Correction Série 5
Les sockets

Exercice 1:

Question 1: Le processus client commence par émettre un message et le serveur lui répond par un écho de cette ligne en majuscule.

```
import java.io.*;
import java.net.*;
public class Serveur {
  static final int port = 1200;
  public static void main(String[] args) throws Exception {
     // Création d'un objet s à l'écoute du port spécifié
     ServerSocket s = new ServerSocket(port);
     System.out.println("En attente de connexion");
     Socket socClient = s.accept(); // L'acceptation d'une connexion d'un client
     System.out.println("Connexion établie");
     // On crée maintenant les flux d'entrée-sortie du Serveur
          BufferedReader entreeServeur = new BufferedReader(
                    new InputStreamReader(socClient.getInputStream())
                   );  // Un BufferedReader permet de lire par ligne.
          PrintWriter sortieServeur = new PrintWriter(
                   new BufferedWriter(
                     new OutputStreamWriter(socClient.getOutputStream())),
                   true);  // Un PrintWriter possède toutes les opérations print classiques.
     String str = entreeServeur.readLine(); // lecture du message envoyé par le client
     String strMajuscule=str.toUpperCase();
     System.out.println("Le message en majuscule est " + strMajuscule);
```

```
        // Fermeture des flux d'entrée-sortie du Serveur
        entreeServeur.close();
        sortieServeur.close();
        // Fermeture du socket client
        socClient.close();
    }
}

import java.io.*;
import java.net.*;

public class Client {
    static final int port = 1200;
    public static void main(String[] args) throws Exception {

        System.out.println("Demande de connexion");
        Socket socket = new Socket("127.0.0.1", port); // construit un socket client connecté à la
                                                       // machine et le port spécifié
        System.out.println("Connexion établie");
        // On crée maintenant les flux d'entrée-sortie du Client
        BufferedReader entreeClient = new BufferedReader( new
InputStreamReader(socket.getInputStream())  ); //Un BufferedReader permet de lire par ligne.
        PrintWriter sortieClient = new PrintWriter( new BufferedWriter( new
OutputStreamWriter(socket.getOutputStream())), true); //Un PrintWriter possède toutes les
                                                       // opérations print classiques.
        String str = "bonjour";
        sortieClient.println(str); // envoi d'un message
        // Fermeture des flux d'entrée-sortie du Client
        entreeClient.close();
        sortieClient.close();
```

```
        // Fermeture du socket client
        socket.close();
    }
}
```

NB:

- Il faut mettre chacune des classe Serveur et Client dans des packages différents ou des projets différents.
- Vous devez aussi lancer le Serveur, qui reste en écoute, puis vous pouvez lancer le Client.

Question 2: Au bout de 10 échanges, le client envoie un message de terminaison ‘END’ et ferme la connexion.

```
import java.io.*;
import java.net.*;
public class Serveur {
  static final int port = 1200;
  public static void main(String[] args) throws Exception {
      // Création d'un objet s à l'écoute du port spécifié
      ServerSocket s = new ServerSocket(port);
      System.out.println("En attente de connexion");
      Socket socClient = s.accept(); // L'acceptation d'une connexion d'un client
      System.out.println("Connexion établie");

      // On crée maintenant les flux d'entrée-sortie du Serveur
          BufferedReader entreeServeur = new BufferedReader(
                  new InputStreamReader(socClient.getInputStream())
                ); // Un BufferedReader permet de lire par ligne.
```

```
        // Un PrintWriter possède toutes les opérations print classiques.
        PrintWriter sortieServeur = new PrintWriter(
                        new BufferedWriter(
                          new OutputStreamWriter(socClient.getOutputStream())),
                        true);
        while (true) {
          String str = entreeServeur.readLine(); // lecture du message envoyé par le client
          if (str.equals("END")) break;
          System.out.println("ECHO = " + str);
          sortieServeur.println(str); // renvoi d'un écho au client
        }
        // Fermeture des flux d'entrée-sortie du Serveur
        entreeServeur.close();
        sortieServeur.close();
        // Fermeture du socket client
        socClient.close();
    }
}

import java.io.*;
import java.net.*;
public class Client {
   static final int port = 1200;
   public static void main(String[] args) throws Exception {

        System.out.println("Demande de connexion");
        Socket socket = new Socket("127.0.0.1", port); // construit un socket client connecté à la
machine et le port spécifié
        System.out.println("Connexion établie");
```

```
        BufferedReader entreeClient = new BufferedReader( new
InputStreamReader(socket.getInputStream())  );
        PrintWriter sortieClient = new PrintWriter( new BufferedWriter( new
OutputStreamWriter(socket.getOutputStream())), true);
        String str = "bonjour";
        for (int i = 0; i < 10; i++) {
          sortieClient.println(str);          // envoi d'un message
          str = entreeClient.readLine();      // lecture de l'écho
        }
        System.out.println("END"); // message de terminaison
        sortieClient.println("END") ; // Le client envoye le message de terminaison "END" au
Serveur
        // Fermeture des flux d'entrée-sortie du Client
        entreeClient.close();
        sortieClient.close();
        // Fermeture du socket client
        socket.close();
    }
}
```

