

Unidade Curricular:

Integração de Sistemas de Informação

Tema da Ficha Prática:

Utilização de RMI – Remote Method Invocation

Objectivos:

Pretende-se com esta ficha prática que os alunos interajam com o conceito de RMI – Remote Method Invocation

Bibliografia:

Para apoio a esta ficha os alunos devem consultar os apontamentos teóricos e práticos da disciplina bem como de outros recursos online.

Índice

1. Utilização do conceito de Remote Method Invocation usando a linguagem de programação JAVA	2
1.1 Primeiro Exemplo	3
1.1.1 Mensageiro.java	3
1.1.2 MensageiroImpl.java	4
1.1.3 MensageiroServidor.java	4
1.1.4 MensageiroCliente.java	5
1.1.3 Execução	5
1.2 Exercício 2	6
1.2.1 Calculator.java	6
1.2.2 CalculatorServer.java	6
1.2.3 CalculatorImpl.java	7
1.2.4 CalculatorClient.java	7
1.2.4 Execução	8
1.3 Contactos	9
1.3.1 RMIArrayList.java	9
1.3.2 RMIArrayListImpl.java	9
1.3.3 RMIArrayListClient.java	10
1.3.4 Execução	11

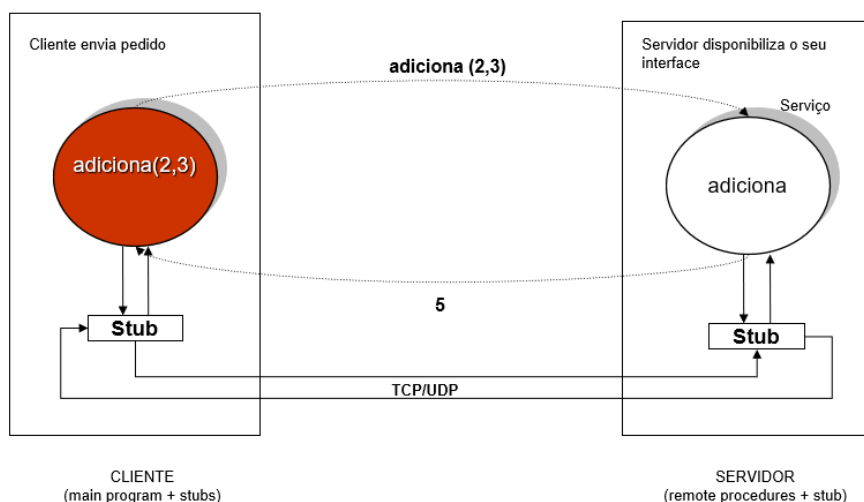
1. Utilização do conceito de Remote Method Invocation usando a linguagem de programação JAVA

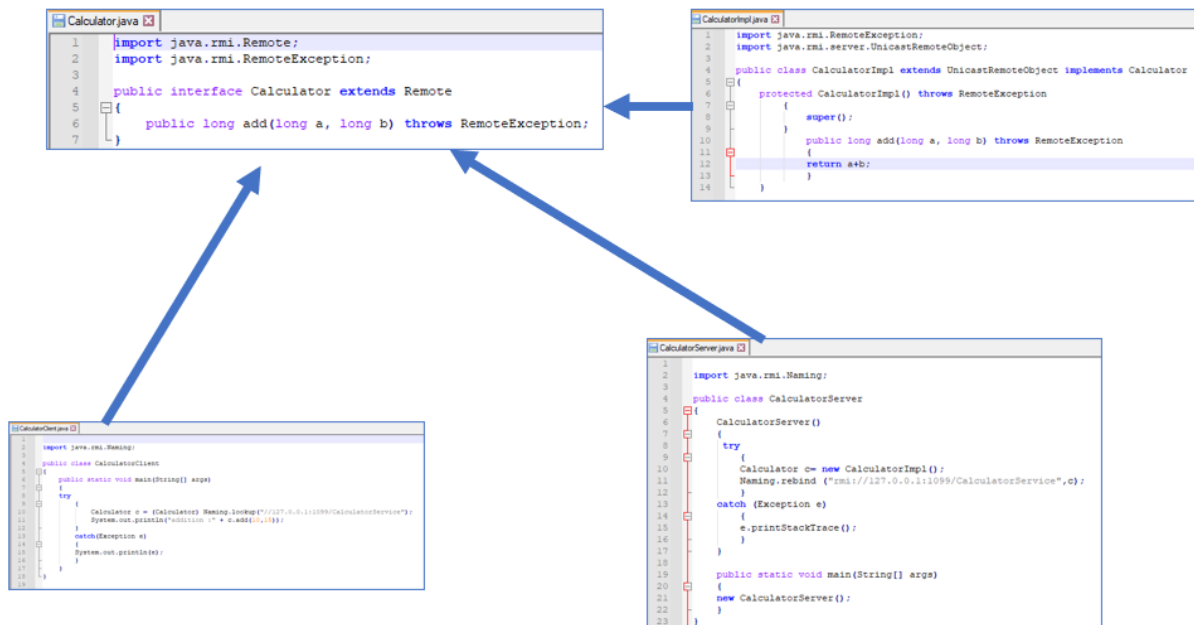
Em termos de resumo os passos que deve seguir para desenvolver um sistema RMI é o seguinte:

