Exercises on RMI

Exercise 1

- The server accept a string as argument and returns the overturned string
- The client sends a string, receives an elaborated string and prints the received string

Exercise 1 – interface

```
import java.rmi.*;

public interface Es1RMIinterface extends
java.rmi.Remote
{
  String overturn(String s) throws
java.rmi.RemoteException;
}
```

Exercise 1 – server

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

public class Es1RMIserver extends UnicastRemoteObject implements
Es1RMIinterface
{
  private String name;
  private static String MyServer = "overturner";

  public Es1RMIserver(String s) throws RemoteException {
    super();
    name = s;
  }

  public String overturn(String s) throws RemoteException {
    return new StringBuffer(s).reverse().toString();
  }
```

Exercise 1 – server

```
public static void main(String args[])
  {
    // security manager needed to load remote classes
    // deprecated from Java 17
    if (System.getSecurityManager() == null) {
        System.setSecurityManager(new SecurityManager());
    }

    try {
        Es1RMIserver obj = new Es1RMIserver("overturner");
        if (args.length > 0)
          MyServer = args[0] + MyServer;
        Naming.rebind(MyServer, obj);
        System.out.println("Es1RMIserver: " + MyServer + " bound in
registry");
    } catch (Exception e) {
        e.printStackTrace();
    }
  }
}
```

Exercise 1 – client

```
import java.rmi.*;

public class Es1RMIclient
{
  public static void main(String args[])
  {
    String ServerStr = "overturner";
    System.setSecurityManager(new
SecurityManager());

    String message = "ciao";
    if (args.length > 0) {
      ServerStr = "//" + args[0] + "/" + ServerStr;
      System.out.println("Server: " + ServerStr);
    }
```

Exercise 1 – client

```
   try {
            Es1RMIinterface obj =
(Es1RMIinterface)Naming.lookup(ServerStr);
            message = obj.overturn(message);
            System.out.println("Message
received: " + message);
     } catch (Exception e) {
          e.printStackTrace();
     }
   }
}
```

Exercise 2

- The server provides 2 services
  - Greeting
  - Local time

- The client receives from command line an argument specifying the service to ask to the server

Exercise 2 – interface

```
import java.rmi.*;

public interface Es2RMIinterface extends
java.rmi.Remote
{
  public String greeting() throws
java.rmi.RemoteException;
  public int hour() throws
java.rmi.RemoteException;
}
```

Exercise 2 – server

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

public class Es2RMIserver extends UnicastRemoteObject implements
Es2RMIinterface
{
  private String name;
  private static String MyServer = "2services";

  public Es2RMIserver(String s) throws RemoteException
  {
    super();
    name = s;
  }
```

Exercise 2 – server

```
  public String greeting() throws
java.rmi.RemoteException
  {
    System.out.println("Received request for
greeting");
    return "Hello RMI world!";
  }

  public int hour() throws java.rmi.RemoteException
  {
    System.out.println("Received request for hour");
    GregorianCalendar calendar=new
GregorianCalendar(TimeZone.getDefault());
    return calendar.get(Calendar.HOUR_OF_DAY);
  }
```

Exercise 2 – server

```
  public static void main(String args[])
  {
    // security manager needed to load remote classes
    // deprecated from Java 17
    if (System.getSecurityManager() == null) {
        System.setSecurityManager(new SecurityManager());
    }

    try {
        Es2RMIserver obj = new Es2RMIserver(MyServer);
        if (args.length > 0)
          MyServer = args[0] + MyServer;
        Naming.rebind(MyServer, obj);
        System.out.println("Es2RMIserver: " + MyServer + " bound in registry");
    } catch (Exception e) {
          System.out.println("Es2RMIserver err: " + e.getMessage());
          e.printStackTrace();
    }
  }
}
```

Exercise 2 – client

```
import java.rmi.*;

public class Es2RMIclient
{
  public static void main(String args[])
  {
    String ServerStr = "2services";
    System.setSecurityManager(new SecurityManager());

    String message = "";
    int h, service = 0;
    if (args.length > 0)
    {
      for (String arg : args)
       if (arg.startsWith("-"))
         service = Integer.parseInt(arg.substring(1));
       else
        if (!ServerStr.startsWith("//")) {
          ServerStr = "//" + args[0] + "/" + ServerStr;
          System.out.println("Server: " + ServerStr);
        }
    }
```

