TRAVAUX DIRIGES 4 Informatique : JAVA IUT R&T FA2

A. Flux réseaux URL

1) URL *openConnection()*

Compléter ce programme selon les commentaires, puis donner le résultat de son exécution.

%%

```
import java.io.*;
import java.net.*;
public class Test_openConnection
 {
   public static void main (String arg[])
   { new Test_openConnection (arg[0]); }
   public Test_openConnection (String nomPage)
    {
      try
      { URL u = new URL (nomPage);
        URLConnection c = u.openConnection ();

        System.out.println ("Je charge " + nomPage);
        BufferedReader din  = new BufferedReader(new InputStreamReader (c.getInputStream () );
        PrintStream    pout = new PrintStream (new FileOutputStream ("resultat.html"));

        String s = "nonnulle";
        while (s != null)
         {
…………………………………………………………………………//récupérer dans s chaque ligne lue du flux d'entrée par readLine()
            System.out.print (".");
…………………………………………………………………………//écrire la chaine s dans le fichier "resultat.html" par println(…)
         }
        pout.close();
        din.close();
       }
      catch (Exception e )
      { System.out.println (e.getMessage ());
         e.printStackTrace ();
       }
    }
 }
```

%%

2) URL *openConnection()*

A partir d'un navigateur WEB, taper cette URL ***http://api.macvendors.com*** et concaténer avec « / » puis avec une adresse MAC (ex : ***00:11:43:00:00:01***) vous obtiendrez en retour « DELL INC. ». Le résultat de la concaténation est « ***http://api.macvendors.com/00:11:43:00:00:01*** ». Le résultat de l'exécution de cette API correspond effectivement au nom du constructeur de l'équipement dont l'adresse MAC est ***00:11:43:00:00:01***.
Dans un second temps, vous devrez afficher le nom du constructeur d'un équipement donné par son adresse MAC saisie au clavier. Pour cela, vous ferez appel aux méthodes :

- ***openConnection()*** pour la connexion à un URL.
- ***getInputStream()*** pour récupérer le flux d'entrée retourné par cette API.
- ***readLine()*** pour lire une ligne de ce flux d'entrée (utiliser pour cela ***BufferedReader*** et ***InputStreamReader***).

3) Exercices complémentaires

a) Compléter ce programme selon les commentaires, puis donner le résultat de son exécution.

%%

```
import java.net.*;
import java.io.*;
public class URLReader {
public static void main (String[] args) throws Exception {
    URL google = new URL("http://www.lemonde.fr/");
```

```
// passer en paramètre d'entrée du constructeur InputStreamReader le flux d'entrée créé par l'appel de la
// méthode openStream() associé à l'URL donné
      BufferedReader in = new BufferedReader( new InputStreamReader( ……………..………………… ) );
      String inputLine;
      while ((inputLine = in.readLine()) != null) System.out.println( inputLine );
      in.close();
  }
}
```

%%

B. Flux réseaux Socket

1) Socket TCP (Mode connecté)

Saisir ces deux programmes dans deux fichiers Java différents. Les exécuter en même temps, puis donner le résultat de leur exécution.

%%

```
public class Serveur
{   public static void main (String args[])  {
        try {
             int port ;
             if (args.length > 0) port=Integer.parseInt( args[0] );
             ServerSocket ss = new ServerSocket(port, 5);
             System.out.println("Le serveur reçoit sur le Port : " + ss.getLocalPort());

             System.out.println(" Prêt !");
             Socket client = ss.accept();
             System.out.print("   Connexion reçue de: ");
             System.out.println(" " + client.getInetAddress());

             BufferedReader in = new BufferedReader (new InputStreamReader(client.getInputStream()));
             PrintWriter out = new PrintWriter  (new OutputStreamWriter(client.getOutputStream()));
             out.println("Allo, quelqu'un au bout du fil ?"); out.flush();

             String line = in.readLine();
             System.out.println("> Recu: " + line) ;
             out.println("Echo: " + line);
             out.println("... patati..." );
             out.println("   ...patata..." );
             out.flush();
             out.close(); client.close(); ss.close();   //fermeture des sockets
         }
        catch (Exception e) { System.err.println(e); System.err.println("Lancer l'appli serveur: java Serveur <port>");
        }
  }
}
```

%%

```
public class Client  {
   public static void main(String[] args) throws IOException
           {  int    port = Integer.parseInt(args[0]);
             String host;

             if (args.length < 2)
                     host = InetAddress.getLocalHost().getHostName();
            else
                     host = args[1];

            Socket sc = new Socket(host, port);

            BufferedReader in = new BufferedReader(new InputStreamReader(sc.getInputStream()));

            PrintWriter out = new PrintWriter(new OutputStreamWriter(sc.getOutputStream()));

            out.println("Allo");
```

```
            out.flush();

            String line;
            while((line = in.readLine()) != null)
            {           System.out.println(line);
            }
            sc.close();
   }
}
```

%%

2) Socket TCP (Mode connecté)

Proposer un programme permettant à un client de récupérer le contenu d'un fichier qu'il aura choisi, il aura envoyé au serveur le nom du fichier dont il souhaite récupérer son contenu.

Côté client :

- récupérer une chaine de car. saisie au clavier correspondant au nom d'un fichier dont il veut récupérer son contenu situé sur le serveur,
- envoyer au serveur le nom du fichier (la chaine de car.),
- réceptionner tout le contenu du fichier envoyé par le serveur,
- le client devra enregistrer ce qu'il reçoit dans un fichier.

Côté serveur :

- récupérer le nom du fichier (chaine de car.) envoyé par le client.
- vérifier si le nom de fichier reçu est contenu dans le tableau de noms de fichiers (présents sur le serveur).
- Si le lire puis envoyer le contenu du fichier au client,

C. Socket et interface Serializable

1) Socket et envoi/réception d'objets

Reprendre le programme du TD3 stockant sur disque dur des objets ***User*** (avec ***Serializable***) contenant un nom, prénom, numéro de tél, email et profession. Vous utilisiez :

- pour l'envoi : la classe ***ObjectOutputStream*** et la méthode ***writeObject(…)*** ,
- pour la réception : la classe ***ObjectInputStream*** et la méthode ***readObject(…)*** .

Réaliser une application client/serveur en mode connecté à travers deux programmes (un pour le client et un pour le serveur).

Côté client :

a) récupérer une chaine de car. saisie au clavier correspondant au nom d'un ***User*** à rechercher,
b) envoyer au serveur le nom recherché (la chaine de car.) d'un ***User***,
c) réceptionner la réponse du serveur, une valeur booléenne, correspondant au résultat de la recherche du nom recherché du ***User***,
d) Si le nom recherché du User a été trouvé, réceptionner et afficher les données membres de ce ***User*** trouvé.

Côté serveur :

a) charger le fichier contenant tous les ***User*** et les copier dans un tableau de ***User***.
b) réceptionner le nom d'un ***User*** à rechercher envoyé par le client,
c) rechercher un ***User*** ayant le nom recherché dans le tableau de ***User***,
d) retourner la valeur booléenne correspondant au résultat de la recherche,
e) si le nom recherché a été trouvé, retourner le premier objet ***User*** dont le nom correspond à celui recherché.

2) Exercices complémentaires

a) Reprendre l'exercice précédent pour que cette application client/serveur puisse proposer l'accès simultané de différents clients à cet unique serveur. Pour cela, utilisez la classe ***Thread***.

b) Reprendre l'exercice précédent pour que cette application client/serveur puisse proposer l'accès simultané de différents clients à ce serveur. Pour cela, utilisez l'interface ***Runnable***.