Projet de programmation orientée objet

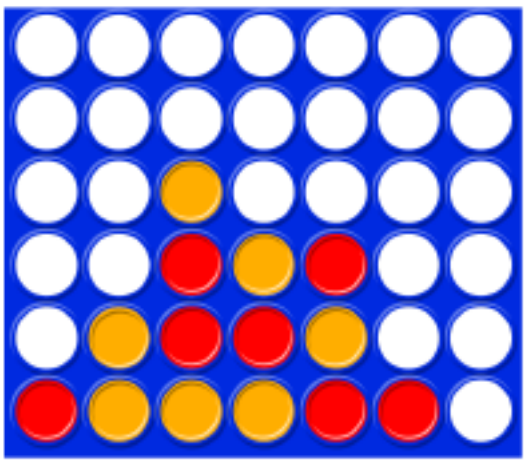
PG220

2020-2021

Introduction

L'objectif est de réaliser un programme permettant de jouer au jeu Puissance 4 depuis le terminal. Puissance 4 est un jeu de plateau dont le but est d'aligner 4 pions sur une grille de 6 lignes et 7 colonnes. Une partie de Puissance 4 se joue en plusieurs manches. Au cours de chaque manche, tour à tour, chaque joueur devra choisir une colonne pour y placer son pion (X ou O), ce dernier glissera jusqu'à la position libre la plus basse dans la grille laissant ainsi la main à son adversaire. Le gagnant de la manche sera celui ayant réussi à aligner au moins 4 de ses pions, l'alignement doit être soit horizontal, soit vertical, soit diagonal.

Dans le cas contraire, si toutes les cases de la grille sont occupées et que personne n'a réussi à aligner 4 pions la manche sera déclarée comme nulle. Dans les deux cas, une nouvelle manche sera lancée. Finalement, le joueur qui remporte la partie est le premier à avoir remporté 3 manches.Il est à noter que c'est toujours le Joueur 1 qui débute dans la première manche, puis ensuite il faut alterner le joueur qui débute la manche.



Exemple de grille 6x7

Etape 1 : Interface Terminal

Le jeu se jouera directement sur le terminal, son interface d'une sorte de shell qui assurera les interactions avec l'utilisateur.

Menu

Faîtes attention de bien respecter et de ne pas modifier le fonctionnement de l'interface terminal (vous ne devez rien demander de plus, et vous devez demander exactement comme cela est spécifié dans le sujet). En effet cette interface terminal servira à faire des tests automatiques.

Avant de commencer à jouer, votre programme doit lors de son lancement afficher un menu permettant au joueur d'indiquer comment seront incarnés les deux joueurs. Pour cela le choix pourra être **humain** ou **ia** . Pour chaque joueur, un nom devra être entré.

Par défaut, le premier joueur aura les pions **X** et le second les pions **O**.

```
Joueur 1?
$ humain Joe
Joueur 2?
$ ia Dark Vador
```

Grille

Une fois le jeu lancé, avant chaque coup, une grille devra s'afficher et le joueur en cours devra saisir la colonne où il souhaite jouer:

```
1 2 3 4 5 6 7
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .

$ 2

1 2 3 4 5 6 7
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. x . . . . .

$ 2

1 2 3 4 5 6 7
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. o . . . . .
. x . . . . .
```

Fin de la partie

Il est à noter qu'il sera possible pour le joueur à tout moment de quitter le programme en utilisant la commande sortir . Le jeu se termine aussi automatiquement quand un des deux joueurs atteint 3 manches gagnées.

Etape 2 : Gestionnaire de Partie

Dans cette partie, vous devrez implémenter un gestionnaire de partie. Son rôle sera de mémoriser l'état de la grille, de demander aux joueurs de jouer, de décider si la partie en cours est finie, et enfin de mémoriser le score. Dans le cas d'une égalité (la grille totalement remplie mais personne n'a réussi à aligner ses pions), la partie sera déclarée nulle, et aucun joueur ne gagne de point. Le joueur qui débute la partie doit alterner, en commençant par le joueur 1.

La première version du jeu devra comporter deux types de joueurs : **humain** et **ia**. Un joueur humain devra entrer le numéro de colonne correspondant à son choix tandis que l'ia jouera une colonne au hasard à chaque tour.

Etape 3 : Historique de la partie

Faîtes attention de bien respecter le nom et le format du fichier d'historique. En effet ce fichier servira à faire des tests automatiques.