Exercise 2 – client

```
  try {
        Es2RMIinterface obj = (Es2RMIinterface)Naming.lookup(ServerStr);
        switch (service) {
        case 1: message = obj.greeting();
                System.out.println("Message received: " + message);
                break;
        case 2: h = obj.hour();
                System.out.println("Time received: " + h);
                break;
        default: System.out.println("Unrecognized service request");
        System.out.println("Use: Es2RMIclient [server] -N");
        }

    } catch (Exception e) { e.printStackTrace(); }
  }
}
```

Exercise 3

- The server receives:
  - The name of a file
  - The position of the byte to be read
- The server open the file, reads the byte at the specified position and returns the read byte

- The client:
  - Sends a string representing the name of a file and the position of a byte to be read
  - Receives the read byte
  - Writes the byte to a local file

Exercise 3 – interface

```
import java.rmi.*;
import java.io.*;

public interface Es3RMIinterface extends
Remote
{
  public byte read(String filename, long
pos) throws RemoteException,
IOException, EOFException;
}
```

Exercise 3 – server

```
import java.rmi.*; import java.io.*; import java.rmi.server.UnicastRemoteObject;

public class Es3RMIserver extends UnicastRemoteObject implements Es3RMIinterface
{
  private String name;
  private static String MyServer = "filereader";

  public Es3RMIserver(String s) throws RemoteException {
    super(); name = s; }

  public byte read(String filename, long pos) throws RemoteException, IOException,
EOFException {
    FileOutputStream fr = new FileOutputStream (filename);
    fr.skip(pos);
    int i = fr.read();
    if (i < 0)
      throw new EOFException();
    return (byte)i;
  }
```

Exercise 3 – server

```
  public static void main(String args[])
  {
    // security manager needed to load remote classes
    // deprecated from Java 17
    if (System.getSecurityManager() == null) {
        System.setSecurityManager(new SecurityManager());
    }

    try {
        Es3RMIserver obj = new Es3RMIserver(MyServer);
        if (args.length > 0)
          MyServer = args[0] + MyServer;
        Naming.rebind(MyServer, obj);
        System.out.println("Es3RMIserver: " + MyServer + " bound in
registry");
    } catch (Exception e) {
          e.printStackTrace();
    }
  }
}
```

Exercise 3 – client

```
import java.rmi.*;

public class Es3RMIclient
{
  public static void main(String args[])
  {
    String ServerStr = "filereader";
    System.setSecurityManager(new SecurityManager());

    byte c;
    String filename = "prova.txt";
    long pos = 3;
    if (args.length > 0) {
      ServerStr = "//" + args[0] + "/" + ServerStr;
      System.out.println("Server: " + ServerStr);
    }
```

Exercise 3 – client

```
  try {
          Es3RMIinterface obj =
(Es3RMIinterface)Naming.lookup(ServerStr);
          c = obj.read(filename, pos);
          System.out.println("Byte received:
" + c);
    } catch (Exception e) {
        System.out.println("Es3client
exception: "+ e.getMessage());
        e.printStackTrace();
    }
  }
}
```

Exercise 4

- Iterate the previous exercise to transfer the content of an entire file
- Define how to manage the end of file

Exercise 4 – client

```
import java.rmi.*; import java.io.*;

public class Es4RMIclient
{
  public static void main(String args[])
  {
    String ServerStr = "filereader";
    byte c;
    String filename = "prova.txt";
    String localfile = "local.txt";
    long pos = 0;
    if (args.length > 0)
    {
      ServerStr = "//" + args[0] + "/" + ServerStr;
      System.out.println("Server: " + ServerStr);
    }
```

Exercise 4 – client

```
FileOutputStream fw = null;
  try {
       fw = new FileOutputStream(localfile);
       Es3RMIinterface obj = (Es3RMIinterface)Naming.lookup(ServerStr);
       while (true){
         c = obj.read(filename, pos);
         fw.write(c);
         pos++;
       }
  }
  catch (EOFException e) { //end of file
    try { fw.close(); }
    catch (IOException ioe) {
     System.out.println("Error in close: " + ioe);
    }
  }
  catch (Exception e) {
     System.out.println("Es4client exception: "+ e.getMessage());
     e.printStackTrace();
  }
}}
```