Java RMI

1 Les outils et concepts

Question 1. Autour de la génération de stubs et de squelettes.

a- Quel est le rôle de l'outil `rmic` ?

b- Qu'est-ce que la génération dynamique de stubs et de squelettes ?

c- Quel est le rôle de l'outil `rmiregistry` ?

2 Hello world

On s'intéresse à l'exemple Hello World dont le code est donné ci-dessous.

Question 2. Donner la procédure exacte pour lancer le serveur puis le client.

Question 3. Que se passe t-il si on remplace la ligne 15 de `Server.java` par la ligne 14 (commentée) ?

Question 4. Donnez les affichages chez le client et chez le serveur.

Question 5. Dans quelle JVM seront créés les objets de type `HelloImpl` ?

Question 6. À quoi sert l'interface `Hello.java` ?

Question 7. Donnez des exemples d'exceptions pouvant être attrapées à la ligne 22 de `Server.java`.

Question 8. Quelle est la différence entre `Naming.bind` et `Naming.rebind` ?

Question 9. Que se passe t-il si on ne passe pas d'argument en ligne de commande quand on lance le client ?

Listing 1 – Hello.java

```
1  package helloWorld;
2  import java.rmi.Remote;
3  import java.rmi.RemoteException;
4  public interface Hello extends Remote{
5      String sayHello() throws RemoteException;
6      void printHello() throws RemoteException;
7  }
```

Listing 2 – HelloImpl.java

```
1  package helloWorld;
2
3  import java.rmi.RemoteException;
4  import java.rmi.server.UnicastRemoteObject;
5
6  public class HelloImpl extends UnicastRemoteObject  implements Hello {
7
8      public HelloImpl() throws RemoteException{
9      }
10
11     public String sayHello() throws RemoteException{
12         return "Hello, world!";
13         }
14
15     public void printHello() throws RemoteException {
16         System.out.println("The server prints: Hello, world!");
17     }
18 }
```

Listing 3 – Server.java

```
1  package helloWorld;
2  import java.rmi.registry.Registry;
3  import java.rmi.registry.LocateRegistry;
4
5  public class Server {
6
7      public Server() {}
8
9
10     public static void main(String args[]) {
11
12         try {
13             HelloImpl obj = new HelloImpl();
14             // Registry registry = LocateRegistry.createRegistry(1099);
15             Registry registry = LocateRegistry.getRegistry();
16             if (registry==null){
17                 System.err.println("RmiRegistry not found");
18             }else{
19                 registry.bind("Hello", obj);
20                 System.err.println("Server ready");
21             }
22         } catch (Exception e) {
23             System.err.println("Server exception: " + e.toString());
24             e.printStackTrace();
25         }
26     }
27 }
```

Listing 4 – Client.java

```
1  package helloWorld;
2
3  import java.rmi.registry.LocateRegistry;
4  import java.rmi.registry.Registry;
5
6  public class Client {
7      private Client() {}
8
9      public static void main(String[] args) {
10
11         String host = (args.length < 1) ? null : args[0];
12         try {
13             Registry registry = LocateRegistry.getRegistry(host);
14             Hello stub = (Hello) registry.lookup("Hello");
15             String response = stub.sayHello();
16             System.out.println("response: " + response);
17             stub.printHello();
18         } catch (Exception e) {
19             System.err.println("Client exception: " + e.toString());
20             e.printStackTrace();
21         }
22     }
23 }
```

3 Gestion de comptes pour un restaurant administratif

On cherche à écrire une application pour la gestion des comptes des utilisateurs d'un restaurant administratif. Les utilisateurs payent leurs repas à l'une des caisses, grâce à une carte dédiée. Ils peuvent recharger leur compte du montant de leur choix à ces mêmes caisses. Un découvert de 8 euros est autorisé. Si le crédit d'un compte descend sous 7 euros, on notifie l'utilisateur lors du passage en caisse qu'il est temps de recharger son compte. On met en place cette application avec RMI. On placera un client par caisse. Les clients accèderont au serveur pour débiter et créditer le compte des différents utilisateurs. Le serveur enverra une alerte au client quand le crédit devient trop bas, le client se chargera de transmettre l'information à l'utilisateur (en pratique l'alerte s'imprime sur le ticket de caisse). Cette alerte sera émise par le serveur vers le client, à l'initiative du serveur. On ne s'intéresse pas à l'interface avec le lecteur de carte, ni au moyen de payement réel.

Question 10. Dans un premier temps, on ne s'intéresse pas au système d'alerte. Quelle infrastructure (classes, interfaces, méthodes) proposez-vous de mettre en place ?

Question 11. Comment faire pour mettre en place le système d'alerte ?