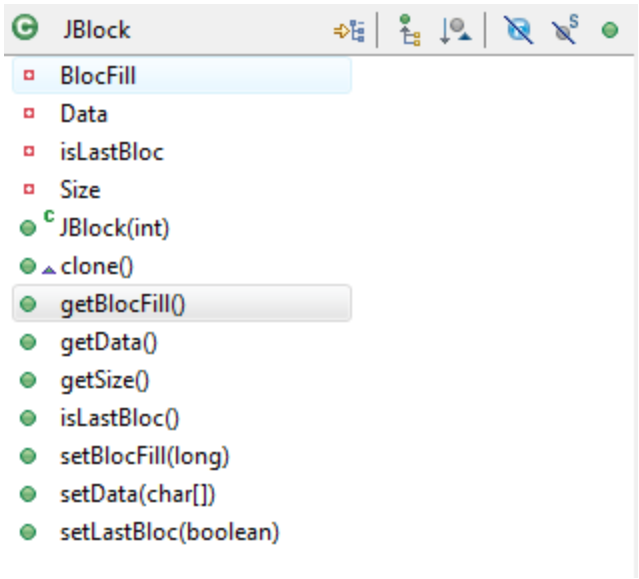
PROJET P2P

Version 1

La version 1 gere simplement le telechargement entre le client et le serveur .

On crée des objets JBlocks afin de gérer la découpe du fichier en morceaux .
Chaque JBlock contient un tableau de données mais aussi nombre d'informations :

- s'agit-il du dernier Bloc ?
- le bloc est-il rempli à 100% ?
- etc ...



à noter qu'on surcharge/Override clone() par :

```
return super.clone();

afin de pouvoir copier des JBlocks ( la méthode étant protégée par
défaut dans Object ... )
```

Ensuite pour transférer les JBlocks du serveur vers le client , on écrit le JBlock (down)casté en (Object) sur une **ObjectOutputStream** représentant la connection réseau .

```
outgoing = new ObjectOutputStream( new
java.io.BufferedOutputStream( client.getOutputStream()) );

((ObjectOutputStream) outgoing).writeObject((Object)jb);
```

Pour récupérer l'objet de l'autre côté , on effectue un ReadObject sur l' **ObjectInputStream** puis un upcast en JBlock

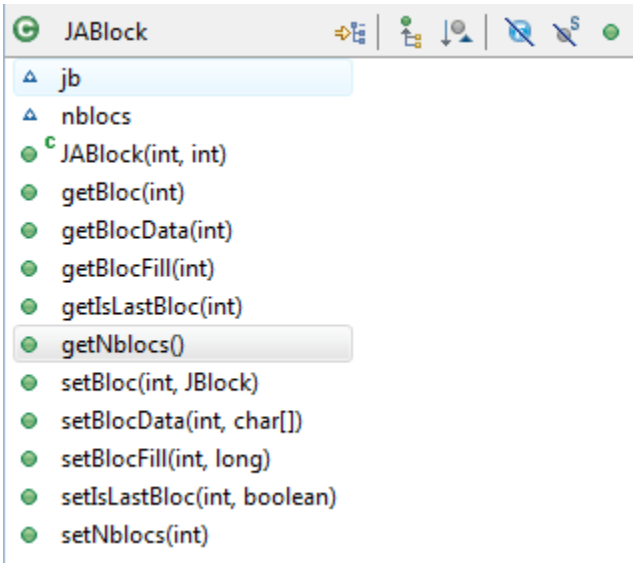
```
ObjectInputStream  incoming2 = new java.io.ObjectInputStream ( new
java.io.BufferedInputStream (connection.getInputStream())  );
```

```
this.MyBlock=(JBlock) incoming2.readObject();
```

Il est à noter que ce procédé provient de la *sérialization* de notre objet JBlock .

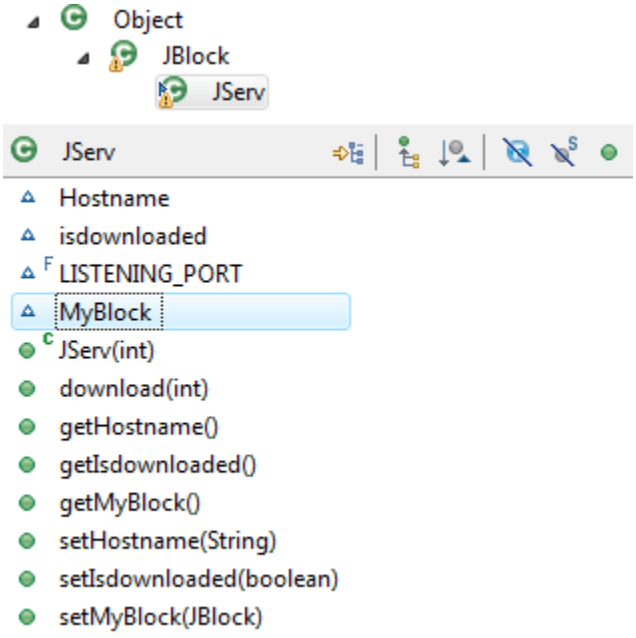
```
public class JBlock implements Serializable
```

Enfin on crée une classe JABlock qui est essentiellement un tableau de JBlocks avec des accesseurs .



