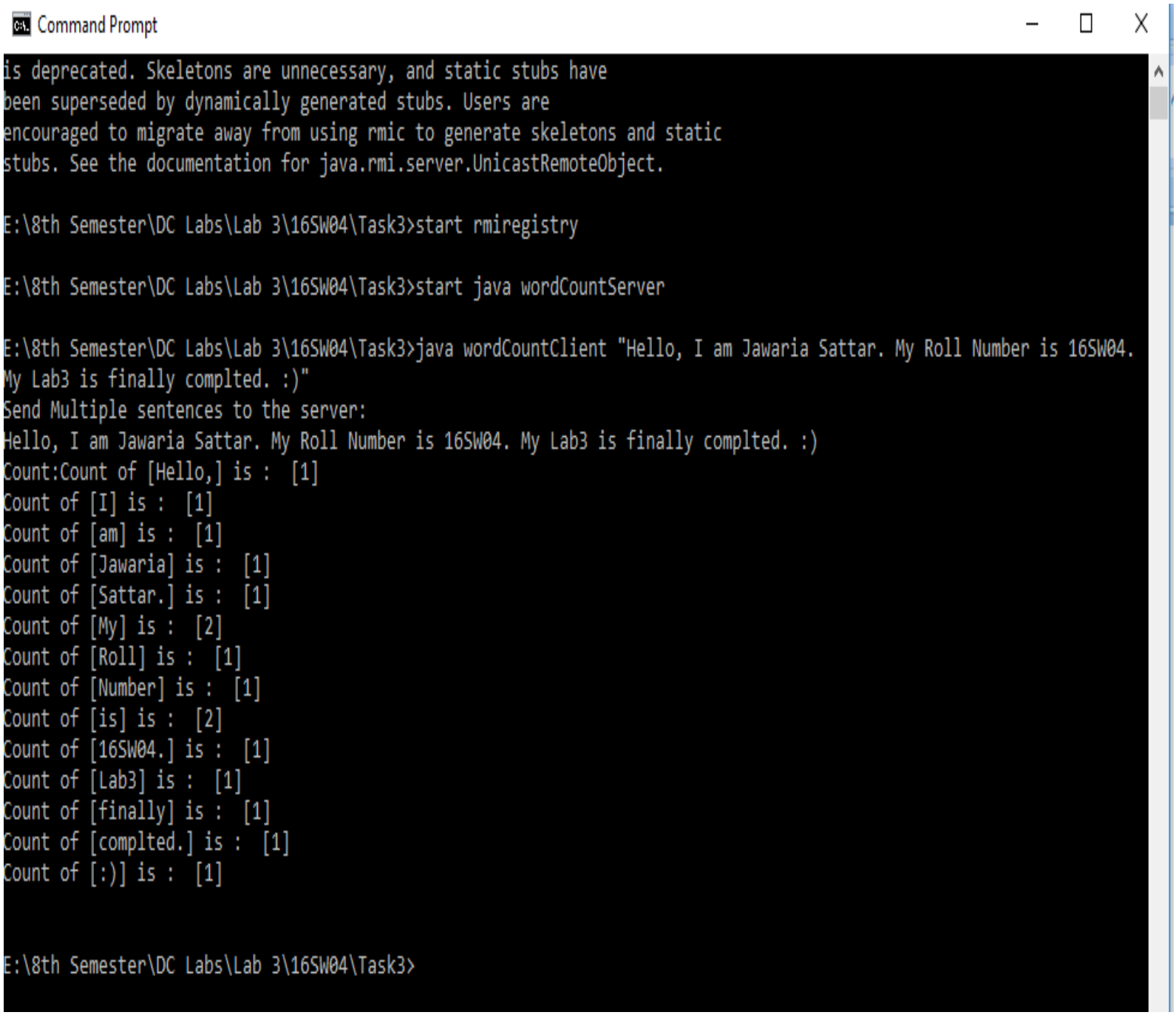
```
40        {
41             String[] uniqueKeys = new String[keys.length];
42             uniqueKeys[0] = keys[0];
43             int uniqueKeyIndex = 1;
44             boolean keyAlreadyExists = false;
45             for(int i=1; i<keys.length ; i++)
46             {   //Jawaria Sattar 16SW04
47                 for(int j=0; j<=uniqueKeyIndex; j++)
48                 {
49                     if(keys[i].equals(uniqueKeys[j]))
50                     {
51                         keyAlreadyExists = true;
52                     }
53                 }
54                 if(!keyAlreadyExists)
55                 {
56                     uniqueKeys[uniqueKeyIndex] = keys[i];
57                     uniqueKeyIndex++;
58                 }
59                 keyAlreadyExists = false;
60             }
61             return uniqueKeys;
62     }
63 }
```

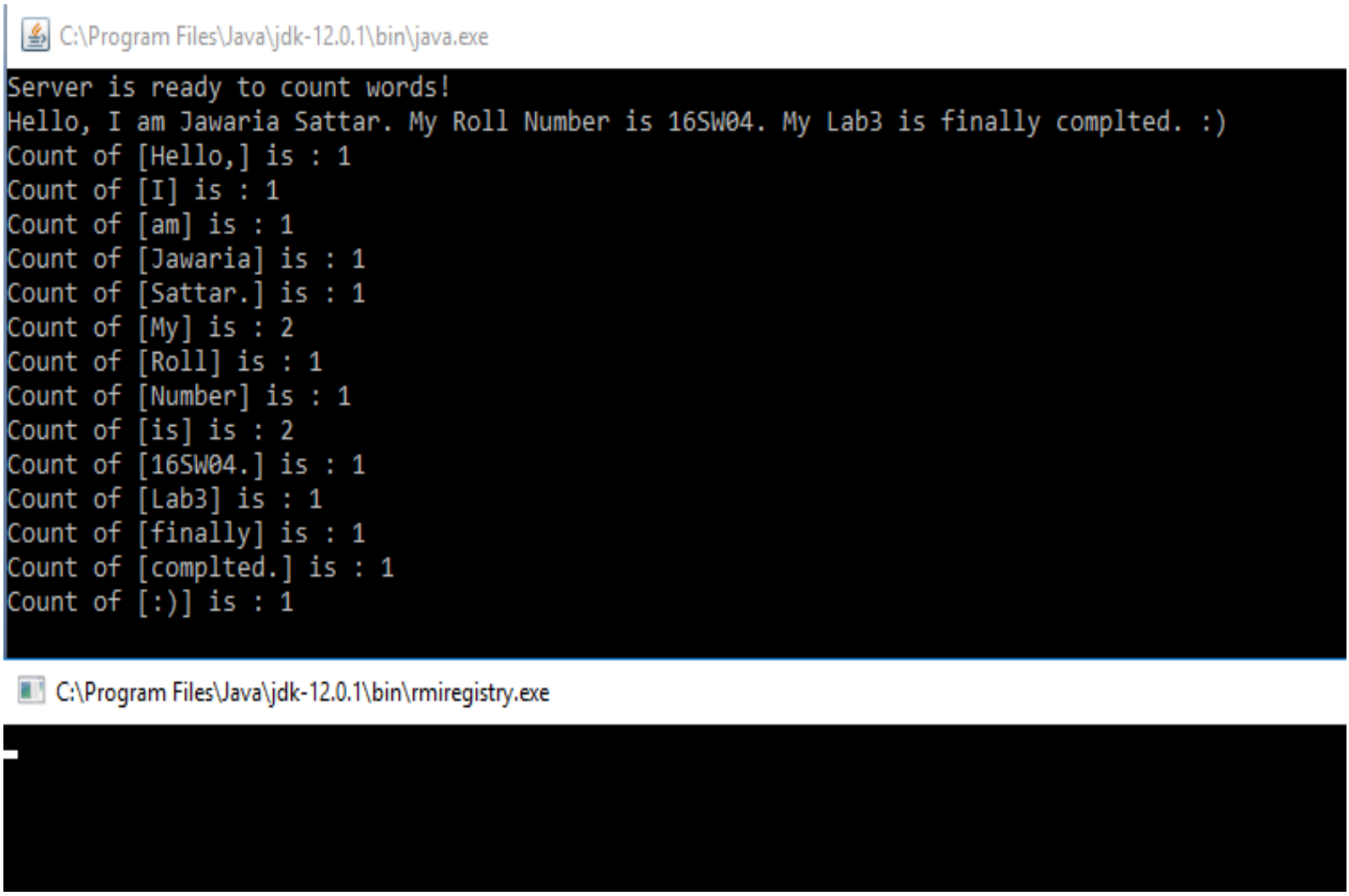
wordCountClient.java | wordCountServer.java | wordCountInterface.java

```java
import java.rmi.*;
 import java.rmi.server.*;
 import java.rmi.registry.*;

 public class wordCountServer extends wordCountImpl
 {
      public wordCountServer() throws RemoteException
      {

      }     //Jawaria Sattar 16SW04

      public static void main(String args[]) throws RemoteException
      {
           try
           {
               String ro = "rmi://localhost:1099/Jawaria";
    wordCountImpl p = new wordCountImpl();
    Naming.rebind(ro,p);
