TP2 sur les sockets
serveur reposant sur des objets

1 Protocoles d'objets sérialisés

De façon générale, les programmeurs considèrent que la programmation via les sockets est de bas niveau. Java a nettement facilité l'utilisation des sockets par rapport à d'autres langages de programmation. La communication via des sockets propose toujours un flot d'octets non structurés entre les deux extrémités.
Pour effectuer propremententdes communication via des sockets, il faut proposer un protocole définissant les données envoyées et reçues. la partie la plus complexe concerne la façon d'empaqueter les données pour les envoyer sur le réseau et de les dépaqueter de l'autre côté (emballage et déballage). Les classes DataInputStream et DataOutputStream résolvent ce problème pour ds types simples. Mais en réaliter nous devons être capables de rassembler des objets simples dans de plus grandres structures.

En réalité tous les types manipulés ne sont pas forcément simples . Nous devons devons donc être capables de rassembler des objets simples dans de plus grandres structure. La **sérialisation en java** permet de résoudre ce problème.
‘

Définition 1 Classe sérialisable *Une classe est sérialisable si elle implémente java.io.Serializable.*

TP 1 Réalisation de serveurs reposant sur des objets.
Pour aborder cette partie un peu délicate, on vous demande dans un premier temps d'analyser et de tester l'application client serveur dont le code est donné en annexe2(code tiré du livre Introduction à Java, P. Niemeyer et J. Knudsen, Editions O'Reilly). Prenez soin de découper la serveur en dégageant bien la partie commune à tous les serveurs parallèles (Serveur.java vu plus haut) et la partie Application.java.

TP 2 Vers un serveur de calcul
On vous demande de faire évoluer le serveur précédent comme suit :

1. *Le serveur est capable de calculer la somme de deux nombres*
2. *Le serveur est capable de calcul le factoriel d'un entier n.*
3. *Le serveur est capable de résoudre une équation du second degré.*
4. *Le serveur est capable de réaliser le produit deux vecteurs.*
5. *Le serveur est capable de réaliser le produit de deux vecteurs.*
6. *Le serveur est capable de réaliser la somme de deux matrices*
7. *Le serveur est vabale de réaliser le produit de deux matrices*
8.
9. *...*