1. Criar o interface remoto
2. Criar a classe que implementa o interface
3. Criar o programa cliente
4. Compilar o código java
`javac *.java`
5. Gerar os stub (cliente) e skeleton (servidor) (rmi interface compiler – rmic)
`rmic MensageiroImpl`
6. Iniciar o registo RMI (RMI registry)
`start rmiregistry`
7. Iniciar o servidor
`java MensageiroServidor`
8. Executar o cliente
`java MensageiroCliente`

Exemplo ilustrativo:

Middleware baseado em Remote Procedures:





•Principais passos:

1. Client program: evoca o procedimento remoto (localmente)
2. Client stub: codifica e envia a mensagem ao server
3. Server stub: recebe e decodifica a mensagem pedido
4. Server stub: evoca o procedimento (real)
5. Server stub: codifica e envia o resultado ao cliente
6. Client stub: recebe e decodifica a mensagem resultado
7. Client stub: devolve o resultado ao cliente

1.1 Primeiro Exemplo

1.1.1 Mensageiro.java

```

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface Mensageiro extends Remote {

    public void enviarMensagem( String msg ) throws RemoteException;

    public String lerMensagem() throws RemoteException;

}
  
```

1.1.2 MensageiroImpl.java

```
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
public class MensageiroImpl extends UnicastRemoteObject implements Mensageiro {
    public MensageiroImpl() throws RemoteException {
        super();
    }
    public void enviarMensagem( String msg ) throws RemoteException {
        System.out.println( msg );
    }
    public String lerMensagem() throws RemoteException {
        return " Olá. Recebi a tua mensagem... Bom fim de semana";
    }
}
```

1.1.3 MensageiroServidor.java

```
import java.rmi.Naming;
public class MensageiroServidor {
    public MensageiroServidor() {
        try {
            Mensageiro m = new MensageiroImpl();
            Naming.rebind("rmi://localhost:1099/ServicoMensageiro", m);
        }
        catch( Exception e ) {
            System.out.println( "Anomalia: " + e );
        }
    }
    public static void main(String[] args) {
        new MensageiroServidor();
    }
}
```

1.1.4 MensageiroCliente.java

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.NotBoundException;
import java.net.MalformedURLException;

