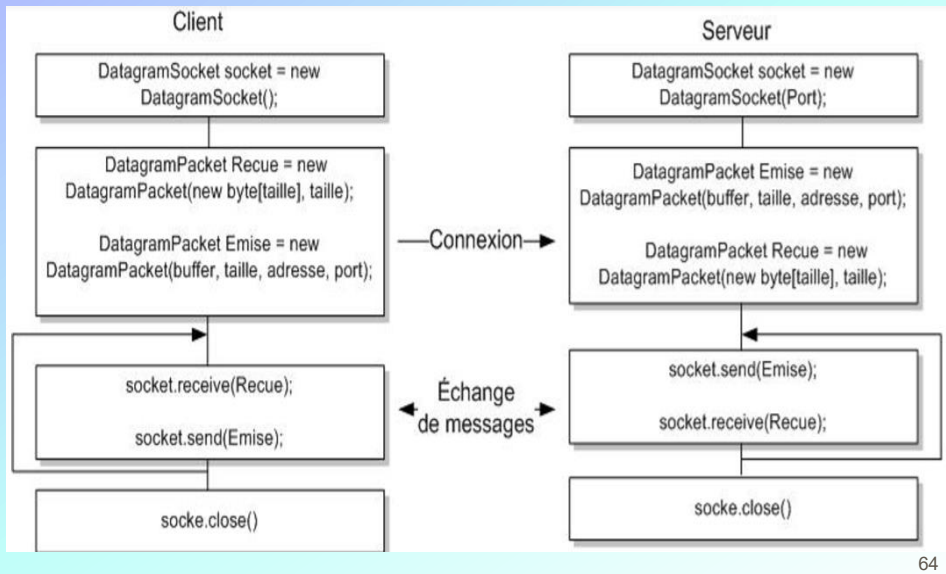
Les sockets UDP



Les sockets UDP : La classe DatagramSocket

Constructeur	Rôle
`DatagramSocket()`	Créé une socket attachée à toutes les adresses IP de la machine et un à des ports libres sur la machine.
`DatagramSocket(int)`	Créé une socket attachée à toutes les adresses IP de la machine et au port précisé en paramètre.
`DatagramSocket(int, InetAddress)`	Créé une socket attachée à adresse IP et au port précisé en paramètre.

Méthode	Rôle
`close()`	Fermer la Socket et ainsi libérer le port.
`receive(DatagramPacket)`	Recevoir des données.
`send(DatagramPacket)`	Envoyer des données.
`int getPort()`	Renvoie le port associé à la socket.

Les sockets UDP : La classe DatagramPacket

Constructeur	Rôle
`DatagramPacket(byte tampon[], int length)`	Encapsule des paquets en réception dans un tampon
`DatagramPacket(byte port[], int length, InetAddress adresse, int port)`	Encapsule des paquets en émission à destination d'une machine

Méthode	Rôle
`InetAddress getAddress ()`	Renvoie l'adresse du serveur
`byte[] getData()`	Renvoie les données contenues dans le paquet
`int getPort ()`	Renvoie le port
`int getLength ()`	Renvoie la taille des données contenues dans le paquet
`setData(byte[])`	Mettre à jour les données contenues dans le paquet

Utilisation des sockets UDP : *Envoie d'un Datagram*

1. créer un **DatagramPacket** en spécifiant :
   - les données à envoyer
   - leur longuer
   - la machine réceptrice et le port
2. utiliser la méthode **send(DatagramPacket)** de **DatagramSocket**
   - pas d'arguments pour le constructeur de la classe DatagramSocket (**DatagramSocket socket = new DatgramSocket();**)

   car toutes les informations se trouvent dans le paquet envoyé

Utilisation des sockets UDP :
Envoie d'un Datagram

```
//Machine destinataire
InetAddress address = InetAddress.getByName(" ensat.ac.ma ");
static final int PORT = 4562;
//Création du message à envoyer
String s = new String (" Message à envoyer");
int longueur = s.length();
byte[] buffer = s.getBytes();
//Initialisation du paquet avec toutes les informations
DatagramPacket paquet;
  paquet = new DatagramPacket(buffer,longueur, address,PORT);
 //Création du socket et envoi du paquet
DatagramSocket socket = new DatgramSocket();
socket.send(paquet);....
```


