IUT Nancy-Charlemagne DUT Informatique S2
Programmation Objet 2 (2020-2021)

Projet Pentamino

Présentation

Le but du projet est de mettre en œuvre les concepts du cours (Listes, Exceptions et Entrées/Sorties) en programmation objet (classes abstraites et interfaces).

Le projet est un jeu inspiré de Tetris et Pentomino, multi-joueurs.

Le jeu se joue en ligne de commande à l'aide d'un Scanner. En cours de jeu, à chaque tour, l'utilisateur choisit la pièce suivante à poser sur la grille parmi une liste de pièces proposées et fournit les coordonnées x et y de l'endroit où poser la pièce.
On considère que des pièces peuvent se superposer, c'est-à-dire se recouvrir partiellement (mais 2 pièces identique ne peuvent pas se recouvrir totalement).

Il y a 3 types de joueurs avec les règles suivantes:

- Débutant : Les pièces peuvent sortir de la grille et se superposer.
- Intermédiaire : Les pièces peuvent sortir de la grille mais pas se superposer.
- Avancé : Les pièces ne peuvent ni sortir de la grille, ni se superposer.

Les principales classes du jeu sont décrites après. Il doit y avoir des classes abstraites et des interfaces.

Jeu

Un jeu possède une liste de joueurs.
Il y a plusieurs types de joueurs.
Au démarrage le jeu charge une sauvegarde ou crée un jeu vide.
Le jeu affiche alors la liste de joueurs triés par ordre alphabétique et par ordre de score moyen décroissant. L'affichage doit être bien formaté, bien aligné : les noms des joueurs prennent la même place, le nombre de chiffre après la virgule est maitrisé… On peut créer un nouveau joueur ou en sélectionner un.

Exemple d'affichage possible :

```
Joueurs par ordre alphabétique:
 0:  Jean-Jac (002,00)
 1:      Paul (001,50)
 2:    Pierre (006,00)
Joueurs triés par score moyen:
 0:      Paul (001,50)
 1:  Jean-Jac (002,00)
 2:    Pierre (006,00)
0: quitter le jeu, 1: créer un nouveau joueur, 2: choisir un joueur. Votre choix ?
```

Joueur

Il y a plusieurs types de joueurs : débutant, intermédiaire et avancé.
Chaque type de joueur correspond à une classe concrète.
C'est la classe de joueur qui détermine comment se passe le jeu : Elle reçoit des Exceptions de la partie mais les intercepte et les gère à sa manière. Par exemple si l'exception CaseOccupeeException

remonte lors de la pose d'une pièce, la classe JoueurAvance demande le retrait de la pièce avant de passer au tour suivant (c.-à-d. avant de demander quelle pièce poser ensuite), alors que la classe JoueurDebutant ne retire pas la pièce superposée.
Chaque joueur possède des parties.
La manière d'établir le score d'un joueur est libre mais on peut tenir compte du type de joueur, de la somme du score des parties.
Une fois un joueur sélectionné, l'utilisateur choisit une partie ou en commence une nouvelle.

Partie

Une partie est composée d'une grille (un tableau de caractères à 2 dimensions), une liste de pièces posées et une liste des futures pièces à poser. À l'instanciation la liste « posées » est vide et la liste de pièce « à poser » est remplie de façon aléatoire (ou autre…). Elle peut être complétée par la suite.
La liste « posées » ne doit pas contenir 2 pièces identiques (même attributs, même coordonnées).
A chaque tour d'une partie, l'utilisateur choisit la pièce suivante à poser sur la grille parmi la liste de pièces « à poser » et fournit les coordonnées x et y de l'endroit où poser la pièce. La classe Partie possède une méthode ajouterPiece(n,x,y) qui reçoit le numéro de la pièce à déplacer de la liste « à poser » vers la liste « posées » et ses coordonnées sur la grille. Elle peut remonter les exceptions CaseDejaRemplieException ou PieceDebordeException. Dans ce cas, la classe Joueur peut intercepter l'exception et invoquer la méthode retirerDerniere Piece() qui va remettre la dernière pièce dans la liste « à poser » s'il n'est pas débutant. Une méthode privée actualiserGrille() peut être envisagée et utilisée lors du placement de pièce pour détecter les superpositions et débordement, aussi pour faciliter l'affichage de la grille.
Une autre approche consisterait à considérer qu'aucune pièce n'a été posée en cas d'exception et qu'il faut alors forcer la pose de la pièce dans le cas débutant.
La manière d'établir le score d'une partie est libre mais on peut se contenter du nombre de pièces posées. Une partie n'a pas de fin : la grille peut être pleine, mais on peut toujours retirer la dernière pièce.