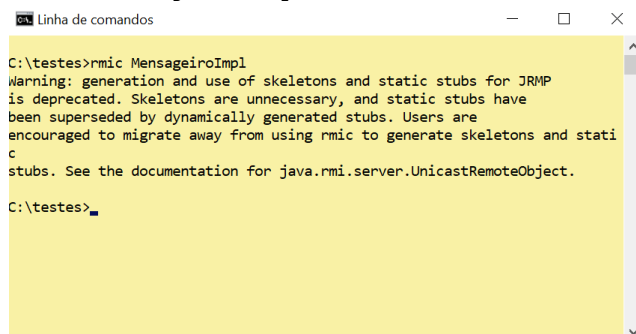
public class MensageiroCliente {
    public static void main( String args[] ) {
        try {
            Mensageiro m = (Mensageiro) Naming.lookup(
                "rmi://localhost/ServicoMensageiro" );
            System.out.println( m.lerMensagem() );
            m.enviarMensagem( "Boa tarde...Hoje é Sexta-feira!" );
        }
        catch( MalformedURLException e ) {
            System.out.println();
            System.out.println( "MalformedURLException: " + e.toString() );
        }
        catch( RemoteException e ) {
            System.out.println();
            System.out.println( "RemoteException: " + e.toString() );
        }
        catch( NotBoundException e ) {
            System.out.println();
            System.out.println( "NotBoundException: " + e.toString() );
        }
        catch( Exception e ) {
            System.out.println();
            System.out.println( "Exception: " + e.toString() );
        }
    }
}
```

1.1.3 Execução

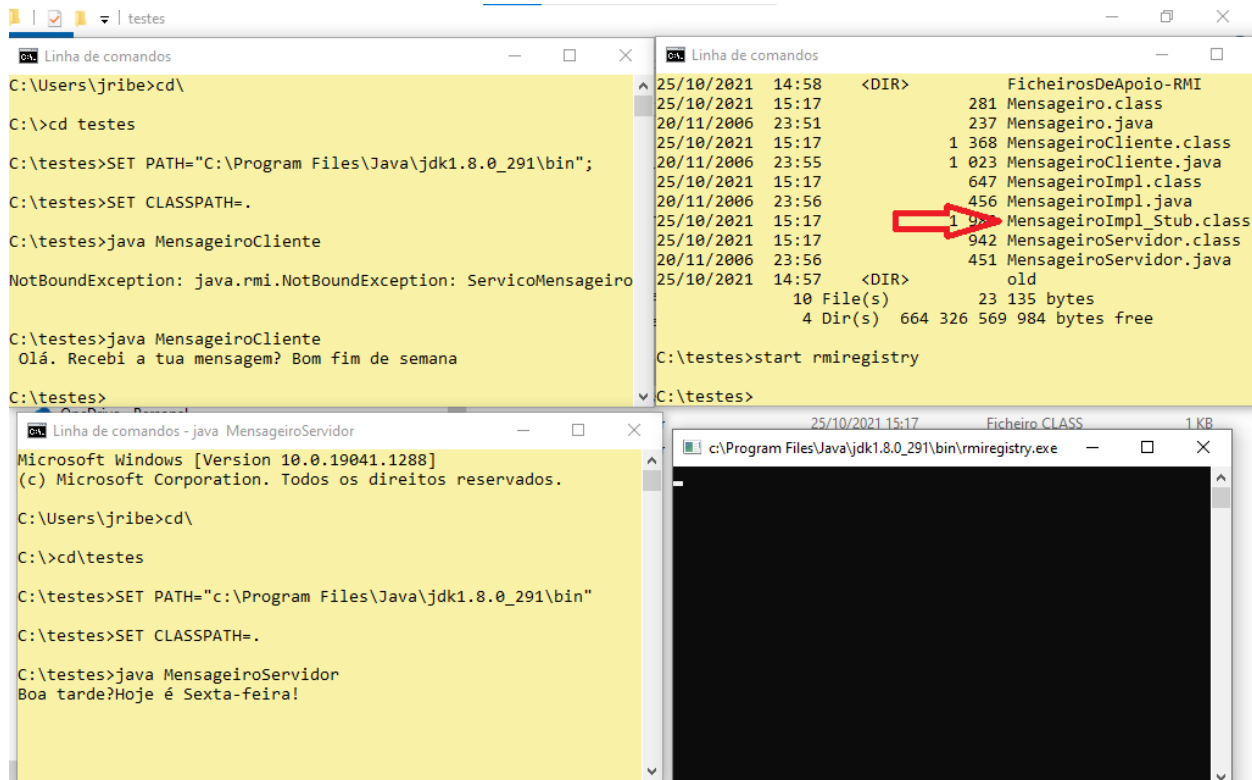
```
set PATH=%PATH%;C:\Program Files\Java\jdk1.7.0_17\bin
```

```
set CLASSPATH=.
```

```
rmic MensageiroImpl
```



```
start rmiregistry
```



1.2 Exercício 2

Implemente no servidor os métodos correspondentes à operação de SOMA e à multiplicação de valores. Pretende-se que o cliente peça ao servidor quanto é por exemplo a soma de $4 + 5$ e a Multiplicação de $3 * 2$;

1.2.1 Calculator.java

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface Calculator extends Remote
{
    public long add(long a, long b) throws RemoteException;
}
```

