Sockets et Flux

Dans ce TP les données ont été envoyées comme étant des octets. Si cela fonctionne bien avec des fichiers quelconques (texte ou image par exemple), ce n'est pas facile d'utilisation si on transmet des données formatées, comme des objets ou des chaînes de caractères. Java offre la possiblité de spécialiser des flux d'octets qui sont à utiliser pour lire et écrire les données formatées.

Flux de chaîne de caractères

L'utilisation de flux spécialisés dans les chaînes de caractères est très utililisé dans des applications standards de l'internet, comme le mail (SMTP), le web (HTTP) ou le transfert de fichier (FTP). Les flux java spécialisés pour le traitement des chaines sont les classes `java.io.Input-StreamReader`, `java.io.BufferedReader` et `java.io.PrintWriter`[1].

Dans ce TP, on ne réalisera en premier que le programme recevant des chaînes de caractères qu'il devra afficher à l'écran. Les programmes émetteurs seront les navigateurs internet à disposition. Pour que le navigateur envoient à votre programme ses données, il faut vérifier qu'un proxy ne sera pas utilisé pour le réseau local ou pour la machine courante (localhost). Enfin, deux cas sont possibles :

- Le récepteur attend sur le port 80 et l'adresse à mettre dans le navigateur sera : `http ://localhost`.
- Le récepteur attend sur un autre port, par exemple le 8080 et l'adresse à mettre sera : `http ://localhost :8080`

Losrque vous affichez toutes les lignes que génèrent les navigateurs, vous pouvez constater que le navigateur attend une réponse indéfiniement, jusqu'à indiquer une erreur. Le programme récepteur peut lui envoyer une simple chaîne de caractères comme :
`<HTML>Hello World</HTML>`
qui s'affichera sur le navigateur. Pour que cela fonctionne, il faut retourner cette chaîne de caractères dès que la première ligne envoyée par le navigateur est reçue par votre programme.

Flux d'objets

Les classes `java.io.ObjectInputStream` et `java.io.ObjectOutputStream` représentent des flux spécialisés dans l'émission et la réception d'objets java. Une contrainte sur ces objets pouvant ainsi être déplacés d'un programme à l'autre est qu'il doivent implémenter l'interface `java.io.Serializable`[2].

Créer une classe simple, contenant un entier publique, et qui soit sérialisable. En vous inspirant des programmes précédents, faites un programme créant une instance de votre classe simple

1. Il en existe d'autres, mais ici nous avons limités celles qui traitent des lignes de caractères se terminant par un "\n".
2. Ce qui n'entraîne pas plus de code car cette interface ne définit aucune méthode !

et l'envoyant à un autre programme via une connexion TCP. Le programme recevant l'objet devra afficher l'entier qu'il contient. Noter que la méthode de lecture d'un objet `readObject()` de la classe `java.io.ObjectInputStream` retourne une intance de la classe `Object`. Il faudra donc faire un "cast" de cet objet reçu avec la classe simple.

Ci dessous un exemple de classe simple :

```
import java.io.Serializable;

public class WalkingClass implements Serializable {

        private int count;

        public WalkingClass(int c){
                count = c;
        }
        public void inc(){
                count++;
        }
        public void dec(){
                count--;
        }
        public int getCount(){
                return count;
        }
}
```