Version 2

On crée des objets JServ étendant le modéle JBlock pour gérer le téléchargement bloc par bloc .



Chaque Jserv possede une methode **Download(int position)** qui va recuperer le bloc depuis le serveur en envoyant une trame de requete puis en lisant l'objet JBlock envoyé depuis le serveur .

l'idée est de downcaster un Jserv en Jblock puis remplir un JABlock avec ces Jblocks , ensuite on peut upcaster ces JBlocks en Jserv et utiliser leur methode download(int pos)

Il faut ensuite que le serveur puisse parser les messages textes envoyés par le client donc définir un protocole basique : .

Exemple :

Client ---------->

SLICEFILE FILENAME
<---------- SERVEUR
REQUETE : "PREPARER FICHIER FILENAME"

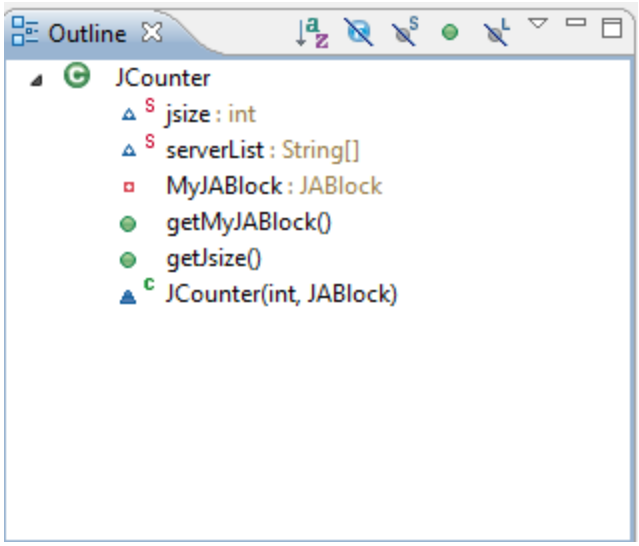
THIS_IS_FILESIZE FILESIZE
THIS_IS_BLOC_SIZE L BLOC SIZE
THIS_IS_NUMBER_OF_BLOCS NUMBER OF BLOCS

Client ---------->

Tag Length Value
REQUESTBLOC POSITION
<---------- SERVEUR
REQUETE : "ENVOYER BLOC POSITION"

JBLOCK

Nous devons ensuite avoir une sorte de "compteur" indépendant qui sait ce qui se télécharge , quels fichiers ont été slicés et sont prêts , etc ...



Dans le constructeur de Jserv , nous écrivons le code suivant qui permettra le téléchargement aléatoire depuis des serveurs p2p

```
Random rand = new Random();

int is=0;
            int il=serverList.length;
            for (int j=0;j<jab.getNblocs();j++)
            {
                  JServ  jserv = new JServ(jab.getBloc(0).getSize());
                  is=max(rand.nextInt() %il,0);
                  jserv.setHostname(serverList[is]);
                  jab.setBloc(j, (JBlock) jserv);
            }
```

VERSION 3

Nous devons "merger" le client et le serveur , Pour ce

faire , nous transformons le code serveur en thread indépendante lancée sur son propre port en écoute puis lancer un shell qui écoute sur un autre port .

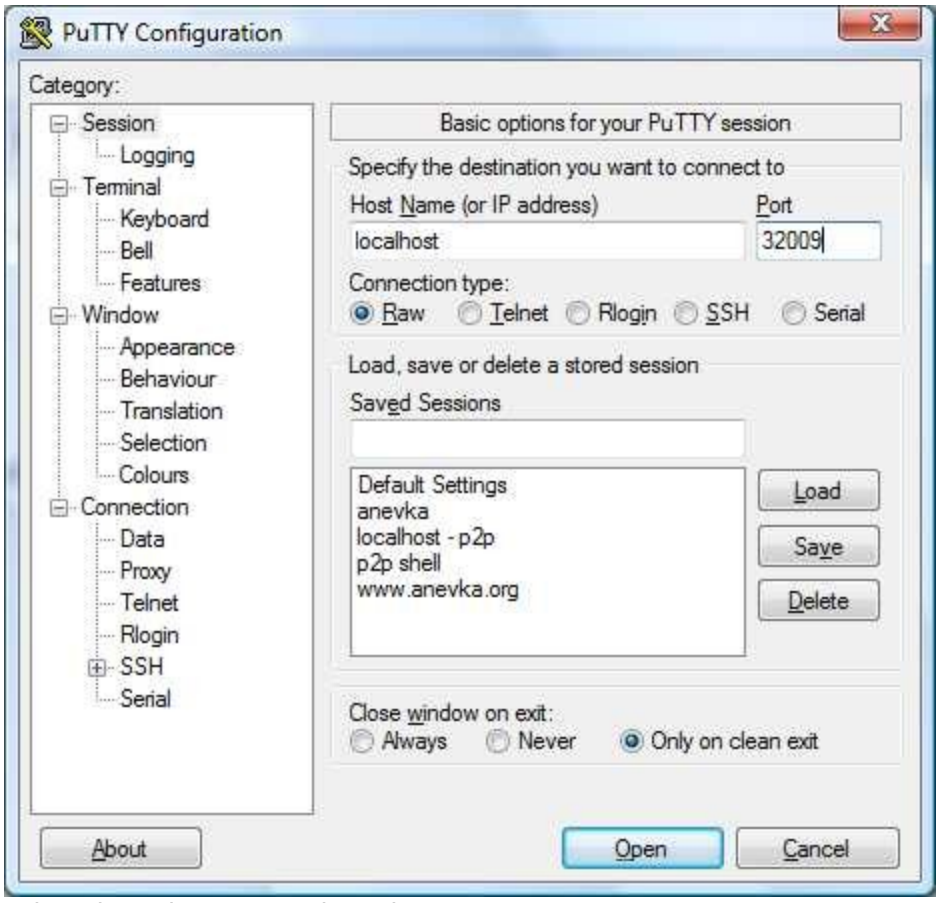
```
p2pserver p2ps = new p2pserver();

                                    Thread Thp2pServer = new
                                    Thread(null,p2ps,"p2pserver");
                                    Thp2pServer.setDaemon(true);
                                    Thp2pServer.start();

                                    System.out.println("serveur started as
daemon \n");
                                    //doserver();

                                    Shell();
```