1.2.2 CalculatorServer.java

```
import java.rmi.Naming;

public class CalculatorServer
{
```

```
CalculatorServer()
{
    try
    {
        Calculator c= new CalculatorImpl();
        Naming.rebind ("rmi://127.0.0.1:1099/CalculatorService",c);
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}

public static void main(String[] args)
{
    new CalculatorServer();
}
}
```

1.2.3 CalculatorImpl.java

```
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class CalculatorImpl extends UnicastRemoteObject implements Calculator
{
    protected CalculatorImpl() throws RemoteException
    {
        super();
    }

    public long add(long a, long b) throws RemoteException
    {
        return a+b;
    }
}
```

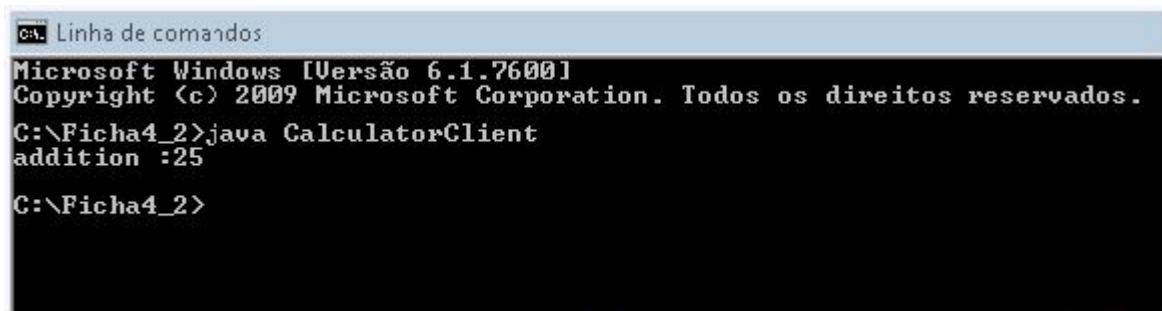
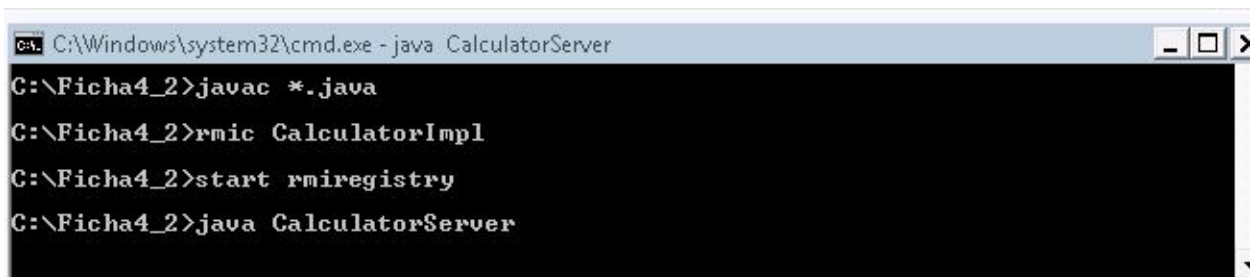
1.2.4 CalculatorClient.java

