# IFT1135 TP1

**Remise** : 19 octobre, 22h00

**Barème** : Ce devoir sera corrigé sur 34.

**Équipe** : Le travail individuel ou en équipe de deux ou trois est permis

**Remise électronique** : Remettez le fichier Jeu.java dans Studium.

**Varia** : L’absence de remise électronique sera pénalisée de 34 points.

À noter : l’envoi par courriel n’est pas considéré comme une remise électronique, donc -34 points.

Les remises en retard seront pénalisées de 5 points par jour.

Si vous utilisez du code qui n’est pas le vôtre, citez vos sources.

Pour ce premier devoir, vous allez écrire le code java pour gérer un jeu de Tic-Tac-Toe, et générer les coups pour le joueur « O », celui qui joue en deuxième.

Sur le site Studium du cours, vous disposez des fichiers **TicTacToe.java** et **GrilleDeJeu.java**.

Le fichier **TicTacToe.java** contient l’interface que votre classe devra implémenter pour communiquer avec l’interface graphique fournie.

Le fichier **GrilleDeJeu.java** contient le code java, utilisant Swing et AWT, pour interagir avec le joueur « X ».

Votre code doit obligatoirement être dans une classe nommée **Jeu** qui va implémenter l’interface **TicTacToe**.

Votre classe va servir à tenir compte des positions des coups joués, ainsi que du nombre de coups joués.

L’index pour les différentes positions sur la grille est comme suit :

|  |  |  |
| --- | --- | --- |
| 0 | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |

La classe **Jeu** doit avoir un constructeur sans paramètre.

Les méthodes de l’interface à implémenter sont :

**initialise** : Méthode pour (ré)initialiser le jeu avant de débuter une nouvelle partie.

**isPartieNulle** : Méthode qui va déterminer si toutes les positions de la grille sont occupées sans qu’il y ait de gagnant. Retourne **true** ou **false**.

**setX** : Méthode qui sera appelée par l’interface graphique pour communiquer la position choisie par le joueur « X ». Cette méthode reçoit paramètre l’entier correspondant à l’index de la position sur la grille.

**gagnant** : Méthode qui va recevoir la chaîne de caractères représentant le joueur, soit « X » ou « O ». La méthode va retourner **true** si une ligne, une colonne ou une diagonale est entièrement occupée par ce joueur. Dans ce cas, un tableau d’entier va contenir les 3 index de la ligne, colonne ou diagonale gagnante. Cette méthode va retourner **false** si le joueur n’a pas (encore) gagné.

Les 8 combinaisons d’index de positions correspondant à une victoire sont :

(0, 1, 2), (0, 4, 8), (0, 3, 6), (6, 7, 8), (6, 4, 2), (3, 4, 5), (1, 4, 7) et (2, 5, 8).

**getO** : Méthode qui doit déterminer le prochain coup de « O » et qui retourne l’index de la position choisie.

Voici un algorithme pour déterminer un coup du joueur « O » :

Premier coup :

Si position 4 libre, alors position 4.

Sinon, alors un coin (0, 2, 6 ou 8)

Deuxième coup

S’il ne manque qu’un coup à X pour gagner, alors le bloquer.

Ex : X : 0 et 1 et O : 4 donc position 2

Sinon et que X occupe 2 positions sur une diagonale et O la troisième alors :

Si O occupe la position 4, alors choisir position 1, 3, 5 ou 7.

Sinon, alors choisir un des coins de libre (0, 2, 6 ou 8)

|  |  |  |
| --- | --- | --- |
| X |  |  |
|  | O |  |
| ***O*** | X |  |

Si O occupe le centre, X occupe un coin et X occupe (1, 3, 5 ou 7), alors choisir le coin le plus près des positions de X.

ex : pour la situation suivant, position choisie : 6

Par symétrie : si X occupe (1, 8) et O occupe 4, donc position 2, etc. (il y a 8 combinaisons).

|  |  |  |
| --- | --- | --- |
|  |  |  |
| X | O |  |
| ***O*** | X |  |

Si O occupe le centre et X occupe 2 positions qui ne sont pas des coins, et ne sont pas sur la même ligne ou colonne, alors la position jouée est le coin contigu aux 2 cellules de X.

Par exemple : si X occupe les positions 3 et 7 et O occupe 4, position choisie : 6.

Il y a quatre combinaisons correspondant à ce cas.

Troisième et quatrième coup

On cherche s’il y a une position qui nous donne une victoire immédiate.

Sinon, on cherche une position qui, si elle n’est pas occupée immédiatement, donnera la victoire à « X » à son prochain coup.

Sinon, on cherche une ligne, colonne ou diagonale où il y a un « O » mais pas de « X ».

Sinon, on joue n’importe quelle position libre.

**testDebug** : Méthode facultative (à implémenter vide si vous ne vous en servez pas) qui va recevoir les indices correspondants à des X et des O de façon à pouvoir tester un cas particulier. Si vous implémentez cette méthode, assurez-vous que l’état de votre objet de la classe **Jeu** correspondra à cette séquence de choix. Le tableau d’entiers reçu en paramètre contiendra les coups du joueur X à ses positions paires et les coups du joueur Y à ses positions impaires.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| X | O |  |
| O | X |  |

Par exemple, pour l’état suivant :

Le tableau reçu en paramètre serait comme suit : indicesCoups = {3,4,7,6};

C’est-à-dire, des X aux positions 3 et 7 de la grille et des O aux positions 4 et 6.

**À remettre** : votre fichier **Jeu.java**

Critères de correction :

Bon fonctionnement de la méthode **initialise**. 3 points

Bon fonctionnement de la méthode **setX** 3 points

Bon fonctionnement de la méthode **getO** 16 points

Si O peut perdre (-3 points)

Conserve correctement les coups joués 2 points

Bon fonctionnement de la méthode **isPartieNulle** 2 points

Bon fonctionnement de la méthode **gagnant**

Bon résultat 4 points

Positions gagnantes transmises 2 points

Lisibilité et présentation du code. 2 points

**Remarque** : Puisque vous n’avez pas à remettre les fichiers **TicTacToe.java** et **GrilledeJeu.java**, toutes modifications que vous allez y faire ne seront pas pris en compte lors de la correction de votre travail.