ATELIER N°3

GL2

LA CLASSE : **INETADDRESS**

- Adreese IP? 4 octets
- Classe **InetAddress:**

Un objet de la classe InetAddress représente une adresse Internet. Cette classe contient des méthodes pour lire une adresse, la comparer avec une autre ou la convertir en chaîne de caractères. Elle ne possède pas de constructeur : il faut utiliser certaines méthodes statiques de la classe pour obtenir une instance de cette classe.

LA CLASSE : **INETADDRESS**

Méthode	Rôle
InetAddress getByName(String)	Renvoie l'adresse internet associée au nom d'hôte fourni en paramètre
InetAddress[] getAllByName(String)	Renvoie un tableau des adresses internet associées au nom d'hôte fourni en paramètre
InetAddress getLocalHost()	Renvoie l'adresse internet de la machine locale
byte[] getAddress()	Renvoie un tableau contenant les 4 octets de l'adresse internet
String getHostAddress()	Renvoie l'adresse internet sous la forme d'une chaîne de caractères
String getHostName()	Renvoie le nom du serveur

```
import java.net.InetAddress;
import java.net.UnknownHostException;
public class TestNet1 {
  public static void main(String[] args) {
  try {
     InetAddress adrLocale = InetAddress.getLocalHost();
     System.out.println("Adresse locale = "+adrLocale.getHostAddress());
     InetAddress adrServeur = InetAddress.getByName("java.sun.com");
     System.out.println("Adresse Sun = "+adrServeur.getHostAddress());
     InetAddress[] adrServeurs = InetAddress.getAllByName("www.microsoft.com");
     System.out.println("Adresses Microsoft : ");
     for (int i = 0; i < adrServeurs.length; i++) {
     System.out.println("     "+adrServeurs[i].getHostAddress());
     }
    } catch (UnknownHostException e) {  e.printStackTrace(); }
  }
}
```

EXERCICE 1: LES SOCKETS

Rappel:

- L'élément de base de la programmation distribuée est la communication entre processus.
- Java gère les protocoles UDP et TCP: sont des protocoles de communication réseau

(mode connecté/mode non connecté) (qualité/rapidité)

EXERCICE 1 : LES SOCKETS

Requis coté Serveur:

Attente d'une connexion cliente :
Utilisation d'un port local sur lequel les connexions sont attendues

Requis coté Client:

Connexion à un serveur donné :
i) Connaissance du nom ou de l'adresse IP du serveur
ii) Connaissance du port ouvert par le serveur
(+ allocation dynamique par le système d'un port sur le client)

! La communication est ensuite bi-directionnelle

EXERCICE 1: LES SOCKETS

Librairies et Classes Utilisées:

On va utilser les librairies standard réseau, entrées sortie et utilitaires.

- import java.net.*;
- import java.io.*;
- import java.util.*;

EXERCICE 1 : LES SOCKETS

Classe Réseau: Principalement vous utiliserez les classes suivantes :

ServerSocket: fournit une classe Serveur, cette classe implémente un objet qui va écouter sur un port donné et attendre une demande de connection. La classe Serveur fournit donc une salle d'attente où faire attendre d'éventuels clients. Quand un tel objet est créé sur un port p tout client qui se connecte sur ce port est accueilli et placé en attente, les paquets qu'il envoie sont aussi acceptés et conservés par le SocketServer.

Méthodes Principales:

ServerSocket():	Constructeur par défaut
ServerSocket(int):	Créer une socket sur le port fourni en paramètre
ServerSocket(int, int):	Créer une socket sur le port et avec la taille maximale de la file fournis en paramètres

EXERCICE 1 : LES SOCKETS

Méthodes Principales:

Accept: Si un client tente de communiquer avec le serveur, la méthode accept() renvoie une socket qui encapsule la communication avec ce client.

Close: car qui termine les connections des clients de la salle d'attente. Cela devrait donc aussi de déconnecter les clients.