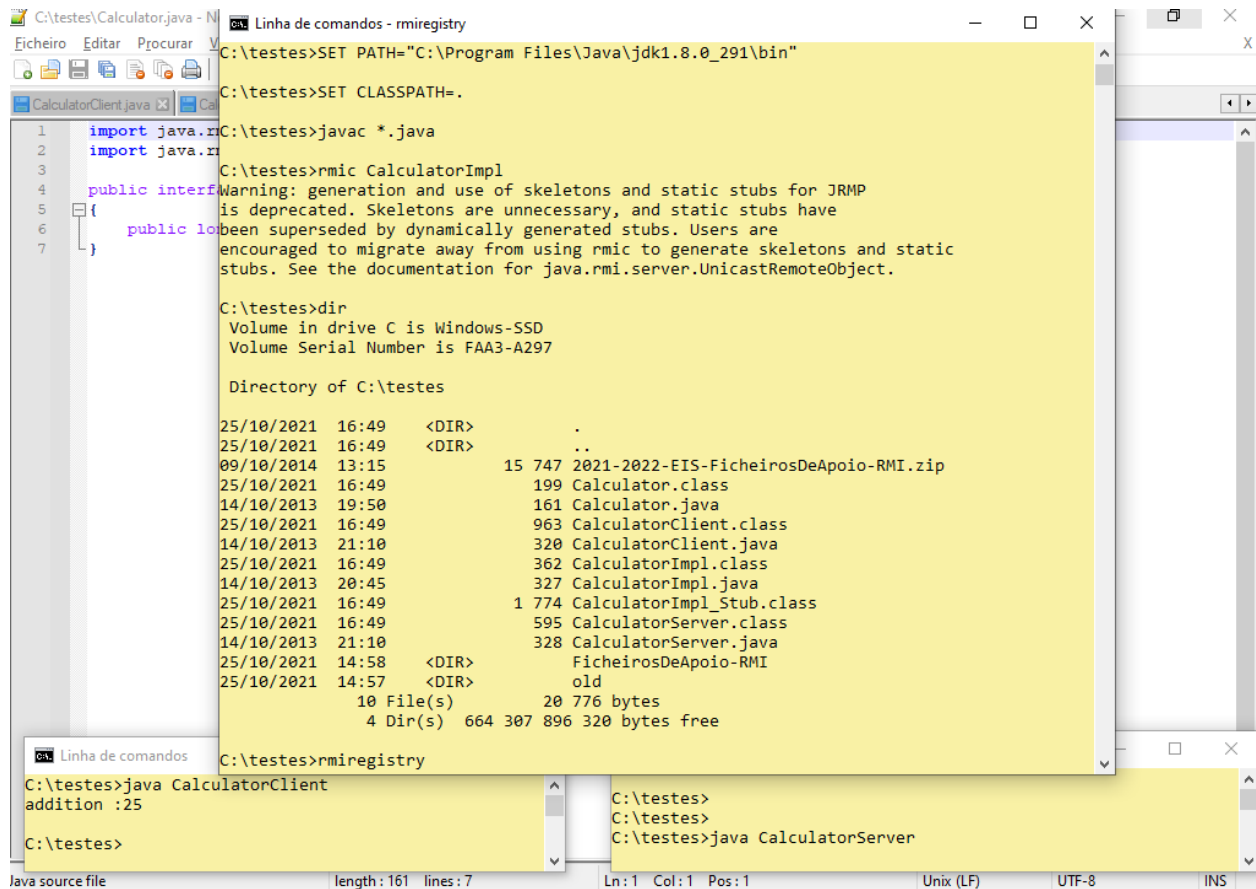
```
import java.rmi.Naming;

public class CalculatorClient
{
    public static void main(String[] args)
    {
        try
        {
            Calculator c = (Calculator) Nam-
ing.lookup("//127.0.0.1:1099/CalculatorService");
            System.out.println("addition :" + c.add(10,15));
        }
        catch (Exception e)
        {
        }
    }
}
```

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

1.2.4 Execução





```
C:\testes\Calculator.java - N
Ficheiro Editar Procurar V
C:\testes>SET PATH="C:\Program Files\Java\jdk1.8.0_291\bin"
C:\testes>SET CLASSPATH=.
C:\testes>javac *.java
C:\testes>rmic CalculatorImpl
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.
C:\testes>dir
Volume in drive C is Windows-SSD
Volume Serial Number is FAA3-A297

Directory of C:\testes

25/10/2021 16:49 <DIR> .
25/10/2021 16:49 <DIR> ..
09/10/2014 13:15 15 747 2021-2022-EIS-FicheirosDeApoio-RMI.zip
25/10/2021 16:49 199 Calculator.class
14/10/2013 19:50 161 Calculator.java
25/10/2021 16:49 963 CalculatorClient.class
14/10/2013 21:10 320 CalculatorClient.java
25/10/2021 16:49 362 CalculatorImpl.class
14/10/2013 20:45 327 CalculatorImpl.java
25/10/2021 16:49 1 774 CalculatorImpl_Stub.class
25/10/2021 16:49 595 CalculatorServer.class
14/10/2013 21:10 328 CalculatorServer.java
25/10/2021 14:58 <DIR> FicheirosDeApoio-RMI
25/10/2021 14:57 <DIR> old
10 File(s) 20 776 bytes
4 Dir(s) 664 307 896 320 bytes free

C:\testes>rmiregistry
C:\testes>java CalculatorClient
addition :25
C:\testes>
C:\testes>java CalculatorServer
```

1.3 Contactos

Implemente uma agenda de contactos (Nome da Pessoa e Numero de Telemóvel) no servidor. Pode conseguir isto através da criação de um ArrayList ou um HashMap, devendo preencher essa estrutura com valores. Crie um cliente que solicite ao servidor para lhe indicar o número de telefone do João.

