Université Paris Diderot **Programmation Orientée Objet**
L2, EIDD et M1 linguistiques **Année 2015-2016**

Projet

Ghosts (jeu de plateau)

`Version finale`

1 Projet

Le travail est à faire en binôme au sein d'un même groupe de TP. Il aboutira à la production d'un jeu implementé en Java et à une soutenance d'une vingtaine de minutes. Un rapport sera également fourni.

En guise de rapport [1], vous fournirez une **liste des fonctionnalités** que vous avez effectivement implémentées, en expliquant à chaque fois si elles fonctionnent ou non, ainsi qu'un diagramme des classes utilisées. Vous devrez aussi fournir une **javadoc** et votre code devra être soigneusement commenté.

Les détails exacts des modalités et un certain nombre de conseils d'organisation se trouvent en dernière partie.

2 Description

"Ghosts" est un jeu de stratégie sur plateau pour 2 joueurs. Chaque joueur va s'occuper d'une équipe de 8 fantômes qu'il va essayer de mener vers un des trois objectifs de victoire. Les deux équipes comprennent chacune quatre bons fantômes (bleus) et quatre mauvais fantômes (rouges). Les couleurs des fantômes sont secrètes pour l'autre joueur.

Il y a 3 manières différentes de remporter une partie et les joueurs devront faire preuve d'un bon sens de la stratégie pour l'emporter.

Un joueur gagne si l'une de ces trois situations se produit :

— un bon fantôme atteint une sortie du côté adverse ;

— ce joueur capture tous les bons fantômes de l'adversaire ;

— les 4 mauvais fantômes du joueur sont capturées par l'opposant.

Dans la suite de cette section, nous allons décrire plus précisément les règles du jeu (comment est formé le plateau, quelles en sont les sorties, comment capturer un fantôme, ...).

2.1 Position initiale

Les fantômes sont placés dans un plateau de jeu de taille 6x6, comme illustré par la Figure 1, sans montrer à l'adversaire le type de chaque fantôme. Chaque case est identifiée de manière unique par des coordonnées alphanumériques (comme pour le jeu d'échecs).

1. Un rapport détaillé dans lequel on expliquerait *comment* les fonctionnalités sont implémentées *n'est pas* demandé.

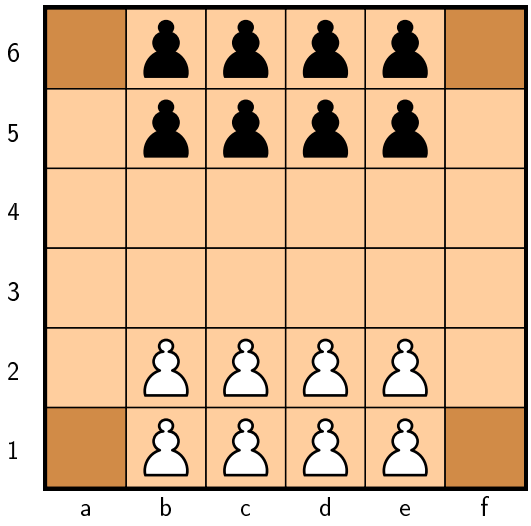

FIGURE 1 – Configuration initiale du jeu.

Les cases de départ pour chaque joueur sont représentées par deux rectangles de 8 cases :

— $[b1 \times e2]$ pour le joueur 1.

— $[b5 \times e6]$ pour le joueur 2.

La configuration des fantômes (bons et mauvais) sur les cases de départ est choisie par le joueur lui-même au début de la partie.

Les sorties sont placées aux 4 coins du plateau de jeu. Un fantôme peut sortir s'il se place sur une case de sortie du côté du joueur adverse.

2.2 Règles

À chaque tour, l'un de vos fantômes est déplacé d'une case. Vous capturez un fantôme adverse en vous plaçant sur la case qu'il occupe.

Les mouvement autorisés sont : une case en avant, en arrière ou sur le coté. On ne peut pas déplacer une pièce en diagonale.