Exercice 2:

Question 1: Le serveur doit envoyer la date au client qui doit l'afficher.

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

public class Serveur {
   static final int port = 1200;
   public static void main(String[] args) throws Exception {
```

```
        // Création d'un objet s à l'écoute du port spécifié
        ServerSocket s = new ServerSocket(port);
        System.out.println("En attente de connexion");
        Socket socClient = s.accept(); // L'acceptation d'une connexion d'un client
        System.out.println("Connexion établie");
        // On crée maintenant les flux d'entrée-sortie du Serveur
        // Un BufferedReader permet de lire par ligne.
        BufferedReader entreeServeur = new BufferedReader(
                        new InputStreamReader(socClient.getInputStream())
                       );
        // Un PrintWriter possède toutes les opérations print classiques.
        PrintWriter sortieServeur = new PrintWriter(
                       new BufferedWriter(
                         new OutputStreamWriter(socClient.getOutputStream())),
                       true);
        sortieServeur.println(new Date());
        // Fermeture des flux d'entrée-sortie du Serveur
        entreeServeur.close();
        sortieServeur.close();
        // Fermeture du socket client
        socClient.close();
    }
}

import java.io.*;
import java.net.*;

public class Client {
   static final int port = 1200;
```

```
  public static void main(String[] args) throws Exception {
     System.out.println("Demande de connexion");
     Socket socket = new Socket("127.0.0.1", port); // construit un socket client connecté à la
                                                    //machine et le port spécifié
     System.out.println("Connexion établie");
     // On crée maintenant les flux d'entrée-sortie du Serveur
      // Un BufferedReader permet de lire par ligne.
      BufferedReader entreeClient = new BufferedReader( new
InputStreamReader(socket.getInputStream())  );
      // Un PrintWriter possède toutes les opérations print classiques.
      PrintWriter sortieClient = new PrintWriter( new BufferedWriter( new
OutputStreamWriter(socket.getOutputStream())), true);
      System.out.println("La date envoyé par le serveur est : "+entreeClient.readLine());
      // Fermeture des flux d'entrée-sortie du Client
      entreeClient.close();
      sortieClient.close();
      // Fermeture du socket client
      socket.close();
  }
}
```

Question 2: Le serveur reçoit un entier positif de la part du client, et doit signaler qu'il est premier ou non.

```
import java.io.*;
import java.net.*;

public class Serveur {
  static final int port = 1200;
  public static void main(String[] args) throws Exception {
```

```
// Création d'un objet s à l'écoute du port spécifié
ServerSocket s = new ServerSocket(port);
System.out.println("En attente de connexion");
Socket socClient = s.accept(); // L'acceptation d'une connexion d'un client
System.out.println("Connexion établie");
// On crée maintenant les flux d'entrée-sortie du Serveur
// Un BufferedReader permet de lire par ligne.
BufferedReader entreeServeur = new BufferedReader(
                    new InputStreamReader(socClient.getInputStream())
                   );
// Un PrintWriter possède toutes les opérations print classiques.
PrintWriter sortieServeur = new PrintWriter(
                   new BufferedWriter(
                     new OutputStreamWriter(socClient.getOutputStream())),
                   true);
 String str = entreeServeur.readLine(); // lecture du message envoyé par le client
   int nombre=0;
   try{
    nombre=Integer.parseInt(str);
   }catch (Exception e){
      System.out.println("L'entrée du clavier n'est pas un entier");
   }
   if(estPremier(nombre)){
      System.out.println("nombre premier");
   }else{
      System.out.println("nombre non premier");
   }