1.3.1 RMIArrayList.java

```
import java.util.*;

import java.rmi.*;

public interface RMIArrayList extends Remote
{
    public ArrayList passArrayList(ArrayList a) throws RemoteException;
    public ArrayList passArrayListnumero(ArrayList b) throws RemoteException;
}
```

1.3.2 RMIArrayListImpl.java

```
import java.rmi.*;
import java.net.*;
import java.util.*;
```

```
public class RMIArrayListImpl extends java.rmi.server.UnicastRemoteObject
    implements RMIArrayList
{
    public RMIArrayListImpl() throws RemoteException
    {
        super();
    }

    // receive an array list, print it, add an element and return it
    public ArrayList passArrayList(ArrayList a) throws RemoteException
    {
        System.out.println(" RMI ArrayList server received array list " + a);
        return a;
    }

    public ArrayList passArrayListnumero(ArrayList b) throws RemoteException
    {
        System.out.println(" RMI ArrayList server received array list " + b);
        return b;
    }

    public static void main(String args[])
    {
        try
        {
            // bind to RMI server
            RMIArrayListImpl h = new RMIArrayListImpl();
            RMIArrayListImpl he = new RMIArrayListImpl();
            Naming.rebind("rmi://localhost/RMIArrayList", h);
            Naming.rebind("rmi://localhost/RMIArrayList", he);
            System.out.println("RMI ArrayList server ready");
        }
        catch (RemoteException re)
        {
            System.out.println("Remote Exception " + re);
        }
        catch (Exception e)
        {
            System.out.println(" Exception " + e);
        }
    }
}
```