A tout moment, il est demandé de maintenir un historique de partie dans un fichier "log.txt" à la racine du dossier dans lequel on a lancé la partie. Ce log doit contenir toutes les informations relatives à la partie sous le format suivant. Il est à noter que le fichier de log doit être remis à zéro lors d'une nouvelle partie. En outre le score est affiché dans l'ordre des joueurs.

```
Joueur 1 est humain John
Joueur 2 est ia Dark Vador
Manche commence
Joueur 1 joue 1
Joueur 2 joue 1
Joueur 1 joue 2
Joueur 2 joue 2
```

```
Joueur 1 joue 3
Joueur 2 joue 3
Joueur 1 joue 4
Joueur 1 gagne // ou “Egalite”
Score 1 - 0
Manche commence
...
Partie finie
```

Etape 4 : Gestion des erreurs

Vous devez faire attention de bien détecter les erreurs lors des saisies utilisateur ou du jeu. Quand une erreur survient, un message d'erreur doit être affiché et la saisie doit être effectuée à nouveau. En outre, les messages d'erreur doivent apparaître dans les logs de la manière suivante. Voici les différents messages d'erreurs.

```
Erreur saisie Joueur 1
Erreur saisie Joueur 2
Erreur saisie colonne xyz
Erreur colonne non valide -1
Erreur colonne pleine 3
```

Évaluation

Ce projet est à réaliser par groupe de 2 (3 si besoin) étudiants. Vous serez évalués sur :

- La conception des classes et l'architecture de votre programme.
- La modularité et l'extensibilité du programme (minimiser l'effort pour inclure de nouveaux type de joueur, de règles etc).
- L'utilisation des concepts fondamentaux de la programmation orientée objet.
- La robustesse de la façon dont vous avez implémenté les fonctionnalités.
- La gestion des erreurs.

Fonctionnalités supplémentaires

Des points bonus pourront être obtenus pour chaque implémentation d'une fonctionnalité proposée dans cette liste:

- Implémenter des ia supplémentaires elles seront identifiées par des noms après le mot ia (ia:random , ia:monkey). Vous pourrez même implémenter des tests qui permettent de faire jouer les ia les unes contre les autres pour voir qui est la meilleure.
- Permettre d'avoir une grille de taille arbitraire (contrainte: lignes x colonnes doit être pair et plus grand ou égal à 8, au moins quatre colonnes) (attention, pour cela ne pas modifier l'interface terminal)
- Permettre d'avoir un nombre arbitraire de joueurs (contrainte au moins 2) (attention, pour cela ne pas modifier l'interface terminal)
- Permettre de configurer le nombre de manche à gagner pour remporter la partie (attention, pour cela ne pas modifier l'interface terminal)
- Implémenter une interface en mode fenêtre (utilisez une autre classe principale que celle pour l'interface terminal)

N'oubliez pas d'expliquer dans le fichier README ce que vous avez implémenté.

Contraintes sur le code

Vous devez absolument respecter ces contraintes. Une pénalité de 2 points sera appliquées sur chaque projet qui ne respecte pas:

- Votre projet contient plusieurs paquetages.

- Les noms des paquetages sont intégralement en minuscules.
- Les noms des classes et interfaces commencent par une majuscule.
- Les noms des méthodes, des attributs et des variables commencent par une minuscule.
- Utilisez le CamelCase pour les noms des classes, interfaces, méthodes et attributs.
- Choisissez une langue pour tout votre projet (de préférence l'anglais), et gardez ce même standard pour tout le projet.
- Les attributs ont une visibilité *private* ou *protected*. Des accesseurs permettent si nécessaire d'accéder en lecture et/ou écriture aux attributs.
- La visibilité des méthodes dépend de leur utilisation.
- Les constructeurs sont définis uniquement s'ils sont nécessaires.
- Le code des méthodes compliquées doit être commenté.

Livrable

Il sera demandé aux étudiants de fournir une archive contenant le code source du programme. À la racine de l'archive, il doit y avoir un fichier README spécifiant les étapes qui ont été complétées et celles qui ne l'ont pas été, ainsi que les fonctionnalités supplémentaires qui ont été implémentées. L'archive doit aussi contenir un fichier java archive puissance.jar qui permet de lancer le jeu à l'aide de la commande java -jar puissance.jar. Le schéma de nommage de l'archive doit être le suivant (pénalité de 3 points en cas de non respect) : pg220-2020-LOGINS.zip, où LOGINS est la liste des logins des membres du groupe, classée par ordre alphabétique et séparés par le symbole "_". L'archive est à déposer, avant le **22/12/2020** à 23h59, via la plateforme `http://moodle.ipb.fr`.

Les tests

L'archive "test.zip" contient le scripts permettant d'exécuter les tests sur une machine possédant une distribution linux avec une interface graphique (X11).