Pièce

Les pièces sont des polyominos (tretraminos -à 4 carrés- comme dans Tetris et/ou pentomino -à 5 carrés-). Une pièce possède des coordonnées x, y, une lettre (qui identifie la pièce à l'affichage) et une liste de carrés.
Chaque forme de pièce correspond à une classe concrète.
La forme de chaque pièce (liste de carrés) est lue à l'instanciation en lisant un fichier texte.
Par exemple la classe U.class lit le fichier U.txt suivant :

```
#_#
###
```

La rotation de pièce n'est pas demandée.

Carré

Un carré est un composant d'une pièce.
Il possède des coordonnées x, y relative à l'origine de la pièce.
La classe du Carre est aussi simple que les classes du cours : Point, UnPoint, PointEgal…

Programme principal

Au démarrage un main() charge la sauvegarde ou instancie un nouveau jeu vide.
Le main peut être placé dans la classe Jeu().
A la fin du jeu, avant de quitter, une sauvegarde de l'objet Jeu est faite dans un fichier binaire de sauvegarde.

Grilles, coordonnées et affichage

Les coordonnées sont des entiers pas nécessairement négatifs. L'axe des ordonnées peut être orienté vers le bas car les fichiers texte de description des pièces se lisent de haut en bas.
La taille de la grille peut être fixe (8x8, 10x10 ou autre…). Les coordonnées des pièces sont relatives à la grille et sont modifiées lors du placement de la pièce.
Les coordonnées des carrés sont relatives à la pièce à laquelle ils appartiennent.
L'origine [de coordonnées (0,0)] de la grille et des pièces peut être en haut à gauche ou en bas à gauche suivant les conventions choisies.
A l'affichage de la grille, les carrés de chaque pièce sont représentés par la lettre stockée dans la pièce. Une case vide peut être représentée par un point.

Exemple d'affichage possible de la grille :

```
.T.U.UL...
.TTUUULLL.
UTUT......
UUUTT.....
.L.T......
.LLL......
..........
..........
..........
..........
```

Exemple d'affichage des pièces suivantes :

```
Pièces suivantes :
0: pièce 0 :
#..
###
1: pièce 1 :
#.
##
#.
2: pièce 2 :
#.#
###
0: quitter la partie, 1: placer pièce. Votre choix ?
```

Barème

Le barème est donné à titre indicatif et peut changer :

- Code fonctionnel (5 points)
- Respect du sujet et des règles du jeu (5 points)
- Diagramme UML au format image et lisible (2 points)
- Une javadoc complète (2 points)
- Mise en œuvre des concepts du cours : (6 points)
    - Des collections
    - Des exceptions
    - Des entrées et sorties fichiers
    - Des interfaces et/ou des classes abstraites
    - Du formatage pour faire de belles sorties consoles

Des points peuvent être retirés si la classe principale est trop longue.

Déroulement

Pour réaliser ce projet, vous disposerez de 5h de TD à la fin du module. Lors de la 1ère séance, vous commencerez votre projet par une phase de réflexion sur la conception du jeu pour aboutir à un diagramme UML. Durant les 4 heures de TD restantes, votre enseignant sera à votre disposition pour répondre à vos questions et vous débloquer en cas de problème. Vous serez en autonomie, nous ne vous donnerons pas une suite d'instructions pour construire le projet.

A l'issue de ce projet, nous vous demanderons de rendre l'ensemble des sources documenté et accompagné du diagramme de classe UML. Possibilité d'utiliser un dépôt git.

Le TP est à rendre, individuellement ou par binôme, le dimanche 04 avril à 23h55 AU PLUS TARD. AUCUN retard ne sera autorisé, le dépôt sera automatiquement fermé au-delà de la date et de l'heure indiquées.

Format à suivre OBLIGATOIREMENT pour le nom du fichier à déposer :

Le nom de votre fichier, si vous avez travaillé seul : S2A_NOM.zip (ou S2B, S2C, S2D selon votre groupe)

Le nom de votre fichier si vous avez travaillé en binôme : S2A_NOM1_NOM2.zip (ou S2B, ...)