Codes mobiles en Java
– mise en œuvre guidée –

Serge Chaumette

6 octobre 2004

Table des matières

Chapitre 1

Guide de mise en œuvre

Etape 0.
Communication client-serveur à base de sockets.

Question 1. Faire fonctionner l'application calculette (cf. 2 page 5) qui illustre un fonctionnement client/serveur à base de sockets.

Etape 1.
Sérialisation/désérialisation dans un fichier.

Question 2. Faire fonctionner l'exemple de sérialisation (cf. 3 page 7) qui illustre une sérialisation/désérialisation dans un fichier. L'objet sérialisé dans cet exemple est une instance de la classe Evenement développée comme simple illustration d'un objet sérialisable.

Question 3. Créez, compilez et testez la classe ClassDemo ci-dessous.

— Program 1 : ClassDemo.java ——————————————————

```
1
2    public class ClasseDeDemo implements java.io.Serialisable{
3
4            private int value;
5
6            public ClasseDeDemo(){
7
8                    value=0:
9
10                   System.out.printn("Classe ClassDeDemo : constructeur exécuté");
11           }
12
13           public void setValue(int newValue){
14
15                   value=newValue:
16           }
17
18           public int getValue(){
19
20                   return value;
21           }
22   }
```

— Program 1 : ClassDemo.java —

Question 4. Modifiez l'application de la question 2 pour sérialiser/désérialiser une instance de la class ClassDemo. Appelez setValue avant la sérialisation de sorte à donner une valeur au champ value. Après désérialisation, appelez getValue pour faire afficher la valeur du champ value dans l'objet désérialisé. L'objectif est de montrer que la valeur du champs value est bien conservée par l'opération de sérialisation, c.a.ds que l'état de l'objet est intégré dans sa sérialisation.

Etape 2.
Echange d'objets à travers le réseau.

On souhaite réaliser une fonctionnalité tout à fait comparable si ce n'est que le client va envoyer un objet Java vers le serveur, objet que le serveur devra

recevoir et reconstruire. On va pour cela utiliser le mécanisme de sérialisation de Java et la communication par sockets.

On utilisera l'application calculette que l'on modifiera pour atteindre notre objectif.

Question 5. Dans l'exemple de calculette identifiez côté client les lignes qui réalisent l'envoi effectif de données (2 entiers) vers le serveur et la réception du résultat (1 entier).

Ce sont les lignes qu'il faudra modifier pour envoyer un objet.

Question 6. Dans l'exemple calculette identifiez côté serveur les lignes qui réalisent la réception effective de données (deux entiers) depuis le client puis l'envoi du résultat (1 entier).

Ce sont ces lignes qu'il faudra modifier pour recevoir un objet.

Question 7. Comment obtient-t-on le flot disponible en écriture sur un socket connecté ?

Question 8. Quelle est la classe de flot dans laquelle on peut écrire un objet (Object) ?

Question 9. Comment construit-on un ObjectOutputStream a partir d'un OutputStream ?

Question 10. Construire un ObjectOuptutStream permettant d'envoyer un objet vers une connexion réseau.

Rappel : un flot de sortie accepte des données en entrée et les envoie vers un collecteur. On peut chaîner les flots, un flot jouant alors le rôle de collecteur pour le flot qui le précède dans la chaîne. Par exemple un FileOutputStream est un flot qui écrit dans un collecteur de type OutputStream.

Question 11. Comment sérialise-t-on un objet dans un ObjectOutputStream ?

Question 12. Quelle est la classe de flot dans laquelle on peut lire un object (Object) ?

Question 13. Comment construit-on un ObjectInputStream a partir d'un InputStream ?

Question 14. Construire un ObjectOuptutStream permettant de recevoir un Objet depuis une connexion réseau.

Question 15. Comment désérialise-t-on un objet depuis un ObjectInputStream ?

Question 16. Réalisez une application permettant d'échanger à travers le réseau un objet de la classe ClassDemo.

Appelez setValue avant la sérialisation de sorte à donner une valeur au champ value. Après désérialisation, appelez getValue pour faire afficher la valeur du champ value dans l'objet désérialisé. L'objectif est de montrer que la valeur du champs value est bien conservée par l'opération de sérialisation/transport réseau, c.a.d. que l'état de l'objet est intégré dans sa sérialisation.

Etape 3.
Système d'agents mobiles.

