Lab​ ​11​ ​-​ ​RMI​ ​Systems

This ​ lab​ ​ contains​ ​ in-class​ ​ exercises​ ​ related​​ to​ ​ RMI​ ​ System.​

Task ​ 1:​ ​ Given ​ the​ ​ remote​ ​ interface:​

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*​ ​Compilation:​ ​javac​ ​DisplayFile.java

\*

\*​ ​Lab​ ​11​ ​exercise​ ​1

\*​ ​Remote​ ​Interface

\*

\*​ ​@author​ ​Jordan​ ​Anastasiade

\*​ ​@version​ ​1.0,​ ​5​ ​Feb​ ​2008

\*​ ​@version​ ​1.1,​ ​22​ ​Aug​ ​2017

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public​ ​interface​ ​DisplayFile​ ​extends​ ​java.rmi.Remote​ ​{

​ /\*\*

​ ​\*​ ​Reads​ ​a​ ​file.​ ​Each​ ​line​ ​is​ ​stored​ ​in​ ​a​ ​vector

​ ​\*​ ​@param​ ​fileName​ ​the​ ​filename

​ ​\*​ ​@return​ ​vector​ ​containing​ ​the​ ​file​ ​lines

​ ​\*​ ​@throws​ ​java.rmi.RemoteException

​ ​\*​ ​@throws​ ​java.io.FileNotFoundException

​ ​\*/

​ ​public​ ​java.util.Vector​ ​display(String​ ​fileName)

​ ​throws​ ​java.rmi.RemoteException,​ ​java.io.FileNotFoundException;

}

Develop ​ an​ ​ RMI​ ​ Service​ ​ that​ ​ can​ ​ display​ ​ the​ ​ content​​ of​ ​ a​ ​ file​ ​ present​ ​ on​​ the​ ​ server​​ site.​

Hint: ​ the​ ​ implementation​ ​ class​ ​ is​ ​ given​ ​ ​to ​ you​

import​ ​java.util.Vector; import​ ​java.io.\*;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*​ ​Compilation:​ ​javac​ ​DisplayFileImpl.java

\*

\*

\*​ ​Solution​ ​Lab​ ​11​ ​exercise​ ​1

\*​ ​The​ ​implementation​ ​class

\*

\*​ ​@author​ ​Jordan​ ​Anastasiade

\*​ ​@version​ ​1.0,​ ​5​ ​Feb​ ​2008

\*​ ​@version​ ​1.1,​ ​22​ ​Aug​ ​2017

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ public​ ​class​ ​DisplayFileImpl​ ​extends​ ​java.rmi.server.UnicastRemoteObject ​ ​implements​ ​DisplayFile​ ​{

​ ​//default​ ​ctr.

​ ​public​ ​DisplayFileImpl()​ ​throws​ ​java.rmi.RemoteException​ ​{

​ ​super();

​ ​}

​ ​ ​/\*\*

​ ​\*​ ​Reads​ ​a​ ​file.​ ​Each​ ​line​ ​is​ ​stored​ ​in​ ​a​ ​vector

​ ​\*​ ​@param​ ​s​ ​the​ ​file​ ​name

​ ​\*​ ​@return​ ​vector​ ​containing​ ​the​ ​file​ ​lines

​ ​\*​ ​or​ ​null​ ​if​ ​file​ ​does​ ​not​ ​exist

​ ​\*​ ​@throws​ ​java.rmi.RemoteException

​ ​\*​ ​@throws​ ​java.io.FileNotFoundException

​ ​\*/

​ ​public​ ​Vector​ ​display(String​ ​s)​ ​throws​ ​java.rmi.RemoteException,

​ ​FileNotFoundException​ ​{

​ ​Vector<String>​ ​v​ ​=​ ​new​ ​Vector<String>();

​ ​String​ ​line​ ​=​ ​null;

​ ​try​ ​{

​ ​BufferedReader​ ​in​ ​=​ ​new​ ​BufferedReader(new​ ​FileReader(s));

​ ​while​ ​((line​ ​=​ ​in.readLine())​ ​!=​ ​null)

​ ​v.add(line);

​ ​}​ ​catch​ ​(IOException​ ​e)​ ​{

​ ​System.out.println("IO​ ​Exception​ ​file:"​ ​+​ ​s);

​ ​return​ ​null;

​ ​}

​ ​return​ ​v;

