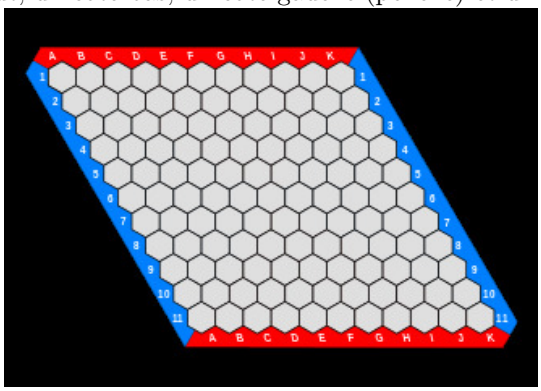
Projet de Programmation
Licence 1 UPEC 2021/2022
Hex

Le jeu de Hex

Le but de ce projet est d'écrire un programme pour jouer le jeu de HEX. Hex est un jeu inventé par le mathématicien danois Piet Hein, et indépendamment par le mathématicien américain John Nash (prix Nobel pour l'économie). Il se joue sur tablier formé par des cases *hexagonales*. La forme des cases fait que le tablier est en forme de losange. On peut imaginer que cette losange soit penchée, de façon qu'il y a un coté haut, un coté bas, un coté gauche (penché) et un coté droit (penché).



Les deux joueurs doivent placer des jetons dans les cases. Chaque joueur a une couleur, disons rouge et bleu. A son tour un joueur doit simplement choisir une case libre. Le joueur bleu gagne si les jetons bleus forment un parcours continu entre le coté bas et le coté haut. Le joueur rouge gagne si les jetons rouges forment un parcours continu entre le coté gauche et le coté droit.

La nature du tablier fait aussi que le match nul est impossible : quand toutes les cases sont remplies, soit c'est le joueur rouge qui a gagné, soit c'est le joueur bleu (vous pouvez chercher la preuve mathématique de ce fait...)

La règle du *swap* Après avoir joué un bon nombre de parties, on peut remarquer que le premier joueur à un avantage à placer son jeton au milieu du tablier. Pour éviter cet avantage, une règle spéciale a été ajouté au jeu : le *swap* (ou échange). Le deuxième jouer peut, s'il le souhaite choisir de prendre le rôle du premier joueur : si le premier joueur est le rouge, alors le deuxième joueur peut devenir lui même le joueur rouge et prendre le coup du premier joueur comme le sien. Ce coup peut être joué seulement au premier coup du deuxième joueur. Le but est d'empêcher un avantage trop important pour le premier joueur, qui devra choisir une case pas trop centrale.

Ce qu'il faut réaliser

Minimum syndical – pour une note de 10

Il vous est demandé de fournir au moins un programme qui puisse faire jouer deux joueurs humains l'un contre l'autre. Votre programme devrait "arbitrer" le jeu, décider si un coup proposé est valide et signaler la perte de la partie pour celui qui a proposé le coup invalide, afficher la configuration courante, gérer le tour de rôle et décider lorsque un joueur a gagné par la règle principale (en créant un parcours continu).

Attention ! Bien lire la section sur la qualité de votre code ci-dessous ! Soumettre du code de bonne qualité est aussi essentiel pour avoir cette note du minimum syndical !

Pour quelques points de plus

Java Un bonus de 2 points sera donnés pour les groupes qui réaliseront leur projet en Java plutôt qu'en Robusta, toujours en faisant attention à la qualité du code.

IA et strategies Vous pouvez ne pas vous contenter du minimum – alors il vous est demandé d'implémenter des "Intelligences Artificielles" qui soient capables de jouer (tant bien que mal) contre un humain. Pour cela il faut imaginer et implémenter des *stratégies* : des "plans" d'action de l'IA pour jouer en fonction

de la situation sur la table. Il faut au moins que l'ordinateur puisse choisir une case libre – c'est la stratégie "aléatoire" bête. Ensuite l'ordinateur pourrait chercher à occuper les cases centrales, à poursuivre un but assez simple comme avancer dans une direction tant qu'il y a des cases libres, ou à bloquer l'avancement de l'adversaire vers son but. A vous de chercher... et de trouver.

Un tel programme devrait donc permettre de faire jouer un humain contre le programme lui-même, ou de faire jouer deux programmes entre eux – et c'est le but du point suivant !

Une intelligence artificielle donnera lieu à un bonus de 1 à 4 points.

Tournoi Pour les groupes qui feraient le projet en Java, et qui développent des IA plus intelligentes que l'IA aléatoire, nous allons organiser un tournoi : l'idée est de lancer en exécution deux programmes proposés par des binômes différents sur la même machine (pour cela il faudra respecter un protocole très précis en Java), chacun considérant l'autre programme comme son adversaire.

Nous fournirons des classes Java qui implementent des tampons permettant la communication entre deux programmes. Votre programme devra être capable d'écrire et de lire les coups dans ces tampons. On vous donnera le format pour lire et écrire les coups.

Chaque programme calculera son prochain coup et l'écrira dans le tampons, puis il attendra que l'autre programme écrive son coup dans dans le tampons pour répondre. On réalisera un tournoi entre les binômes (et peut-être on inscrira dans ce tournoi nos programmes à nous, les enseignants...), et que le meilleur gagne.

La participation au tournoi donnera lieu à un bonus de 2 points.

Affichage Vous pouvez afficher la configuration courante (position de chaque jeton) dans la console, comme un tableau de caractères. Un affichage avec des pixels serait un plus, et donner lieu à un bonus de 2 points. On vous fournira une librairie Java qui pourra vous aider dans la démarche (ou bien vous utiliserez les fonctionnalités de Robusta)

1 Critères d'évaluation de la qualité de votre code

- Votre programme principal (c'est à dire la fonction `main`) doit comporter moins de 30 lignes et ne doit comporter que des appels de fonctions et des déclarations.
- Chaque fonction doit contenir moins de 10 lignes. Si une de vos fonctions contient plus de 10 lignes, vous devez alors écrire au moins une sous-fonction afin de découper votre programme.
- Vous devez séparer le traitement des données des interactions avec l'utilisateur, c'est à dire que chaque sous-programme (fonction ou procédure) effectuant une action sur le jeu doit prendre en paramètre toutes les variables nécessaires et non demander à l'utilisateur de lui donner. De cette manière, vous pourrez tester plus facilement chaque sous-programme. Toutes les interactions avec l'utilisateur seront donc programmées dans les menus.
- Vous devez commenter votre code : avant chaque sous-programme et plus généralement, avant chaque bloc, un commentaire devra expliquer ce que fait le code.
- Vous devez respecter les conventions de nommage : les noms des variables et des sous-programmes devront toujours être en minuscules, seules les premières lettres des noms composés pourront être en majuscules (ex : `int nombre, nombreJoueurs;`); les constantes devront être en majuscules (ex : `MAXUSER`), et les noms des structures s'écrivent en minuscules, sauf la première lettre (ex : `Utilisateur`). De plus, les noms doivent être significatifs pour la clarté du code.
- Chaque menu sera un sous-programme et quitter le menu reviendra à quitter ce sous-programme.

2 Dates

Vous devrez choisir votre binôme avant **mercredi 9 Mars**. On créera un espace sur EPREL pour nous communiquer votre binôme.

Le code du projet devra être envoyé, sur EPREL **au plus tard Dimanche 22 Mai, à 23h59**.