// Fermeture des flux d'entrée-sortie du Serveur
```

```
        entreeServeur.close();
        sortieServeur.close();
        // Fermeture du socket client
        socClient.close();
    }

   static boolean estPremier(int n)
    {
      boolean res;  // Result of the test
      res = true;
      int i=2;
      while(i<=n/2 && res==true){
        if (n%2 == 0)
          res = false;
        i++;
      }
      return(res);
    }

}

import java.io.*;
import java.net.*;

public class Client {
   static final int port = 1200;
   public static void main(String[] args) throws Exception {

       System.out.println("Demande de connexion");
```

```
Socket socket = new Socket("127.0.0.1", port); // construit un socket client connecté à la machine et le port spécifié

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

// On crée maintenant les flux d'entrée-sortie du Serveur

 // Un BufferedReader permet de lire par ligne.

BufferedReader entreeClient = new BufferedReader( new InputStreamReader(socket.getInputStream())  );

 // Un PrintWriter possède toutes les opérations print classiques.

PrintWriter sortieClient = new PrintWriter( new BufferedWriter( new OutputStreamWriter(socket.getOutputStream())), true);

 BufferedReader keybord=new BufferedReader(new InputStreamReader(System.in));

// permet de lire à partir d'un clavier

System.out.println("Saisir au clavier un entier positif");

 String str = keybord.readLine();

 sortieClient.println(str);

 // Fermeture des flux d'entrée-sortie du Client

 entreeClient.close();

 sortieClient.close();

 // Fermeture du socket client

 socket.close();

  }

}
```

Question 3: Le serveur doit afficher tout les nombres premiers inférieurs à un entier envoyé par le client.

```
import java.io.*;

import java.net.*;

public class Serveur {

  static final int port = 1200;

  public static void main(String[] args) throws Exception {

     // Création d'un objet s à l'écoute du port spécifié
```

```
ServerSocket s = new ServerSocket(port);
System.out.println("En attente de connexion");
Socket socClient = s.accept(); // L'acceptation d'une connexion d'un client
System.out.println("Connexion établie");
// On crée maintenant les flux d'entrée-sortie du Serveur
// Un BufferedReader permet de lire par ligne.
BufferedReader entreeServeur = new BufferedReader(
                    new InputStreamReader(socClient.getInputStream())
                   );
// Un PrintWriter possède toutes les opérations print classiques.
PrintWriter sortieServeur = new PrintWriter(
                   new BufferedWriter(
                     new OutputStreamWriter(socClient.getOutputStream())),
                   true);
  String str = entreeServeur.readLine(); // lecture du message envoyé par le client
  int nombre=0;
  try{
   nombre=Integer.parseInt(str);
  }catch (Exception e){
      System.out.println("L'entrée du clavier n'est pas un entier");
  }
  for(int i=2;i<=nombre;i++){
     if(estPremier(i))
        System.out.println(i);
  }
// Fermeture des flux d'entrée-sortie du Serveur
```

```
        entreeServeur.close();
        sortieServeur.close();
        // Fermeture du socket client
        socClient.close();
    }

    static boolean estPremier(int n)
     {
       boolean res;  // Result of the test
       res = true;
       int i=2;
       while(i<=n/2 && res==true){
         if (n%2 == 0)
           res = false;
         i++;
       }
       return(res);
     }
}

import java.io.*;
import java.net.*;
public class Client {
   static final int port = 1200;
   public static void main(String[] args) throws Exception {
```

