TDM05 d'Informatique Répartie
RMI

ASI4 - INSA Rouen
CORRECTION

Calculatrice distribuée sur entiers positifs en RMI

Le but de cet exercice est de réaliser une calculatrice distribuée sur entiers positifs en RMI. Pour cela, vous respecterez les contraintes suivantes :

- Votre calculatrice supportera 4 types d'opération : additions, soustractions, multiplications et divisions. Le serveur proposera donc 4 services distribués et ne travaillera que sur des entiers positifs.
- Une exception sera levée en cas de paramètre passé négatif ou de résultat d'opération négatif.
- Vous créerez 3 répertoires pour le lancement du rmiregistry, du serveur et du client et contenant uniquement les fichiers nécessaires à leur lancement.

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

Correction

Calculatrice.java

```
package calculatriceRMI;

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

public interface Calculatrice extends Remote {
   public int additionner(int e1, int e2) throws RemoteException, EntierNegatif;
   public int soustraire(int e1, int e2) throws RemoteException, EntierNegatif;
   public int multiplier(int e1, int e2) throws RemoteException, EntierNegatif;
   public int diviser(int e1, int e2) throws RemoteException, EntierNegatif;
}
```

CalculatriceImpl.java

```
package calculatriceRMI;

import java.io.Serializable;
import java.rmi.RemoteException;

import calculatriceRMI.Calculatrice;

public class CalculatriceImpl implements Calculatrice, Serializable {
  public int additionner(int e1, int e2) throws EntierNegatif {
    int resultat = e1+e2;
    if (e1<0 || e2<0)
      throw new EntierNegatif();
    else
      return resultat;
   }

   public int soustraire(int e1, int e2) throws EntierNegatif {
    int resultat = e1-e2;
    if (e1<0 || e2<0 || resultat<0)
      throw new EntierNegatif();
    else
      return resultat;
   }

   public int multiplier(int e1, int e2) throws EntierNegatif {
    int resultat = e1*e2;
    if (e1<0 || e2<0)
      throw new EntierNegatif();
    else
      return resultat;
   }

   public int diviser(int e1, int e2) throws EntierNegatif {
    int resultat = e1/e2;
    if (e1<0 || e2<0)
      throw new EntierNegatif();
```

```
    else
      return resultat;
   }
}
```

EntierNegatif.java

```
package calculatriceRMI;

public class EntierNegatif extends Exception {
  public EntierNegatif(){
    super("Entier non positif !");
  }
}
```

Serveur.java

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

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

Client.java

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import calculatriceRMI.Calculatrice;

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);
      Calculatrice calculatriceRMI = (Calculatrice)registry.lookup("CalculatriceEntiers");
      int operande1 = Integer.parseInt(args[args.length-3]),
          operande2 = Integer.parseInt(args[args.length-1]),
          resultat=-1;

      if(args[args.length-2].equals("+"))
        resultat = calculatriceRMI.additionner(operande1, operande2);
      if(args[args.length-2].equals("-"))
        resultat = calculatriceRMI.soustraire(operande1, operande2);
      if(args[args.length-2].equals("x"))
        resultat = calculatriceRMI.multiplier(operande1, operande2);
      if(args[args.length-2].equals("/"))
        resultat = calculatriceRMI.diviser(operande1, operande2);

      System.out.println(operande1 + args[args.length-2] + operande2 + " = " + resultat);
    } catch (Exception e) {
       System.out.println("Client exception: " +e);
    }
  }
}
```

compile.sh

```
javac calculatriceRMI/*.java
cp calculatriceRMI/*.class Serveur/calculatriceRMI/
cp calculatriceRMI/Calculatrice.class Client/calculatriceRMI/
cp calculatriceRMI/EntierNegatif.class Client/calculatriceRMI/
cp calculatriceRMI/Calculatrice.class rmiregistry/calculatriceRMI/
cp calculatriceRMI/EntierNegatif.class rmiregistry/calculatriceRMI/
```

```
cd Serveur
javac *.java

cd ../Client
javac *.java

cd ..
```

Calculatrice distribuée sur Complexes en RMI

Le but de cet exercice est de réaliser une calculatrice distribuée sur Complexes en RMI. Pour cela, vous respecterez les contraintes suivantes :

- Votre calculatrice supportera 4 types d'opération : additions, soustractions, multiplications et divisions. Le serveur proposera donc 4 services distribués et ne travaillera que sur des Complexes (Objet à créer).
- Une exception sera levée en cas de division par 0.
- Vous réfléchirez bien sur le type des Complexes à transmettre (objets Serializable ou objets RMI).
- Vous créerez 3 répertoires pour le lancement du rmiregistry, du serveur et du client et contenant uniquement les fichiers nécessaires à leur lancement.

Correction

Calculatrice.java