3 Le jeu à implémenter en Java

Votre programme doit savoir gérer les fonctionnalités de base du jeu, à savoir :

— L'application doit être capable de jouer automatiquement (à partir d'un fichier texte) ou interactivement (avec 2 joueurs qui vont placer à la main les pièces et qui vont choisir les mouvements). Un exemple de fichier est montré dans la Figure 2.

— Votre programme devra aussi avoir un mode "triche" qui permettra de savoir qui sont les bons ou mauvais fantômes et donc vérifier que le

input1.gf

```
1  # Player1
2  G G B G
3  B B G B
4
5  # Player2
6  G G G G
7  B B B B
8
9  # Move
10 1 - a2,a3 - a5,a4
11 2 - c2,c3 - c5,c4
```

FIGURE 2 – Position initiale et liste des mouvements

programme respecte les règles du jeu.

— Une interface graphique est attendue, éventuellement les déplacements des pièces peuvent êtres saisis en mode texte (soit dans un terminal, soit dans une boîte de l'interface graphique.)

Il sera tenu compte dans la note de la qualité de programmation et de la clarté du code. En particulier, votre programmation doit impérativement utiliser l'héritage (c'est-à-dire classes dérivées et/ou classes abstraites et/ou interface). Dans la mesure du possible votre programmation doit permettre de programmer des variantes de ce jeu sans tout refaire et avec le minimum de copié-collé possible. Pour vous donner une idée, voici quelques variantes possibles[2] : ajouter un type de fantômes, changer les règles de fin de jeu, agrandir le plateau...

La beauté graphique du jeu n'a pas d'importance, mais il faut tout de même que l'on puisse facilement comprendre l'état du jeu. Enfin, une petite partie du barème (environ 2 points) sera accordée à la personnalisation du projet : implémentation d'extensions, variantes, fonctionnalités en plus, etc...

4 Modalités du rendu et soutenances – conseils d'organisation

Les dates exactes du rendu du projet et des soutenances vous seront communiquées ultérieurement.

4.1 Ce qui est à rendre

Vous devrez rendre le **projet** (y compris le code). Comme précisé en introduction, en guise de rapport vous fournirez une **liste des fonctionnalités** que vous avez effectivement implémentées, en expliquant à chaque fois si elles fonctionnent ou non, ainsi qu'un diagramme des classes utilisées. Vous devrez aussi fournir une **javadoc** et votre code devra être soigneusement commenté.

2. Nous ne vous demandons pas de programmer les variantes !

4.2 Soutenances

Les soutenances se feront à deux, mais la note pourra être individualisée si le travail a été trop inégalement réparti, naturellement chacun doit être capable de répondre à toutes les questions.

Il nous faudra pouvoir tester le projet durant les soutenances, il est donc préférable qu'il fonctionne sur les machines du script si vous êtes en L2 ou sur celles de l'UFR pour les EIDD et les linguistes. Ceci même si vous avez prévu d'apporter un portable pour l'occasion car il peu tomber en panne au mauvais moment.

4.3 Conseils pour réaliser le projet

1. Prenez le temps de réfléchir à la conception du projet avant de coder.
2. Réfléchissez aussi à la manière de répartir le travail entre les deux personnes du binôme. Faites des points réguliers entre vous.
3. Compilez et testez régulièrement ce que vous faites.
4. Vous pouvez éventuellement dans un premier temps vous contenter d'une interface texte pour pouvoir tester votre jeu et ajouter l'interface après. Si jamais vous n'arriviez pas à obtenir une interface graphique qui fonctionne, vous pourrez présenter le projet juste avec une interface texte. Vous ne pourrez pas avoir tous les points, mais vous en aurez plus que si vous présentez un projet non testable.
5. Enfin, pensez à faire des sauvegardes fréquentes, sur au moins deux supports différents et en particulier sauvez les versions testables de votre projet même s'il reste des corrections à y apporter.