La classe dispose de bien d'autres méthodes, on peut en effet limiter la taille de la salle d'attente, spécifier un timeout.

EXERCICE 1 : LES SOCKETS

La mise en œuvre de la classe ServerSocket suit toujours la même logique :

- créer une instance de la classe ServerSocket en précisant le port en paramètre
- définir une boucle sans fin contenant les actions ci-dessous:

i. appelle de la méthode accept() qui renvoie une socket lors d'une nouvelle connexion

ii. obtenir un flux en entrée et en sortie à partir de la socket

iii. écrire les traitements à réaliser

```
import java.net.*;
import java.io.*;
public class TestServeurTCP {
  final static int port = 9632;
  public static void main(String[] args) {
    try {
      ServerSocket socketServeur = new ServerSocket(port);
      System.out.println("Lancement du serveur");
      while (true) {
        Socket socketClient = socketServeur.accept();
        String message = "";
        System.out.println("Connexion avec : "+socketClient.getInetAddress());
        BufferedReader in = new BufferedReader(  new InputStreamReader(socketClient.getInputStream()));
        PrintStream out = new PrintStream(socketClient.getOutputStream());
        message = in.readLine();
        out.println(message);
        socketClient.close();
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

```
import java.net.*;
import java.io.*;
public class TestServeurTCP {
  final static int port = 9632;
  public static void main(String[] args) {
    try {
      ServerSocket socketServeur = new ServerSocket(port);
      System.out.println("Lancement du serveur");
      while (true) {
        Socket socketClient = socketServeur.accept();
        String message = "";
        System.out.println("Connexion avec : "+socketClient.getInetAddress());
        BufferedReader in = new BufferedReader(  new InputStreamReader(socketClient.getInputStream()));
        PrintStream out = new PrintStream(socketClient.getOutputStream());
        message = in.readLine();
        out.println(message);
        socketClient.close();
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

EXERCICE 1 : LES SOCKETS

Classe Réseau:

Socket Cette classe encapsule la connexion à une machine distante par le réseau. Elle gère la connexion, l'envoi de données, la réception de données et la déconnexion.

La classe Socket possède plusieurs constructeurs dont les principaux sont :

socket()	Constructeur par défaut
Socket(String, int)	Créer une socket sur la machine dont le nom et le port sont fournis en paramètres
Socket(InetAddress, int)	Créer une socket sur la machine dont l'adresse IP et le port sont fournis en paramètres

EXERCICE 1 : LES SOCKETS

Méthodes:

InetAddress getInetAddress()	Renvoie l'adresse I.P. à laquelle la socket est connectée
void close()	Fermer la socket
InputStream getInputStream()	Renvoie un flux en entrée pour recevoir les données de la socket
OutputStream getOutputStream()	Renvoie un flux en sortie pour émettre les données de la socket
int getPort()	Renvoie le port utilisé par la socket

EXERCICE 1 : LES SOCKETS

Le mise en oeuvre de la classe Socket suit toujours la même logique :

- créer une instance de la classe Socket en précisant la machine et le port en paramètres
- obtenir un flux en entrée et en sortie
- écrire les traitements à réaliser

EXERCICE 1 : LES SOCKETS

La mise en œuvre de la classe Socket suit toujours la même logique :

- créer une instance de la classe Socket en précisant la machine et le port en paramètres
- obtenir un flux en entrée et en sortie
- écrire les traitements à réaliser

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

public class TestClientTCP {
  final static int port = 9632;
  public static void main(String[] args) {
    Socket socket;
    DataInputStream userInput;
    PrintStream theOutputStream;
    try {
      InetAddress serveur = InetAddress.getByName(args[0]);
      socket = new Socket(serveur, port);
      BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
      PrintStream out = new PrintStream(socket.getOutputStream());
      out.println(args[1]);
      System.out.println(in.readLine());
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```