```
package calculatriceRMI;

import java.rmi.Remote;
import java.rmi.RemoteException;
import calculatriceRMI.Complexe;

public interface Calculatrice extends Remote {
   public Complexe additionner(Complexe c1, Complexe c2) throws RemoteException;
   public Complexe soustraire(Complexe c1, Complexe c2) throws RemoteException;
   public Complexe multiplier(Complexe c1, Complexe c2) throws RemoteException;
   public Complexe diviser(Complexe c1, Complexe c2) throws RemoteException,ArithmeticException;
}
```

CalculatriceImpl.java

```
package calculatriceRMI;

import java.io.Serializable;
import java.rmi.RemoteException;
import java.lang.ArithmeticException;
import calculatriceRMI.Calculatrice;
import calculatriceRMI.Complexe;

public class CalculatriceImpl implements Calculatrice, Serializable {
  public Complexe additionner(Complexe c1, Complexe c2) throws RemoteException {
    Complexe z = new Complexe();
    z.re=c1.re+c2.re;
    z.im=c1.im+c2.im;
    return z;
  }

  public Complexe soustraire(Complexe c1, Complexe c2) throws RemoteException {
    Complexe z = new Complexe();
    z.re=c1.re-c2.re;
    z.im=c1.im-c2.im;
    return z;
  }

  public Complexe multiplier(Complexe c1, Complexe c2) throws RemoteException {
    Complexe z = new Complexe();
    z.re = c1.re * c2.re - c1.im * c2.im;
    z.im = c1.re * c2.im + c1.im * c2.re;
    return z;
  }

  public Complexe diviser(Complexe c1, Complexe c2) throws RemoteException,ArithmeticException {
    float r = c2.re*c2.re + c2.im*c2.im;
    Complexe z = new Complexe();
    if (r==0) {
      throw new ArithmeticException("Division par zero !");
    }else {
      z.re = (c1.re * c2.re + c1.im * c2.im) / r;
      z.im = (c1.im * c2.re - c1.re * c2.im) / r;
      return z;
    }
  }
}
```

Complexe.java

```
package calculatriceRMI;

import java.io.Serializable;

public class Complexe implements Serializable {
  public float re,im;

  public Complexe() {
    this.re=(float)0;
    this.im=(float)0;
  }

  public Complexe(int re,int im) {
    this.re=(float)re;
    this.im=(float)im;
  }

  public Complexe(float re,float im) {
    this.re=re;
    this.im=im;
  }

  public String toString(){
    String resultat="";
    if (this.re!=0)
      resultat=String.valueOf(this.re);
    if (this.im!=0) {
      if (this.im>0){
        if (resultat!="")
          resultat=resultat+"+"+String.valueOf(this.im);
        else
          resultat=String.valueOf(this.im);
      }else
        resultat=resultat+String.valueOf(this.im);
      resultat=resultat+"i";
    }
    return resultat;
  }
}
```

Serveur.java

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

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

Client.java

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import calculatriceRMI.Calculatrice;
import calculatriceRMI.Complexe;

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 {
      Complexe c1=new Complexe(2,4), c2=new Complexe(1,2), un=new Complexe(1,0), zero=new Complexe(0,0);
      Registry registry = LocateRegistry.getRegistry(machine, port);
      Calculatrice calculatrice = (Calculatrice)registry.lookup("CalculatriceComplexes");
```

```
      System.out.println("(2+4i) + (1+2i) = " + calculatrice.additionner(c1,c2));
      System.out.println("(2+4i) - (1+2i) = " + calculatrice.soustraire(c1,c2));
      System.out.println("(2+4i) x (1+2i) = " + calculatrice.multiplier(c1,c2));
      System.out.println("(2+4i) / (1+2i) = " + calculatrice.diviser(c1,c2));
      System.out.println("(2+4i) / 1 = " + calculatrice.diviser(c1,un));
      System.out.println("(2+4i) / 0 = " + calculatrice.diviser(c1,zero));
    } catch (Exception e) {
       System.out.println("Client exception: " +e);
    }
  }
}
```

compile.sh

```
javac calculatriceRMI/*.java
cp calculatriceRMI/*.class Serveur/calculatriceRMI/
cp calculatriceRMI/Calculatrice.class Client/calculatriceRMI/
cp calculatriceRMI/Complexe.class Client/calculatriceRMI/
cp calculatriceRMI/Calculatrice.class rmiregistry/calculatriceRMI/
cp calculatriceRMI/Complexe.class rmiregistry/calculatriceRMI/

cd Serveur
javac *.java

cd ../Client
javac *.java

cd ..
```

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 durant une semaine à compter de la fin du TDM.
- **Déposez un compte-rendu de 2 pages** `TDMRMI-NomPrenom.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.