MATH EN INFO

TRAVAUX PRATIQUES N° 1

Premiers pas

1 Objectifs

Le principal objectif des séances de TP de ce module est l'implémentation des notions de bases et de quelques algorithmes présentés durant les CM dans le cadre concret du développement d'une IA pour un petit jeu 2D inspiré du jeu "Stardew Valley". Dans ce jeu, le joueur contrôle Abigail qui vient tout juste d'acquérir sa propre ferme. L'objectif d'Abigail est de s'enrichir le plus rapidement possible. Pour cela, elle devra acheter des graines, les planter, les arroser pour enfin pouvoir vendre les légumes récoltés. Une fois bien installée, Abigail pourra investir dans quelques poules et ainsi pouvoir récolter des œufs qu'elle pourra revendre ou transformer en mayonnaise.

L'objectif de cette première séance est de prendre en main le moteur du jeu et de comprendre quelques rudiments de communications client-serveur qui nous seront utiles durant ce projet.

2 Présentation du projet

2.1 Ce que vous trouverez sur le commun

Sur le commun, aux côtés de ce fichier, vous trouverez un répertoire "Projet" contenant les différents fichiers nécessaires à ce projet :

- **Serveur.jar :** il s'agit du serveur du jeu. Il faudra le lancer avant de lancer votre IA lors de vos tests.
- **IAScript.jar :** il s'agit d'une IA de test scriptée (c'est-à-dire effectuant toujours la même série d'actions).
- **TestServeur.jar :** il s'agit d'un serveur de test que nous utiliserons dans la partie suivante.
- **ProtocoleRéseau.pdf :** il s'agit d'un descriptif complet du protocole réseau utilisé par le serveur. En d'autres termes, il s'agit de la liste de tout ce que votre IA pourra dire/demander au serveur et des réponses possibles du serveur.
- **Manuel.pdf :** il s'agit d'un document regroupant un court descriptif des règles du jeu ainsi que des explications sur le fichier de configuration.

2.2 Prise en main du serveur de jeu

Commençons par prendre en main le fonctionnement du serveur de jeu. Pour cela, nous allons déjà le lancer. Bien qu'il soit possible de le démarrer rien qu'en double cliquant sur le fichier .jar, cela ne nous permettrait pas d'avoir accès à la console dans laquelle s'affichera toute la discussion entre l'IA et le serveur.

Nous allons donc démarrer le serveur via l'invite de commande de Windows. Une fois celle-ci ouverte, rendez-vous dans le répertoire où se trouve le fichier "Serveur.jar" et exécutez la commande suivante :

java -jar Serveur.jar

Un premier écran apparait alors.

Si le serveur ne se lance pas et qu'un message mentionnant javafx apparait dans la console, c'est que vous ne disposez pas de la bonne version de la jdk sur votre ordinateur.

Après une courte animation, deux boutons apparaissent entourant une zone de texte. Le premier bouton sert à lancer une partie et le deuxième à quitter le jeu. La zone de texte sert à spécifier la seed de la carte générée, dans un premier temps, ne la changez pas.

Si vous cliquez sur le premier bouton, l'écran se figera comme si le serveur était bloqué. Le serveur attend en fait la connexion de l'IA. La partie débutera réellement quand le serveur aura détecté votre IA. Notez que le serveur ne dispose pas d'un système de TimeOut (système arrêtant l'attente du serveur si l'IA met trop de temps à se connecter), si vous ne disposez d'aucune IA (ou que celle-ci ne parvient pas à se connecter) vous pouvez fermer le serveur depuis l'invite de console (ctrl+C).

Nous allons maintenant tester l'IA scriptée fournie avec le sujet. Pour cela, commencez par cliquer sur Start pour mettre le serveur en position d'attente, puis dans un deuxième invite de commande, lancez IAScript.jar (avec la même commande que pour lancer le serveur). La partie doit alors commencer et Abigail doit se déplacer seule dans le jeu. Notez que l'intégralité des communications entre l'IA et le serveur apparait progressivement dans les deux invites de commande.

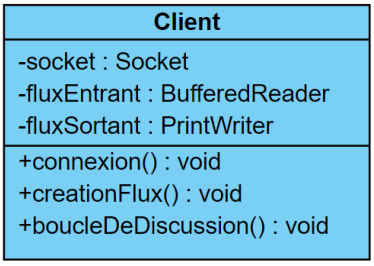
1. **A FAIRE : Étudiez le script des communications entre l'IA et le serveur et déterminez le sens à chacune des lignes de ce script.**

3 Communication client-serveur

3.1 Un client de base

Nous allons maintenant voir comment gérer une communication client-serveur (côté client) en Java. Pour cela, nous allons dans un premier temps créer un client pour le serveur TestServeur.jar.

