TDM04 d'Informatique Répartie
RMI

ASI4 - INSA Rouen
CORRECTION

Service de chiffrement en RMI

Le but de cet exercice est de réaliser un service de chiffrement distribué en RMI ; celui-ci proposera les 3 méthodes suivantes :

- `chiffrerEntier(int):int` qui, comme chiffrement, incrémentera de 1 l'entier transmis en paramètre ;
- `chiffrerDocument(Document):Document` qui chiffrera un document textuel passé par valeur en lui ajoutant simplement une chaîne de caractères ; la logique métier d'un objet de type `Document` vous est fournie dans le fichier `Annexes.tar.gz` ;
- `chiffrerFichier(Fichier):void` qui chiffrera un fichier passé par référence, là encore en lui ajoutant une chaîne de caractères ; la logique métier d'un objet de type `Fichier` vous est également fournie dans le fichier `Annexes.tar.gz` ;

NB : il s'agit d'Informatique Répartie donc pensez à tester votre programme avec serveur et client sur des machines différentes !

Remarques

- Vous aurez alors accès à la correction du TDM dès la fin de la séance.
- **Déposez un compte-rendu de 2 pages `TDMRMI-LOGIN.pdf` sur moodle chez TOUTES les personnes du binôme. Ce CR contiendra les informations que vous jugerez nécessaires.**
- Votre CR sera disponible pour vous lors de l'examen machine.

Correction

Encodeur.java

```
package serviceDeChiffrement;

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

public interface Encodeur extends Remote {
    public int chiffrerEntier(int aEncoder) throws RemoteException;
    public Document chiffrerDocument(Document aEncoder) throws
        RemoteException;
    public void chiffrerFichier(Fichier aEncoder) throws RemoteException;
}
```

EncodeurImpl.java

```
package serviceDeChiffrement;

import java.rmi.RemoteException;

public class EncodeurImpl implements Encodeur {
    public int chiffrerEntier(int aEncoder) throws RemoteException {
        return ++aEncoder;
    }

    public Document chiffrerDocument(Document aEncoder) throws
        RemoteException{
        return new Document("Traduction de :\"" + aEncoder.getContenu() + "\
            "");
    }

    public void chiffrerFichier(Fichier aEncoder) throws RemoteException{
```

```
        String contenu = aEncoder.getContenu();
        contenu = "Une ligne en plus\n" + contenu + "Une ligne en plus\n";
        aEncoder.setContenu(contenu);
    }
}
```

Fichier.java

```
package serviceDeChiffrement;

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

public interface Fichier extends Remote {
    public String getContenu() throws RemoteException;
    public void setContenu(String nouveauContenu) throws RemoteException;
}
```

FichierImpl.java

```
package serviceDeChiffrement;

import java.io.*;
import java.rmi.RemoteException;

public class FichierImpl implements Fichier {
    private String nomDuFichier;

    public FichierImpl() {
        this.nomDuFichier = "";
    }

    public FichierImpl(String nom) {
        this.nomDuFichier = nom;
    }

    public String getContenu() throws RemoteException {
        String contenu = "";
        try{
            BufferedReader br = new BufferedReader(new InputStreamReader(new
                FileInputStream(this.nomDuFichier)));
            while (br.ready()) {
                contenu += br.readLine() + "\n";
            }
            br.close();
        } catch(IOException ioe){ ioe.printStackTrace(); }
        return contenu;
    }

    public void setContenu(String nouveauContenu) throws RemoteException {
        try {
            BufferedWriter output=new BufferedWriter(new FileWriter(this.
                nomDuFichier,false));
            output.write(nouveauContenu);
            output.flush();
            output.close();
        } catch(IOException ioe){ ioe.printStackTrace(); }
    }
}
```

Document.java

```
package serviceDeChiffrement;

import java.io.Serializable;

public class Document implements Serializable {
    private String contenu;

    public Document(){
        this.contenu = "Contenu non defini";
    }

    public Document(String nouveauContenu){
        this.contenu = nouveauContenu;
```

```
    }

    public String getContenu(){
        return this.contenu;
    }

    public void setContenu(String nouveauContenu){
        this.contenu = nouveauContenu;
    }

    public String toString() {
        return this.contenu;
    }
}
```

Serveur.java

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.UnicastRemoteObject;
import java.util.Arrays;
import serviceDeChiffrement.*;

public class Serveur {
  public static void main(String args[]) {
    int port  = 1099;
    if(args.length==1)
      port = Integer.parseInt(args[0]);
    try {
      Encodeur skeleton = (Encodeur)UnicastRemoteObject.exportObject(new
          EncodeurImpl(), 0);
      Registry registry = LocateRegistry.getRegistry(port);
      if(!Arrays.asList(registry.list()).contains("Encodeur"))
        registry.bind("Encodeur", skeleton);
      else
        registry.rebind("Encodeur", skeleton);
      System.out.println("Service \"Encodeur\" lie au registre");
    } catch (Exception e) {
      System.out.println(e);
    }
  }
}
```

Client.java

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.UnicastRemoteObject;
import serviceDeChiffrement.*;

public class Client {
  public static void main(String args[]) {
    String machine = "localhost";
    int port = 1099;
    if(args.length==5) {
      machine = args[0];
      port = Integer.parseInt(args[1]);
    } else if(args.length==4)
      machine = args[0];
    try {
      Registry registry = LocateRegistry.getRegistry(machine, port);
      Encodeur stub = (Encodeur)registry.lookup("Encodeur");
      Fichier fichier = new FichierImpl(args[args.length-1]),
          stubFichier = (Fichier)UnicastRemoteObject.exportObject(fichier,
              0);
      System.out.println("==> " + stub.chiffrerEntier(Integer.parseInt(
          args[args.length-3])));
      System.out.println("==> " + stub.chiffrerDocument(new Document(args[
          args.length-2])));
      stub.chiffrerFichier(fichier);
      UnicastRemoteObject.unexportObject(fichier, true);
    } catch (Exception e) {
       System.out.println("Client exception: " +e);
```

```
    }
  }
}
```

compile.sh

```
javac serviceDeChiffrement/*.java

cp serviceDeChiffrement/Encodeur.class rmiregistry/serviceDeChiffrement
cp serviceDeChiffrement/Document.class rmiregistry/serviceDeChiffrement
cp serviceDeChiffrement/Fichier.class rmiregistry/serviceDeChiffrement

cp serviceDeChiffrement/Encodeur.class Serveur/serviceDeChiffrement
cp serviceDeChiffrement/EncodeurImpl.class Serveur/serviceDeChiffrement
cp serviceDeChiffrement/Document.class Serveur/serviceDeChiffrement
cp serviceDeChiffrement/Fichier.class Serveur/serviceDeChiffrement
cd Serveur
javac Serveur.java

cd ..
cp serviceDeChiffrement/Encodeur.class Client/serviceDeChiffrement
cp serviceDeChiffrement/Document.class Client/serviceDeChiffrement
cp serviceDeChiffrement/Fichier.class Client/serviceDeChiffrement
cp serviceDeChiffrement/FichierImpl.class Client/serviceDeChiffrement
cd Client
javac Client.java
```

Remarques

— Vous aurez alors accès à la correction du TDM dès la fin de la séance.
— **Déposez un compte-rendu de 2 pages `TDMRMI-LOGIN.pdf` sur moodle chez TOUTES les personnes du binôme. Ce CR contiendra les informations que vous jugerez nécessaires.**
— Votre CR sera disponible pour vous lors de l'examen machine.