TP n°6

Sockets en mode non connecté

Le but de ce TP est de manipuler les sockets en mode non connecté en *Java*. Nous allons réaliser différentes applications distribuées qui pourront être exécutées localement ou testées sur plusieurs postes.

Avant de commencer le TP, n'oubliez pas de tester les différents programmes d'exemple des fiches concernant les sockets en mode non connecté.

1 Fichiers de configuration en JSON

Le résultat de cet exercice devra être utilisé dans tous vos futurs programmes *Java* (projets compris). Il s'agit d'un exercice préliminaire et doit être traité **rapidement**.

Dans les prochaines applications à développer, il est nécessaire de spécifier les adresses IP et les numéros de port des clients et/ou du serveur (selon l'application). Pour éviter de spécifier ces données directement dans le code (en constantes) ou en arguments, nous souhaitons développer une classe `Config` pour gérer les fichiers de configuration au format JSON. Ainsi, pour configurer un client qui se connecte à un serveur, nous pourrions utiliser le fichier suivant :

```
{
    "adresse" : "127.0.0.1",
    "port" : 1056
}
```

Dans chaque programme (classe *Java* contenant un `main`), nous pouvons ajouter les éléments suivants :

- Une fonction `creerFichierConfiguation` qui crée un fichier de configuration avec des valeurs par défaut ; le nom du fichier est passé en paramètre. Elle retourne l'objet `Config` créé.
- Dans le main :
  - Aucun argument n'est spécifié : si le fichier par défaut existe, il est ouvert, sinon il est créé avec la méthode `creerFichierConfiguation`.
  - Un argument est spécifié : si le fichier spécifié existe, il est ouvert, sinon il est créé avec la méthode `creerFichierConfiguation`.
- Dans tous les cas, un objet `Config` est créé et il peut être consulté pour chaque valeur nécessaire.

Questions

1. Complétez la classe `Config` (à télécharger sur *Moodle*).
2. Testez votre classe `Config` à l'aide de la classe `Exemple` (à télécharger sur *Moodle*).

2 Prise en main d'UDP

Nous souhaitons écrire une application distribuée constituée de deux applications. La première, nommée *serveur*, écoute sur un numéro de port donné (spécifié dans le fichier de configuration) et se met en attente de réception de messages (une chaîne de caractères, dans un premier temps). Dès qu'un message est reçu, il est affiché à l'écran. La deuxième application, nommée *client*, envoie un message au *serveur*, le message étant lu au clavier. L'adresse IP et le numéro de port du *serveur* sont spécifiés dans son fichier de configuration.

Questions

1. Écrivez les applications et vérifiez leur fonctionnement.
2. Cette application distribuée fonctionne-t-elle sur le modèle de communication client/serveur ?
3. Que se passe-t-il si le *serveur* n'est pas démarré et que le *client* envoie un message ?

3 Jeu de tennis en réseau

À partir des applications précédentes, nous souhaitons développer un jeu de tennis en réseau. Les données échangées sont maintenant au format JSON. Elles contiennent le score des deux joueurs, le coup du joueur qui envoie le message, des infos (service, renvoi, *etc.*) et un commentaire éventuel (au format texte) :

```
{
    "score" : [0, 0],
    "coup" : 1,
    "infos" : "service",
    "commentaire" : "prends␣ça,␣bouffon␣!"
}
```

Le joueur qui démarre le serveur est le joueur 1 et commence en premier. Il choisit un coup qui correspond à un chiffre entre 1 et 3 (correspondant à droite, milieu ou gauche) et peut écrire un commentaire. Un message est envoyé au client correspondant au joueur n°2. Celui-ci choisit lui aussi un nombre entre 1 et 3 pour tenter de relancer la balle :

- Si le nombre est égal au coup, il relance la balle.
- Si la différence est de 1, il a 50% de chance de relancer la balle.
- Si la différence est de 2, il manque la balle.

Si la balle est relancée, le joueur choisit un coup (et peut écrire un commentaire). Sinon, le joueur adversaire marque un point, par contre le joueur perdant rejoue immédiatement (ce n'est pas la règle, mais on s'en moque). La partie s'arrête lorsque l'un des joueurs atteint le score de 5.

Exemple de déroulement d'une partie :

- Le joueur 1 choisit le coup n°1
- Le joueur 2 choisit 1 et relance la balle. Il choisit le coup n°3.

- Le joueur 1 choisit 2 mais manque la balle (50% de chance de rattraper). Le joueur 2 marque un point. Le joueur 1 choisit le coup n°1.
- Le joueur 2 choisit 3 et manque la balle. Le joueur 1 marque un point. Le joueur 2 choisit le coup n°2.
- *etc.*

Questions

1. Quelle est la différence entre un envoi d'une chaîne de caractères correspondant à un document JSON et un envoi d'un objet `JSONObject` sérialisé ? À votre avis, quel est le meilleur choix ?
2. Écrivez les applications.
3. Si possible, testez avec un autre étudiant.