```
        System.out.println("Demande de connexion");
        Socket socket = new Socket("127.0.0.1", port); // construit un socket client connecté à la
                                                       // machine et le port spécifié
        System.out.println("Connexion établie");
        // On crée maintenant les flux d'entrée-sortie du Serveur
         // Un BufferedReader permet de lire par ligne.
        BufferedReader entreeClient = new BufferedReader( new InputStreamReader(socket.getInputStream())  );
         // Un PrintWriter possède toutes les opérations print classiques.
        PrintWriter sortieClient = new PrintWriter( new BufferedWriter( new OutputStreamWriter(socket.getOutputStream())), true);
        BufferedReader keybord=new BufferedReader(new InputStreamReader(System.in));
// permet de lire à partir d'un clavier
        System.out.println("Saisir au clavier un entier positif");
        String str = keybord.readLine();
        sortieClient.println(str);
        // Fermeture des flux d'entrée-sortie du Client
        entreeClient.close();
        sortieClient.close();
        // Fermeture du socket client
        socket.close();
    }
}
```

Question 4: Le client envoie au serveur une liste d'entiers strictement positifs : le serveur indique si ces nombres sont premiers entre eux.

```
import java.io.*;
```

```
import java.net.*;
public class Serveur {
  static final int port = 1200;
  public static void main(String[] args) throws Exception {
      // Création d'un objet s à l'écoute du port spécifié
      ServerSocket s = new ServerSocket(port);
      System.out.println("En attente de connexion");
      Socket socClient = s.accept(); // L'acceptation d'une connexion d'un client
      System.out.println("Connexion établie");
      // On crée maintenant les flux d'entrée-sortie du Serveur
      // Un BufferedReader permet de lire par ligne.
      BufferedReader entreeServeur = new BufferedReader(
                        new InputStreamReader(socClient.getInputStream())
                       );
      // Un PrintWriter possède toutes les opérations print classiques.
      PrintWriter sortieServeur = new PrintWriter(
                       new BufferedWriter(
                         new OutputStreamWriter(socClient.getOutputStream())),
                       true);
      int tab[]=new int[100]; // Tableau ou on va mettre nos entiers
      int indiceTab=0;
       while (true) {
        sortieServeur.println("Donnez moi un entier positif");
        String str = entreeServeur.readLine(); // lecture du message envoyé par le client
        if (str.equals("END")) break;
        tab[indiceTab]=Integer.parseInt(str);
```

```
            indiceTab++;
        }
        boolean premier=true;
        for(int i=0;i<indiceTab && premier==true;i++){
            for(int j=0;j<indiceTab && premier==true;j++)
                if(tab[i]%tab[j]==0 && i!=j && tab[i]!=1 && tab[j]!=1)
                    premier=false;
        }
        if(premier)
            System.out.println("ces nombres sont premiers entre eux");
        else
            System.out.println("ces nombres ne sont pas premiers entre eux");

        // Fermeture des flux d'entrée-sortie du Serveur
        entreeServeur.close();
        sortieServeur.close();
        // Fermeture du socket client
        socClient.close();
    }
}

import java.io.*;
import java.net.*;
public class Client {
    static final int port = 1200;
    public static void main(String[] args) throws Exception {
```

```
System.out.println("Demande de connexion");
Socket socket = new Socket("127.0.0.1", port); // construit un socket client connecté à la machine et le port spécifié
System.out.println("Connexion établie");
// On crée maintenant les flux d'entrée-sortie du Serveur
// Un BufferedReader permet de lire par ligne.
BufferedReader entreeClient = new BufferedReader( new InputStreamReader(socket.getInputStream()) );
// Un PrintWriter possède toutes les opérations print classiques.
PrintWriter sortieClient = new PrintWriter( new BufferedWriter( new OutputStreamWriter(socket.getOutputStream())), true);

BufferedReader keybord=new BufferedReader(new InputStreamReader(System.in)); // permet de lire à partir d'un clavier
System.out.println("Donnez le nombre total d'entiers que vous allez saisir");
int nbrToatal=Integer.parseInt(keybord.readLine());
for(int i=1;i<=nbrToatal;i++){
String str=entreeClient.readLine();
System.out.println(str);
String strKeybord = keybord.readLine();
sortieClient.println(strKeybord);
}
System.out.println("END"); // message de terminaison
sortieClient.println("END") ; // Le client envoye le message de terminaison "END" au Serveur

// Fermeture des flux d'entrée-sortie du Client
```

```
        entreeClient.close();
        sortieClient.close();
        // Fermeture du socket client
        socket.close();
    }
}
```