***Atelier 3 : Rmi-Java***

Systèmes Client serveur

|  |  |
| --- | --- |
| \*\*\*\*\*\*\*\*\*\*\*\*Définition d’interface\*\*\*\*\*\*\*\*\*\*\*\*\* *import java.rmi.\*;*  *public interface HelloInterface extends java.rmi.Remote {*  */\* méthode qui imprime un message*  *prédéfini dans l’objet appelé \*/*  *public String sayHello() throws java.rmi.RemoteException;*  *}* **\*\*\*\*\*\*\*\*\*\*Programme du client\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*** *import java.rmi.\*;*  *public class HelloClient {*  *public static void main (String [ ] argv) {*  */\* lancer SecurityManager \*/*  *System.setSecurityManager (new RMISecurityManager ()) ;*  *try {*  */\* trouver une référence vers l’objet distant \*/*  *HelloInterface hello=(HelloInterface) Naming.lookup("rmi://localhost/Hello") ;*  */\* appel de méthode à distance \*/*  *System.out.println (hello.sayHello()) ;*  *} catch (Exception e) {*  *System.out.println*  *(" Erreur client : " + e) ;*  *}*  *}* | \*\*\*\*\*\*\*\*\*\*\*\*Classe réalisant l’interface\*\*\*\*\*\*\*\*\* *import java.rmi.\*;*  *import java.rmi.server.\*;*  *public class Hello extends java.rmi.server.UnicastRemoteObject implements HelloInterface{*  *private String message;*  */\* le constructeur \*/*  *public Hello (String s)throws RemoteException*  *{*  *message = s ;} ;*  */\* l’implémentation de la méthode \*/*  *public String sayHello ()throws RemoteException{return message ;} ;*  *}* **\*\*\*\*\*\*\*\*\*\*\* Programme du serveur\*\*\*\*\*\*\*\*\*\*\*\*\*** *import java.rmi.\*;*  *public class HelloServer {*  *public static void main (String [ ] argv) {*    */\* lancer SecurityManager \*/*  *System.setSecurityManager (new RMISecurityManager()) ;*  *try {*  */\* créer une instance de la classe Hello et*  *l’enregistrer dans le serveur de noms \*/*  *Naming. rebind ("Hello",new Hello (" Hello world !")) ;*  *System. out. println (" Serveur prêt.") ;*  *} catch (Exception e) {System.out.println(" Erreur serveur : " + e) ;}*  *}* *}* |

|  |
| --- |
| **Version deux du programme server**  *import java.rmi.\*;*  *import java.rmi.server.\*;*  *public class Hello extends java.rmi.server.UnicastRemoteObject implements HelloInterface{*  *private String message;*  */\* le constructeur \*/*  *public Hello (String s)throws RemoteException*  *{ message = s ;} ;*  */\* l’implémentation de la méthode \*/*  *public String sayHello ()throws RemoteException{return message ;} ;*  *public static void main(String A[]){*  *//lancer security manager*  *System.setSecurityManager(new RMISecurityManager());*  *try{*  *//création d'une instance de la classe compte et l'enregistrer dans le serveur de noms*  *LocateRegistry.createRegistry(1199);*  *Naming. rebind ("Hello",****new Hello (" Hello world !")****) ;*  *System.out.println("Serveur pret");*  *}catch(Exception e){System.out.println("erreur serveur \t"+e); }}*  *}* |

# *Phase de compilation*

* *Sur la machine serveur : compiler les interfaces et les programmes du serveur*

*javac HelloInterface.java Hello.java HelloServer.java*

* *Sur la machine serveur : créer les talons client et serveur pour les objets appelés à distance (à partir de leurs interfaces) - ici une seule classe, Hello*

*rmic -keep Hello*

*N. B. cette commande construit et compile les talons client Hello\_ Stub. java et serveur Hello\_ Skel. java. L’option -keep permet de garder les sources de ces*

*Talons*

*The only problem one might encounter with this command is that rmic might not be able to find the files Hello.class and HelloInterface.class even though they are in the same directory where rmic is being executed. If this happens to you, then try setting the CLASSPATH environment variable to the current directory, as in the following command:*

***setenv CLASSPATH .***

*If your CLASSPATH variable already has some directories in it, then you might want to add the current directory to the others.*

*Ou rmic -d . Hello*

*Ou Add the current directory to your classpath.c-à-d* ***rmic –classpath c:/rmi Hello***

* ***S****ur la machine client : compiler les interfaces et le programme client*

*javac HelloInterface. java HelloClient. java*

###### Phase d’exécution

##### *Sécurité*

*La sécurité est importante lorsqu’il y a téléchargement de code (il peut être dangereux d’exécuter le code chargé depuis un site distant)*

###### Mise en oeuvre

*La politique de sécurité spécifie les actions autorisées, en particulier*

*sur les sockets*

* *Exemple de contenu du fichier java. policy*

*Permet d’utiliser les sockets comme indiqué. Toute autre utilisation est interdite*