​ ​}

}

Task ​ 2:​ ​ Develop​ ​ an​ ​ RMI​ ​ system​ ​ that​​ is​ ​ capable​ ​ of​ ​ finding​ ​ all​ ​ the​ ​​lines​ of​ ​ the​ ​ file​ ​ on​​ the​ server ​ site​ ​ which​ ​ contains​ ​ a​ ​ given​ ​ string​

Here ​ is​ ​ the​ ​ remote​ ​ interface:​

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*​ ​Compilation:​ ​javac​ ​Grep.java

\*

\*

\*​ ​Solution​ ​Lab​ ​11​ ​exercise​ ​2

\*​ ​Remote​ ​Interface

\*

\*​ ​@author​ ​Jordan​ ​Anastasiade

\*​ ​@version​ ​1.0,​ ​5​ ​Feb​ ​2008

\*​ ​@version​ ​1.1,​ ​22​ ​Aug​ ​2017

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ public​ ​interface​ ​Grep​ ​extends​ ​java.rmi.Remote​ ​{

​ /\*

​ ​\*​ ​@param​ ​file​ ​input​ ​file​ ​of​ ​type​ ​text

​ ​\*​ ​@param​ ​s​ ​string​ ​to​ ​look​ ​for​ ​in​ ​the​ ​file

​ ​\*​ ​@return​ ​the​ ​lines​ ​where​ ​the​ ​string​ ​was​ ​found

​ ​\*​ ​@throws​ ​java.rmi.RemoteException

​ ​\*​ ​@throws​ ​java.io.FileNotFoundException

​ ​\*/

​ ​public​ ​java.util.Vector​ ​find(String​ ​file,​ ​String​ ​s)

​ ​throws​ ​java.rmi.RemoteException,​ ​java.io.FileNotFoundException;

}

Task ​ 3:​ ​ Given​ ​ the​ ​ class​ ​Car

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*​ ​Compilation:​ ​javac​ ​Car.java

\*

\*

\*​ ​Solution​ ​Lab​ ​11​ ​exercise​ ​3

\*​ ​Serializable​ ​object​ ​to​ ​be​ ​sent​ ​to​ ​RMIServer

\*

\*​ ​@author​ ​Jordan​ ​Anastasiade

\*​ ​@version​ ​1.0,​ ​5​ ​Feb​ ​2008

\*​ ​@version​ ​1.1,​ ​22​ ​Aug​ ​2017

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ public​ ​class​ ​Car​ ​implements​ ​Serializable​ ​{

​ ​private​ ​String​ ​model;

​ ​private​ ​String​ ​owner;

​ ​private​ ​double​ ​mileage;

​ ​private​ ​Integer​ ​registration;

​ ​public​ ​Car(String​ ​brand,​ ​String​ ​name,​ ​double​ ​k)​ ​{

​ ​model​ ​=​ ​brand;

​ ​owner​ ​=​ ​name;

​ ​mileage​ ​=​ ​k;

​ ​registration​ ​=​ ​0;

​ ​}

​ ​public​ ​String​ ​toString()​ ​{

​ ​return​ ​"Model:​ ​"​ ​+​ ​model​ ​+​ ​"​ ​Owner:​ ​"​ ​+​ ​owner​ ​+

​ ​"​ ​mileage:​ ​"​ ​+​ ​mileage​ ​+

​ ​"​ ​Registration:​ ​"​ ​+​ ​registration;

​ ​}

​ ​public​ ​void​ ​getRegistered(Integer​ ​plate)​ ​{

​ ​registration​ ​=​ ​plate;

​ ​}

}

Develop ​ an​ ​ RMI​ ​ System​ ​ that​ ​ allows​ ​ a​ ​ car​ ​ to​​ be​ ​ registered.​ ​ The​ ​ system​ ​ should​​ work​ ​ in​​ this​ way:

1. The ​ RMI​ ​ client​ ​ builds​ ​ a​ ​ car​ ​ and​ ​ sends​ ​ ​it​ to​ ​ RMI​ ​ Server​ ​ for​ ​ ​registration
2. The ​ RMI​ ​ server​ ​ registers​ ​ the​ ​ car​ ​ and​ ​ sends​ ​ ​it ​ back​ ​ to​​ the​ ​ client​
3. The ​ name​ ​ of​ ​ RMI​ ​ system​​ should​ ​ be​ ​ “wonderful”.​