Utilisation des sockets UDP :
Réception d'un Datagram

1. créer un **DatagramSocket** qui écoute sur le port de la machine du destinataire
2. créer un **DatagramPacket** pour recevoir les paquets envoyés par le serveur
   - Dimensionner le buffer assez grand pour pouvoir recevoir le paquet
3. utiliser la méthode **receive(DatagramPacket)** de **DatagramSocket**
   - Cette méthode est bloquante

Utilisation des sockets UDP : *Réception d'un Datagram*

```
//Définir un buffer de réception
byte[] buffer = new byte[1024];
//On associe un paquet à un buffer vide pour la réception
DatagramPacket paquet;
  paquet = new DatagramPacket(buffer , buffer.length());
//On crée un socket pour écouter sur le port
DatagramSocket socket = new DatgramSocket(PORT);
while (true) {
//attente de réception
socket.receive(paquet );
//affichage du paquet reçu
String s = new String(buffer , 0 , paquet.getLength());
System.out.println("Paquet reçu : + s);
 }
```


TP3

Réaliser un échange simple entre un Client et un serveur en mode non connecté en utilisant le protocole UDP.

1. **Le client se connectera au serveur tout en lui envoyant un message contenant son login**
2. **Le serveur recevra le login et lui enverra un message de Bienvenu**

Transformer le serveur pour qu'il servira plusieurs clients simultanément

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

public class Exemple_ServeurUDP {
  final static int port = 9632;
  final static int taille = 1024;
  static byte buffer[] = new byte[taille];

  public static void main(String argv[]) throws Exception {
      //Création d'un socket du coté erveur qui écoute sur le port
      //Création d'un socket du coté serveur sur le port 9632

      System.out.println("Lancement du serveur");
      while (true) { //Boucle infinie pour pouvoir traiter plusieurs clients un par un
        //Préparation des paquets pour
        //Préparation du paquet pour la réception
        //Préparation du paquet pour l'envoie
        //Réception du message du client qui va se connecter
        //Réception du message du client
      //Affichage de l'adresse du client et les donnée du client
      System.out.println("\n"+paquet.getAddress());
          int taille = paquet.getLength();
          String donnees = new String(paquet.getData(),0, taille);
          System.out.println("Donnees reçues = "+donnees);
      //Envoi d'un message de bienvenu au client en préparant un paquet d'envoie
      String message = "Bonjour "+donnees;
          System.out.println("Donnees envoyees = "+message);
          //Préparation du paquet pour l'envoie
          //Envoie du paquet
          }
} }
```

```
import java.io.*;
import java.net.*;
import java.util.Scanner;
public class Exemple_ClientUDP {
  final static int port = 9632;
  final static int taille = 1024;
  static byte buffer[] = new byte[taille];
  public static void main(String argv[]) throws Exception {
          try {
          InetAddress serveur = InetAddress.getLocalHost();
          System.err.println(serveur.getHostAddress().toString());
          int length = 1024;
          byte buffer[] = new byte[length];
            // préparation du tableau d'octets à envoyer (tampon)
            //Lecture du message du client depuis l'entrée standard (clavier)
            //Préparation du tableau d'octets (tampon ou buffer)
            //...
            //...
            //
          // Création du socket Client et préparation des paquets pour l'envoie et pour la réception
          //Création du socket Client
          //Préparation du paquet pour l'envoie
          //Préparation du paquet pour la réception
                                                                                    ort);
         //Envoie du message et attente de la réception du message du serveur
          //Envoie du message du Client (Login)
          //Attente de la réception du message du serveur
        //Affichage du message recu du serveur ainsi que l'addrese du serveur et le port
          //Affichage du message reçu de la part du serveur
          //Affichage de l'adresse du serveur ainsi que le numéro du port
          }
          catch (Exception e) { e.printStackTrace(); }
              } }
```

```
import java.net.*;
public class Exemple_ServeurUDPThread extends Thread {
  protected DatagramSocket socket;
  public static void main(String[] args) {
          try {  //Création et démarrage d'un thread sur le constructeur de la classe
                 //…
              } catch (Exception e) { e.printStackTrace(); }
  }
  public Exemple_ServeurUDPThread() throws Exception{
          //Traitement du constructeur
                                    }
  public void run() {
    try {   while (true) {
                 //Redéfinition de la méthode run()
                 //Tout le traitement qui doit se faire du coté client avec chaque client qui va se connecter
                 //…
                 //…
                 //…
                 //…
                 //…
                 //…
                 //…
                 //…
                 //…
                 //…
                 //…
                 //…
                 //…
                 //…
                    }
          }catch (Exception e) { e.printStackTrace();}
                    } }
```