2. **A FAIRE : Dans un projet Netbeans (version 8.x), créez une classe suivante :**



Pour permettre la discussion entre un client et un serveur, nous avons besoin de trois choses (objets) :

- **Socket :** il s'agit grossièrement du canal de discussion entre le client et le serveur (le "tube" dans lequel le client et le serveur vont pouvoir parler).
- **fluxEntrant :** est l'objet en charge d'écouter ce que le serveur dit (écouter).
- **fluxSortant :** est l'objet en charge de transmettre les messages au serveur (parler).

La méthode connexion s'occupe de créer le socket. Son constructeur demande deux paramètres :

- l'IP du serveur. Ici, nous travaillerons en local (client-serveur sur le même ordinateur) l'IP sera donc 127.0.0.1.
- Le port du serveur. De même, comme nous travaillons en local, le serveur pourrait utiliser à peu près n'importe quel port libre. C'est ici le port 1234 qui a été retenu.

3. **A FAIRE : Codez la méthode connexion dont l'entête sera : "public void connexion() throws IOException".**

La méthode creationFlux s'occupe de créer les deux objets fluxEntrant et fluxSortant. Pour créer fluxEntrant on va procéder en 3 étapes :

- On récupère l'InputStream du socket (à l'aide du getter).
- On construit un objet du type InputStreamReader à partir de cet InputStream.
- On construit notre fluxEntrant à partir de l'InputStreamReader.

Pour créer fluxSortant on va procéder en 2 étapes :

- On récupère l'OutputStream du socket (à l'aide du getter).
- On construit notre fluxSortant à partir de l'OutputStream en donnant au constructeur un deuxième paramètre : true (pour que le fluxSortant envoie automatiquement les messages au serveur quand on lui les donnera).

4. **A FAIRE : Codez la méthode creationFlux dont l'entête sera : "public void creationFlux() throws IOException".**

La méthode boucleDeDiscussion va gérer l'intégralité de la discussion. En voici le code :

```
public void boucleDeDiscussion() throws IOException {
        String messageRecu = "";
        String messageAEnvoyer = "";
        System.out.println("-- Debut de la transmission --");
        do{
                //Reception du message du serveur
                messageRecu = this.fluxEntrant.readLine();
                System.out.println("< "+messageRecu);
                //Envoi du message de réponse
                messageAEnvoyer = "Quoi ?";
                fluxSortant.println(messageAEnvoyer);
                System.out.println(">"+messageAEnvoyer);
        }while(!messageRecu.equals("FIN"));
        System.out.println("-- Fin de la transmission --");
}
```

5. **A FAIRE : Après avoir compris ce que faisait ce code, codez la méthode boucleDeDiscussion.**

Enfin, le main de votre programme va successivement créer un nouvel objet "Client" puis en exécuter les méthodes connexion,creationFlux et boucleDeDiscussion.

6. **A FAIRE : Codez le main de votre programme et testez le à l'aide du serveur de test.**

3.2 Une communication un peu plus évoluée

Comme vous avez dû le remarquer dans le test, bien que votre client communique avec le serveur, ses réponses ne satisfont pas le serveur de test. En effet, dans l'état actuel, votre client répond systématiquement "Quoi ?" au serveur. Nous souhaitons changer cela. Voici ce qu'attend le serveur :

- Le serveur peut envoyer (aléatoirement) trois messages différents : "Bonjour", "Ca va ?" ou "FIN".
- Si le serveur dit "Bonjour", il attend "Bonjour" comme réponse.
- Si le serveur dit "Ca va ?", il attend "Oui" comme réponse.
- Si le Serveur dit "FIN", il attend "FIN" comme réponse et la communication s'arrêtera là.

7. **A FAIRE : Modifiez votre code pour que votre client respecte ce protocole.**

3.3 Une première IA

Nous avons maintenant tout ce qu'il nous faut pour réaliser une première IA pour notre jeu.

8. **A FAIRE : En vous inspirant du client de test que vous venez de réaliser, développez une petite IA respectant le script suivant :**
   - **l'IA commence par se déplacer de deux cases vers le bas.**
   - **l'IA doit se déplacer autant que possible vers la droite.**
9. **A FAIRE : En observant les réponses du serveur, modifiez votre IA pour que celle-ci envoie le message "END" dès que le personnage est bloqué.**
10. **A FAIRE : Modifiez votre IA pour que votre personnage se déplace aléatoirement dans la carte.**

Une telle IA est couramment appelée une "Drunken blind IA" :

- **Drunken** car elle agit aléatoirement.
- **Blind** car elle n'a récupéré aucune information du jeu (carte...).

Lorsque l'on cherche à développer une IA, il est fortement recommandé de commencer par une IA de ce type afin de tester non seulement que l'on arrive à établir la connexion avec le serveur mais aussi que l'on comprend correctement les bases du protocole de discussion.