***java.policy*** traduit la sécurité système

exemple de fichier gérant la sécurité système :

### grant {

### permission java.net.SocketPermission "\*:1024-65535","connect, accept";

### permission java.net.SocketPermission "\*:80","connect";

### permission java.security.AllPermission;

### };

* Sauvegarder ce code sous forme de ***java.policy et mettez dans le même répertoire contenant les autres classes***

*Compilation avec rmic :*

*The* ***rmic*** *compiler generates stub, and skeleton class files for remote objects. These classes files are generated from the compiled Java programming language classes that contain remote object implementations. A remote object is one that implements the interface java.rmi.Remote. The classes named in the* ***rmic*** *command must be classes that have been compiled successfully with the* ***javac*** *command and must be fully package qualified. For example, running* ***rmic*** *on the class file name HelloImpl as shown here: rmic hello.HelloImpl*

*creates the HelloImpl\_Skel.class and HelloImpl\_Stub.class files in the hello subdirectory (named for the class's package).*

### Start the Java RMI registry sur le port 1099

### start rmiregistry 1099

### Start the server

Sur une fenêtre command dos lancer la serveur

### java -Djava.security.policy=java.policy HelloServer

Sur une autre fenêtre command dos lancer le client

### Run the client

### java -Djava.security.policy=java.policy HelloClient

### Note : faire attention aux espace blanc

### The output from the client is the following message:

### Response: Hello, world!

Remarque: par fois lorsque vous compiler le HelloServeur.java vous aurez l'erreur suivant:

|  |
| --- |
| ava.rmi.ServerError: Error occurred in server thread; nested exception is: java.lang.UnsupportedClassVersionError: MyRemote : Unsupported major.minor version 51.0 at sun.rmi.server.UnicastServerRef.oldDispatch(Unknown Source) at sun.rmi.server.UnicastServerRef.dispatch(Unknown Source) at sun.rmi.transport.Transport$1.run(Unknown Source) at java.security.AccessController.doPrivileged(Native Method) at sun.rmi.transport.Transport.serviceCall(Unknown Source) at |

Solution:

### he error basically means you have compiled the classes in a higher version of Java than you are running them on. Make sure you are running the same JRE version or make sure you compile them against the version you want to run them on (using -target option of javac)

# En utilisant EDI Eclipse

# RMI Plug-in for Eclipse version 2.0 <http://www.genady.net/rmi/v20/downloads.html>

En utilisant EDI Netbean

***import java.rmi.\*;***

***import java.rmi.server.\*;***

***import java.rmi.registry.\*;***

***public class Hello extends java.rmi.server.UnicastRemoteObject implements HelloInterface***

***{***

***private String message;***

***/\* le constructeur \*/***

***public Hello (String s)throws RemoteException***

***{***

***message = s ;} ;***

***/\* l’implémentation de la méthode \*/***

***public String sayHello ()throws RemoteException{return message ;} ;***

***public static void main(String A[]){***

***//lancer security manager***

***//if(System.getSecurityManager() == null) System.setSecurityManager(new RMISecurityManager());***

***try{ Naming.rebind("Hello", new Hello("bonjour"));***

***System.out.println("Serveur pret");***

***}***

***catch(Exception e){***

***System.out.println("erreur serveur \t"+e);***

***}***

***}***

***}***

***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Client\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

***import java.rmi.\*;***

***public class HelloClient {***

***public static void main (String [ ] argv) {***

***/\* lancer SecurityManager \*/***

***// System.setSecurityManager (new RMISecurityManager ()) ;***

***try {/\* trouver une référence vers l’objet distant \*/***

***HelloInterface hello=(HelloInterface) Naming.lookup("rmi://localhost/Hello") ;***

***/\* appel de méthode à distance \*/***

***System.out.println (hello.sayHello()) ;***

***} catch (Exception e) {***

***System.out.println***

***(" Erreur client : " + e) ;***

***}}}***

***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Interface\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

***import java.rmi.\*;***

***public interface HelloInterface extends java.rmi.Remote {***

***/\* méthode qui imprime un message***

***prédéfini dans l’objet appelé \*/***

***public String sayHello() throws java.rmi.RemoteException;}***

* *Pour pouvoir utiliser les programme Javac, Java et rmiregistry à partir de n’importe qu’il répertoire, le système doit connaître leur chemin de Parcours utilisez la commande suivant pour fixer le chemin menant au path du répertoire \bin contenant les diverses programme de java.* 
  + set path=C:\Program Files\Java\jdk1.5.0\_01\bin;
* *Sur la machine serveur : compiler en utilisant netBean les interfaces et l’application serveur*
  + HelloInterface. java
  + Hello.java
  + HelloClient.java

Ces deux fichiers sont enregistrés par netbean dans

* C:\Documents and Settings\karim\rmireg\src

Les fichiers compiler

* + HelloInterface.class
  + Hello.class
  + HelloClient.class

Sont enregistrés dans:

* C:\Documents and Settings\karim\rmireg\build\classes

Avant de lancer le serveur il faut se positionner au répertoire :

* C:\Documents and Settings\karim\rmireg\build\classes> start rmiregistry

***Travail à faire :***

***L’objectif : Eécrire une application répartie permettant de gérer des comptes bancaires***

***Soit la classe compte suivante :***

***class Compte{***

***public int nbComptes;***

***protected float solde;***

***private int code;***

***public Compte(float soldeInitial){***

***nbComptes++;***

***code=nbComptes;***

***solde=soldeInitial;***

***}***

***public int getCode(){return code;}***

***public float getDecouvert(){return decouvert;}***

***public float getSolde(){ return solde;}***

***public void retirer(float mt){ if(mt<solde+decouvert)***

***solde-=mt; }***

***public void verser(float mt){ solde+=mt;}***

***public void setDecouvert(float d){decouvert=d;***

***}***

***}***

* ***Créer un compte en banque.***
* ***De verser un montant dans votre compte***
* ***Retirer un montant de votre compte***
* ***Fixer un découvert .***

***Les méthodes distantes sont :***

***float getSolde();***

***void verser(float mt);***

***void retirer(float mt);***

***float getDecouvert();***

1. ***Ecrire une interface Compte dérivant de Remote qui déclare les méthodes distantes.***
2. ***Ecrire une classe CompteImpl qui gère la partie serveur de notre application répartie.***
3. ***Ecrire une classe ClientBanque qui gère la partie client de notre application répartie.***
4. ***On veut maintenant que le serveur, prévienne le client quand le solde de son compte devient négatif ou inférieur à une valeur choisie par le client. Quel mécanisme, vu en cours, peut être utilisé ? Modifiez l’interface et les classes pour gérer cette situation.***
5. ***Compiler toutes les classes, générer les stubs (rmic CompteImpl …), récupérer le fichier*** [***java.policy***](file:///C:\AFDEL-2011\ESGSystemClientServeurRMI_banquejava.policy.txt)
6. ***Lancer le serveur :***

***java -Djava.security.policy=banque\java.policy compteImpl***

1. ***Lancer clients et tester l’application répartie.***

***Solution :***

***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*serveur\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

***import java.rmi.\*;***

***import java.rmi.server.\*;***

***import java.rmi.registry.\*;***

***class Compte extends java.rmi.server.UnicastRemoteObject implements InterfaceCompte{***

***private int nbComptes=0;***

***protected float solde;***

***protected float decouvert;***

***private int code;***

***public Compte(float soldeInitial) throws RemoteException{***

***nbComptes++;***

***code=nbComptes;***

***solde=soldeInitial;***

***}***

***public Compte(float soldeInitial,float \_decouvert)throws RemoteException{***

***nbComptes++;***

***code=nbComptes;***

***solde=soldeInitial;***

***decouvert=\_decouvert;***

***}***

***public int getCode() {return code;}***

***public float getSolde()throws RemoteException{return solde;}***

***public String retirer(float mt) throws RemoteException{ if(mt<solde+decouvert)***

***{solde-=mt;***

***return "";}***

***else return "vous avez depasse decouvert permis"; }***

***public void verser(float mt) throws RemoteException{solde+=mt;}***

***public void setDecouvert(float d){ decouvert=d; }***

***public float getDecouvert() throws RemoteException{return decouvert;}***

***public static void main(String A[]){***

***//lancer security manager***

***//System.setSecurityManager(new RMISecurityManager());***

***try{***

***//creation d'une instance de la classe compteet l'enregistrer dans le serveur de noms***

***LocateRegistry.createRegistry(1099);***

***Naming.rebind("compte",new Compte(100000));***

***System.out.println("Serveur pret");***

***}***

***catch(Exception e){***

***System.out.println("erreur serveur \t"+e);***

***}}}***

***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*client\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

***import java.rmi.\*;***

***public class ClientCompte{***

***public static void main(String A[]){***

***try{***

***//Trouver une reference vers l'objet Distant***

***InterfaceCompte IC=(InterfaceCompte)Naming.lookup("rmi://127.0.0.1/compte");***

***IC.verser(1009);***

***System.out.println(IC.retirer(90000000));***

***System.out.println("le solde est de : \t"+IC.getSolde());***

***}***

***catch(Exception e){***

***System.out.println("erreurs de clients \t"+e);***

***}***

***}***

***}***

***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*InterfaceCompte\*\*\*\*\*\*\*\*\*\*\*\*\****

***import java.rmi.\*;***

***public interface InterfaceCompte extends java.rmi.Remote {***

***public float getSolde()throws java.rmi.RemoteException;***

***public String retirer(float mt)throws java.rmi.RemoteException;***

***public void verser(float mt)throws java.rmi.RemoteException;***

***public float getDecouvert()throws java.rmi.RemoteException;***

***}***