Sarrera

Laborategi honetan aplikazio banadu bat garatuko dugu RMI teknologia erabiliz. Horretarako urruneko zerbitzari batean negozio logika bat kokatuko dugu eta beste makina batean dagoen bezero batetik atzituko dugu. Garatuko den aplikazioak login eta password bat baieztatuko du.

Helburuak

- RMI teknologiak dauzkan interfaze eta klaseak ezagutu
- RMI zerbitzari bat definitu negozio logika bat inplementatzen duena.
- Bezero bat inplementatu, RMI-ren bitartez zerbitzariaren negozio logikara atzitzen duena.

Jarraitzeko pausoak

1. Java proiektu bat sortu, eta bere egituran labRMI paquete bat.

2. Urruneko interfazea definitu. Intefaze honetan urruneko negozio logikak aurkezten dituen metodoak espezifikatzen dira. Gure kasuan hurrengo kodea izango du:

```
package labRMI;

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

public interface InterfazeNegozioLogika extends Remote{
  boolean loginEgin(String a,String b) throws RemoteException;

}
```

Ikusten den bezala, interfaze honek Remote klasetik hedatzen du eta bere metodo guztiek RemoteException salbuespena altxatu dezakete.

3. Urruneko interfazea inplementatzen duen klasea definitu, hurrengo kodea aurkezten duen bezala:

```
package labRMI;

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

public class SistemaSarrera extends UnicastRemoteObject implements
InterfazeNegozioLogika{

      public SistemaSarrera() throws RemoteException{}
      public boolean loginEgin(String a, String b) throws RemoteException {
            return a.compareTo(b)==0;
      }
}
```

Klase honen ezaugarriak hurrengoak dira:

a) UnicastRemoteObject klasetik hedatzen du.

b) Aurreko pausoan definitutako interfazea inplementatzen du (InterfazeNegozioLogika).

c) Derrigorrez metodo eraikitzaile bat eduki behar du RemoteException salbuespenarekin.

d) Inplementatzen diren metodo guztiek RemoteException salbuespena altxatzeko aukera eduki behar dute.

4. Klase jaurtitzailea definitu. Klase honen helburua hurrengoa da:

a) RMI zerbitzari bat definitu urruneko makinan (erregistroSortu() metodoa).

b) Aurreko puntuan definitu dugun negozio logikaren objektu bat sortu eta erregistratu zerbitzarian bezeroek atzitzeko aukera izan dezaten (init() metodoa).

c) Aurreko metodoak exekutatu main() metodo batetik.

Klase honen kodea hurrengoa da:

```
package labRMI;

import java.rmi.Naming;
import java.rmi.RMISecurityManager;
import java.rmi.Remote;


public class RMIJaurtitzailea {

 public static void erregistroSortu(){
  System.setProperty("java.security.policy", "/Users/iturrioz/java.policy");
  //System.setProperty("java.rmi.server.codebase", "C:\\jwnl\\java.policy");
  System.setSecurityManager(new RMISecurityManager());
  try { java.rmi.registry.LocateRegistry.createRegistry(9999);//RMIREGISTRY
sortu
      } catch (Exception e) {System.out.println(e.toString()+"\nRMI registry
sortuta zegoen");}

  }

   public static void init() {
    try {
        Remote urrunekoObjektua = new SistemaSarrera();
        System.out.println("objektu sorturta");
        String zerbitzua = "rmi://localhost:9999//sistemaSarrera";
          //         "//localhost:PortNumber/ServiceName"
          // urruneko zerbitzua erregistratu


          Naming.rebind(zerbitzua,urrunekoObjektua);
          System.out.println("objektua erregistratua");
        } catch (Exception e)
           {System.out.println(e.toString());}
        }
```

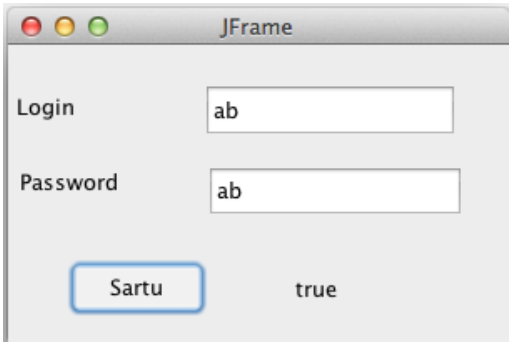
```
        public static void main(String[] args) {

       RMIJaurtitzailea.erregistroSortu();
       RMIJaurtitzailea.init();
        }
  }
```

OHAR: Oso garrantzitsua da, "java.policy" fitxategia kokatzea System.*setProperty*("java.security.policy", "c:\\kodea\\java.policy") definitu dugun helbidean (erregistroSortu() metodoan).

5. Klase jaurtitzailea exekutatu eta zerbitzaria martxan egongo da definitutako negozio logikako objektuarekin eskuragarri "sistemaSarrera" izenaran bitartez.

6. Frame bat definitu hurrego itxurarekin:



eta jarraian hurrengo aldaketak egin klasean:

a) Negozio logikaren atributu bat definitu bere esleipen metodoarekin.

```
        InterfazeNegozioLogika intNL;
        public void setNegozioLogika(InterfazeNegozioLogika i) {
                  intNL=i;
       }
```

b) botoien listener-ean negozio logikari deitu hurrengo kodean agertzen den bezala (kontuan hartu jTextField eta jLabel-en aldagaien izenak aldatzea)

```
public void actionPerformed(java.awt.event.ActionEvent e) {
       try {
         boolean b=intNL.loginEgin(login.getText(), pass.getText());
         emaitza.setText(Boolean.toString(b));
       } catch (RemoteException e1) {
       e1.printStackTrace();
       }
}
```

c) main metodo bat definitu urruneko neogozio logika eskuratzeko eta Frame-ari esleitzeko.

```
 public static void main(String[] args) {
   Aurkezpena p = new Aurkezpena();
   System.setProperty("java.security.policy", "/Users/iturrioz/java.policy");
   try{
      InterfazeNegozioLogika
inl=(InterfazeNegozioLogika)Naming.lookup("rmi://localhost:9999//sistemaSarrer
a");
     //negozio logika esleitu interfazeari
     p.setNegozioLogika(inl);
                 } catch(Exception e) {System.out.println("Error negozio
logika atzitzerakoan: "+e.toString());}
                p.setVisible(true);

               }
```

7. Interfaze grafikoaren klasea exekutatu eta frogatu.

Ariketak:

1. Nola egokituko zenuke "Aurkezpena+negozio logika bananduz" laborategia, beste **urruneko** negozio logika klase berri batekin, hau da, interfazeari pasatzen zaion negozio logika, urruneko zerbitzari batean kokatuta badago.

2. Nola aldatuko zenuke sistema urruneko negozio logikak, db4o datu basean baieztatzen baditu pasatutako datuak.