                System.out.println("Server is ready to count words!");
           }
           catch(Exception e)
           {
                System.out.println(e);
           }
      }
 }
```

wordCountClient.java | wordCountServer.java | wordCountInterface.java

```
1  import java.rmi.*;
2   import java.rmi.server.*;
3
4   public class wordCountClient
5   {
6       static wordCountInterface wc;
7       private static String data="";
8       static String name="rmi://localhost/Jawaria";
9        public static void main(String args[])throws RemoteException
10       {   //Jawaria Sattar 16SW04
11            try
12            {
13                 if (args.length > 0)
14                 {
15                 System.out.println("Send Multiple sentences to the server:");
16                       for (String value:args)
17                        data += value;
18                        System.out.println(data);
19
20                              }
21
22                 wc=(wordCountInterface)Naming.lookup(name);
23                 String result = wc.wordCount(data);
24                 System.out.println("Count:"  + result);
25
26
27            }
28            catch(Exception e)
29            {
30                 System.out.println(e);
31            }
32       }
33  }
```


Command Prompt
is deprecated. Skeletons are unnecessary, and static stubs have
been superseded by dynamically generated stubs. Users are
encouraged to migrate away from using rmic to generate skeletons and static
stubs. See the documentation for java.rmi.server.UnicastRemoteObject.
E:\8th Semester\DC Labs\Lab 3\16SW04\Task3>start rmiregistry
E:\8th Semester\DC Labs\Lab 3\16SW04\Task3>start java wordCountServer
E:\8th Semester\DC Labs\Lab 3\16SW04\Task3>java wordCountClient "Hello, I am Jawaria Sattar. My Roll Number is 16SW04.
My Lab3 is finally complted. :)"
Send Multiple sentences to the server:
Hello, I am Jawaria Sattar. My Roll Number is 16SW04. My Lab3 is finally complted. :)
Count:Count of [Hello,] is : [1]
Count of [I] is : [1]
Count of [am] is : [1]
Count of [Jawaria] is : [1]
Count of [Sattar.] is : [1]
Count of [My] is : [2]
Count of [Roll] is : [1]
Count of [Number] is : [1]
Count of [is] is : [2]
Count of [16SW04.] is : [1]
Count of [Lab3] is : [1]
Count of [finally] is : [1]
Count of [complted.] is : [1]
Count of [:)] is : [1]
E:\8th Semester\DC Labs\Lab 3\16SW04\Task3>


C:\Program Files\Java\jdk-12.0.1\bin\java.exe
Server is ready to count words!
Hello, I am Jawaria Sattar. My Roll Number is 16SW04. My Lab3 is finally complted. :)
Count of [Hello,] is : 1
Count of [I] is : 1
Count of [am] is : 1
Count of [Jawaria] is : 1
Count of [Sattar.] is : 1
Count of [My] is : 2
Count of [Roll] is : 1
Count of [Number] is : 1
Count of [is] is : 2
Count of [16SW04.] is : 1
Count of [Lab3] is : 1
Count of [finally] is : 1
Count of [complted.] is : 1
Count of [:)] is : 1
C:\Program Files\Java\jdk-12.0.1\bin\rmiregistry.exe