Nous communiquons avec le p2p via le shell : on implémenta les commandes suivantes :

STARTDOWNLOAD
STOP
CLOSE
HELP

L'Application lance un thread p2p server sur le port 32008 puis lance un shell de commandes sur le port 32009

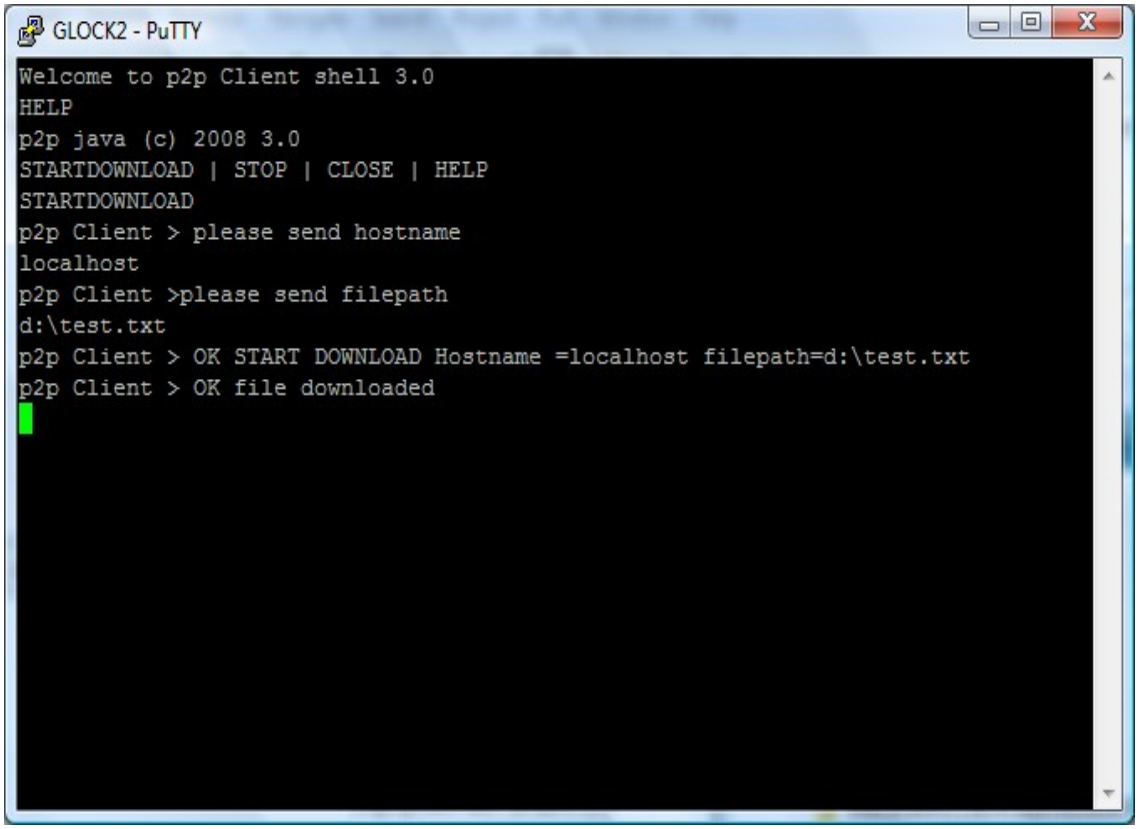
pour communiquer avec le shell , on lance un putty en mode "raw" sur le port 32009 (ou netcat ...)



puis on lance les commandes suivantes :

STARTDOWNLOAD
HOSTNAME
FILEPATH

Exemple :



cela fera appel au client qui se connectera au serveur p2p lancé en daemon .