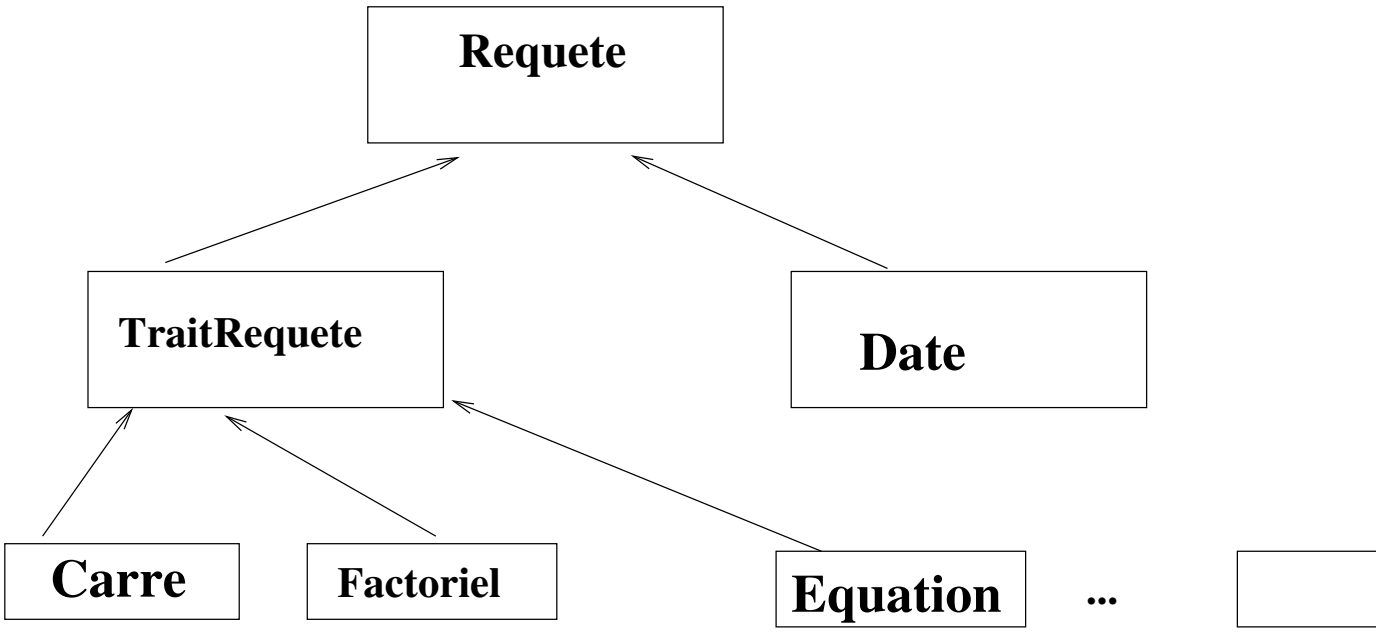

FIGURE 1 – Serveur reposant sur des objets.

Annexe

```
// Classe requete
public class Requete implements java.io.Serializable{}

// Classe Traitrequete
public  abstract class TraitRequete extends Requete {
public abstract Object execute();
}

// Classe Date
 public   class Date  extends Requete {}

// Classe Carre
public class Carre extends TraitRequete {
int n;

public Carre (int n) {
this.n =n;
}

public Object execute() {
return new Integer (n*n);
}
}

// Client

import java.io.*;
```

```
import java.net.*;
import java.util.*;

public class Client {
public static void main(String argv[]) {
try{
Socket emission = new Socket(argv[0], Integer.parseInt(argv[1]));

// System.out.flush();


ObjectOutputStream  out =
new ObjectOutputStream(emission.getOutputStream());
ObjectInputStream in  =
new ObjectInputStream(emission.getInputStream());



System.out.println("Connexion établie ");


// Demande de la date
System.out.println(" Demande de la date ");
out.writeObject(new Date());
out.flush();
System.out.println(in.readObject());
System.out.flush();

//  Lancer le calcul du carré
System.out.println(" Calcul du carré ");
System.out.flush();

out.writeObject(new Carre(2));
out.flush();

// Résultat du carré
System.out.println(" Résultat du  carré ");
System.out.flush();

System.out.println(in.readObject());



emission.close();
out.flush();
} catch( IOException  e) {
System.out.println("Erreur d'entree sortie " +e );
```

```
}   catch(ClassNotFoundException e2) {
System.out.println(e2);
}
}
}
```

```
//Serveur
   import java.io.*;
import java.net.*;
import java.util.*;
public class Serveur {
public static void main(String argv[])throws IOException  {
ServerSocket ecoute = new ServerSocket(Integer.parseInt(argv[0]));
System.out.println("Le serveur reçoit sur le port: " + ecoute.getLocalPort());

while(true)
new Application(ecoute.accept()).start();
}
}// Fin de la classe serveur
```

```
     import java.io.*;
import java.net.*;
import java.util.*;
class Application extends Thread {
Socket client;
Application (Socket client ) throws SocketException {
this.client = client;
}

public void run() {
try  {
ObjectInputStream in  =
new ObjectInputStream(client.getInputStream());
ObjectOutputStream  out =
new ObjectOutputStream(client.getOutputStream());
while(true) {
out.writeObject(traiteRequete(in.readObject()));
out.flush();}
}  catch(EOFException e3) {//  Fin de fichier normale
try{
client.close();
}  catch( IOException  e) { }
}  catch(IOException e) {
System.out.println("Erreur d'entree sortie " +e );
}  catch(ClassNotFoundException e2) {

System.out.println(e2); // Le type de l'objet demandé est inconuu
}
```

```
}
private Object traiteRequete(Object requete) {

if (requete instanceof Date )
return new java.util.Date();
else if (requete instanceof Carre)
return ((Carre)requete ).execute();
else if  (requete instanceof Somme )
return ((Somme)requete).execute();
else return null;
}
}
```