1.3.3 RMIArrayListClient.java

```
import java.rmi.*;
import java.util.*;

public class RMIArrayListClient
{
    public static void main(String args[])
    {
        String host="localhost";    // RMI host name
```

```

/* System.setSecurityManager(new RMISecurityManager()); */
try
{
    // lookup RMI server
    RMIArrayList h = (RMIArrayList) Naming.lookup("rmi://" + host + "/RMIArrayList");
    RMIArrayList he = (RMIArrayList) Naming.lookup("rmi://" + host + "/RMIArrayList");
    // create ArrayList and add some elements
    ArrayList<String> v = new ArrayList<String>();
    ArrayList<String> ve = new ArrayList<String>();
    v.add( "Jorge" );
    ve.add("915554");
    v.add( "Ribeiro" );
    ve.add("915553");
    v.add( "Joao" );
    ve.add("915552");
    v.add( "Outro" );
    ve.add("915551");
    System.out.println("RMIArrayListClient sending: " + v + ve);
    // send array list to server and print array list received back
    ArrayList a = h.passArrayList(v);
    ArrayList b = h.passArrayList(ve);
    System.out.println("RMIArrayListClient received: " + a + b);

    System.out.println("CONTACTO:");
    System.out.println("NOME: " + a.get(2));
    System.out.println("NUMERO: " + b.get(2));
}
catch (Exception e)
{
    System.out.println("Exception in main " + e);
}
}

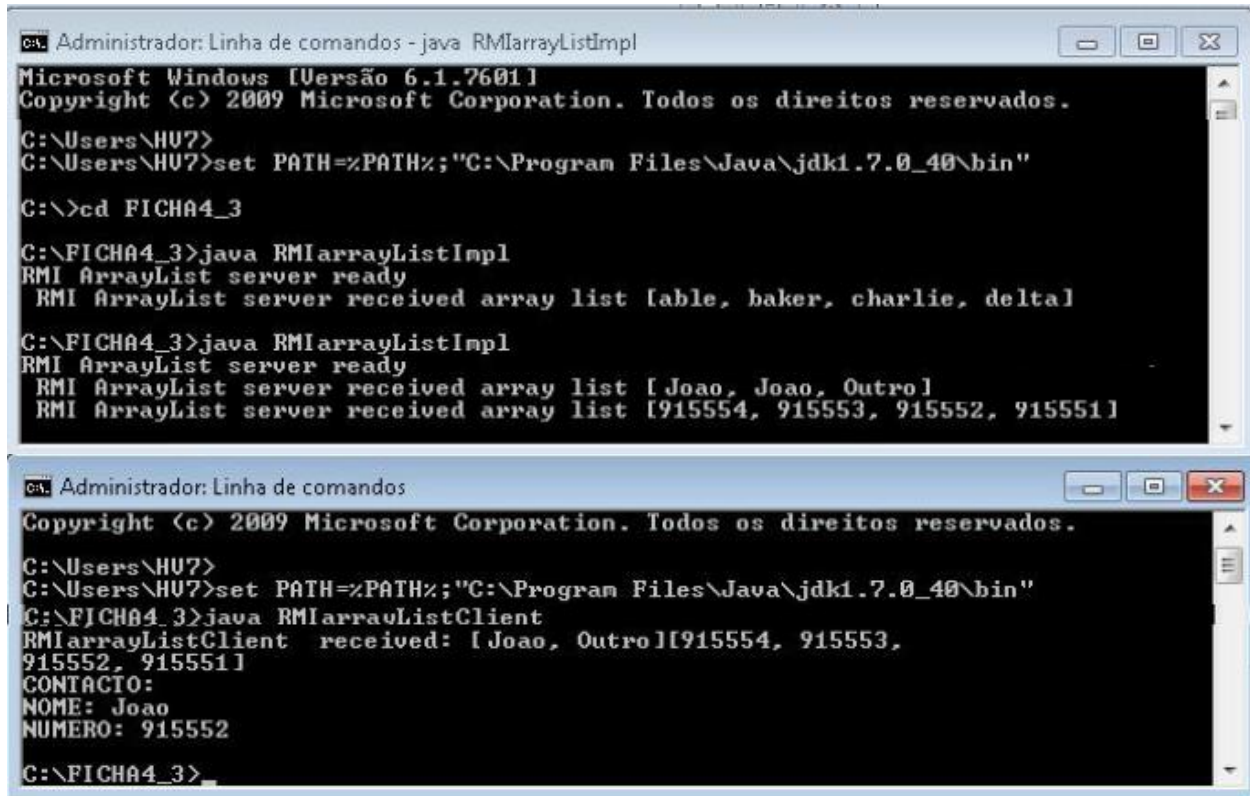
```

1.3.4 Execução

```

C:\FICHA4_3>javac RMIArrayListImpl.java
C:\FICHA4_3>rmic RMIArrayListImpl
C:\FICHA4_3>rmiregistry
C:\FICHA4_3>javac RMIArrayListImpl.java
C:\FICHA4_3>javac *.java
C:\FICHA4_3>rmic RMIArrayListImpl
C:\FICHA4_3>rmiregistry

```



The image displays two screenshots of a Windows command prompt window titled "Administrador: Linha de comandos - java RMICollectionImpl".

The first screenshot shows the following commands and output:

```
C:\Users\HU7>
C:\Users\HU7>set PATH=%PATH%; "C:\Program Files\Java\jdk1.7.0_40\bin"
C:\>cd FICHA4_3
C:\FICHA4_3>java RMICollectionImpl
RMI ArrayList server ready
RMI ArrayList server received array list [able, baker, charlie, delta]
C:\FICHA4_3>java RMICollectionImpl
RMI ArrayList server ready
RMI ArrayList server received array list [Joao, Joao, Outro]
RMI ArrayList server received array list [915554, 915553, 915552, 915551]
```

The second screenshot shows the following commands and output:

```
C:\Users\HU7>
C:\Users\HU7>set PATH=%PATH%; "C:\Program Files\Java\jdk1.7.0_40\bin"
C:\FICHA4_3>java RMICollectionClient
RMICollectionClient received: [Joao, Outro][915554, 915553,
915552, 915551]
CONTRATO:
NOME: Joao
NUMERO: 915552
C:\FICHA4_3>
```