Un agent mobile est un objet doté d'une autonomie de comportement. Il doit donc disposer d'une méthode allerSur qu'il invoque pour se déplacer et qui le sérialise dans une connexion réseau l'envoyant vers un serveur.

A son arrivée sur le serveur, il est réactivé par celui-ci. Pour cela, l'agent implémente une méthode arriveDe invoquée par le serveur, après qu'il l'ait désérialisé.

Question 17. Ecrire une application dans laquelle un agent mobile fait le tour de plusieurs machines (serveurs) et retourne le total de mémoire libre des machines visitées et la machine ayant le plus de mémoire libre.

Chapitre 2

Application de communication par sockets

— Program 2 : sockets/exemples/sum/Client.java ——————————

```
1   import java.net.Socket;
2   import java.io.InputStream;
3   import java.io.OutputStream;
4
5   public class Client{
6
7     static public void main(String args[])
8          throws java.io.IOException{
9
10           String nomDuServeur=new String(args[0]);
11           int numeroDePort=Integer.parseInt(args[1]);
12
13           Socket socket= new Socket(nomDuServeur, numeroDePort);
14
15           InputStream entree=socket.getInputStream();
16           OutputStream sortie=socket.getOutputStream();
17
18           sortie.write(Integer.parseInt(args[2]));
19           sortie.write(Integer.parseInt(args[3]));
20           System.out.println(entree.read());
21    }
22  }
```

Program 2 : sockets/exemples/sum/Client.java

Program 3 : sockets/exemples/sum/Server.java

```
1   import java.net.ServerSocket;
2   import java.net.Socket;
3   import java.io.InputStream;
4   import java.io.OutputStream;
5
6   public class Server{
7
8     static public void main(String args[])
9          throws java.io.IOException{
10
11           int numeroDePort=Integer.parseInt(args[0]);
12
13           ServerSocket serverSocket= new ServerSocket(numeroDePort);
14
15           while (true){
16
17               Socket socket=serverSocket.accept();
18
19               InputStream entree=socket.getInputStream();
20               OutputStream sortie=socket.getOutputStream();
21
22               int entier1=entree.read();
23               int entier2=entree.read();
24               sortie.write(entier1+entier2);
25           }
26       }
27  }
```

Program 3 : sockets/exemples/sum/Server.java

Chapitre 3

Application de sérialisation

— Program 4 : serialisation/exemples/basique/Evenement.java ————————

```
1   import java.io.*;
2   import java.util.Date;
3
4   public class Evenement implements Serializable{
5
6       private Date dateDebut;
7       private Date dateFin;
8
9       public Evenement(){
10      }
11
12      public void debut(){
13          dateDebut=new Date();
14      }
15
16      public void fin(){
17          dateFin=new Date();
18      }
19
20      public String toString(){
21          return new String(dateDebut + " <-----> " + dateFin);
22      }
23  }
```

Program 4 : serialisation/exemples/basique/Evenement.java

Program 5 : serialisation/exemples/basique/Sauver.java

```
1   import java.io.*;
2   import java.util.Date;
3
4   public class Sauver{
5
6       static public void main(String args[])
7           throws java.io.IOException {
8
9
10              Evenement evenement=new Evenement();
11              evenement.debut();
12              for (int i=0; i<1000; i++) java.lang.System.gc();
13              evenement.fin();
14
15
16              FileOutputStream fos=new FileOutputStream("evenement.ser");
17
18              ObjectOutputStream oos=new ObjectOutputStream(fos);
19              oos.writeObject(evenement);
20              oos.flush();
21              oos.close();
22              fos.close();
23          }
24  }
25
```

Program 5 : serialisation/exemples/basique/Sauver.java

Program 6 : serialisation/exemples/basique/Restaurer.java

```
1   import java.io.*;
2   import java.util.Date;
3
4   public class Restaurer{
5
6       static public void main(String args[])
```

```
7           throws java.io.IOException {
8
9               FileInputStream fis=new FileInputStream("evenement.ser");
10
11
12              ObjectInputStream ois=new ObjectInputStream(fis);
13              try{
14                  Evenement evenement = (Evenement)ois.readObject();
15                  System.out.println(evenement);
16              } catch (java.lang.ClassNotFoundException ex){};
17              ois.close();
18              fis.close();
19
20          }
21  }
```

Program 6 : serialisation/exemples/basique/Restaurer.java