Projet Puissance4

Dans ce projet, on se propose d'étudier et d'implémenter en JAVA un puissance 4. Le but du jeu est d'aligner 4 pions sur une grille comptant 6 rangées et 7 colonnes (voir les règles ici). Le projet sera divisé en quatre parties :

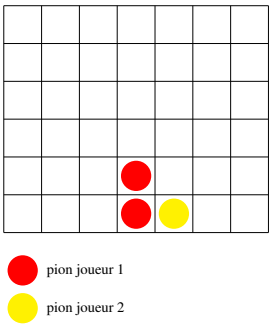
— la représentation du jeu (de la grille en JAVA) et son affichage à l'écran,
— la détection de la fin de la partie par le gain d'un des joueurs ou par match nul,
— la boucle principale du jeu,
— une ia.

1. Représentation du jeu

Exercice .1 Représentation et initialisation de la grille

On se propose de représenter la grille par un tableau bidimensionnel d'entiers, de 6 lignes et 7 colonnes. La première ligne du tableau correspondant à la ligne du bas de la grille de puissance 4. Les joueurs sont les joueurs 1 et 2. On représente un pion du joueur 1 (respectivement du joueur 2) dans la grille du puissance 4, par un 1 (respectivement 2) dans la case correspondante du tableau. On représentera une case vide du puissance 4 par un 0 dans la case correspondante du tableau.

Exemple :




```
[[ 0, 0, 0, 1, 2, 0, 0 ],
 [ 0, 0, 0, 1, 0, 0, 0 ],
 [ 0, 0, 0, 0, 0, 0, 0 ],
 [ 0, 0, 0, 0, 0, 0, 0 ],
 [ 0, 0, 0, 0, 0, 0, 0 ],
 [ 0, 0, 0, 0, 0, 0, 0 ]]
```

Situation Représentation JAVA

1. Dans votre programme, déclarer la variable globale *grille* de type tableau bidimensionnel d'entier.

2. Écrivez une fonction *initialiseGrille*, sans paramètre et ne renvoyant rien, initialisant la variable globale *grille* par un tableau bidimensionnel de 6 lignes et 7 colonnes contenant uniquement des 0.

Exercice .2 Représentation d'un coup d'un joueur

Lorsqu'un joueur joue, il lache son pion dans une colonne qui tombe et rempli la première case libre de cette colonne (première en partant du bas de la grille). Avec notre représentation, cela revient à chercher dans la colonne correspondante dans *grille*, la première case contenant un 0 et de changer cette valeur en le numéro du joueur.

1. On suppose que lorsqu'on joue un coup, il y a toujours de la place dans la colonne correspondante. Écrivez une fonction java *jouer* prenant en paramètre le numéro du joueur (entier) ainsi qu'un numéro de colonne et qui fait jouer le coup correspondant dans *grille*.

Exercice .3 Affichage de la grille

On souhaite avoir une visualisation de l'état de la grille avant de pouvoir jouer. Pour cela, on souhaite que la grille s'affiche de cette manière :

— la première ligne doit être en bas,
— une case contenant un 0 doit être représentée par un espace, un 1 par un X et un 2 par un O,
— au bout de chaque ligne et entre chaque ligne, les cases doivent être séparées par le caractère | (AltGr+6),
— on doit afficher une ligne supplémentaire contenant les numéros des colonnes.

Exemple :

```
                                  | | | | | | | |
[[ 0, 0, 0, 1, 2, 0, 0 ],         | | | | | | | |
 [ 0, 0, 0, 1, 0, 0, 0 ],         | | | | | | | |
 [ 0, 0, 0, 0, 0, 0, 0 ],         | | | | | | | |
 [ 0, 0, 0, 0, 0, 0, 0 ],         | | | |X| | | |
 [ 0, 0, 0, 0, 0, 0, 0 ],         | | | |X|O| | |
 [ 0, 0, 0, 0, 0, 0, 0 ]]          0 1 2 3 4 5 6
```

Représentation JAVA Affichage à l'écran

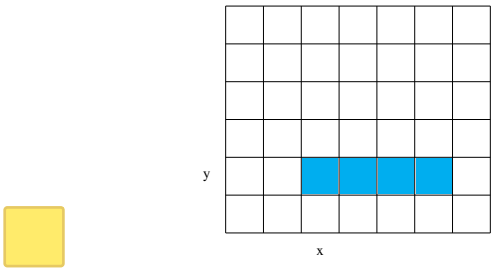
1. Écrivez une fonction *afficheGrille* ne prenant pas de paramètre et ne renvoyant rien, et affichant la grille à l'écran.

2. Fin de partie

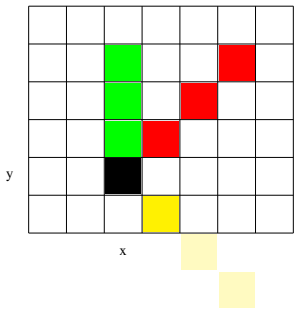
Exercice .4 Détection d'un alignement de 4 cases

Un joueur gagne quand il a réussi à aligner 4 pions, ce qui revient dans notre représentation à avoir 4 cases alignées avec le même numéro de joueur.

Tout d'abord, on s'intéresse aux alignements horizontaux. Pour vérifier si le joueur à réussi à faire un alignement horizontal, on se propose de tester, pour toutes les cases, si un alignement 'débute' à cette case, c'est à dire que cette case et les 3 cases vers la droite contiennent le même joueur :



1. Écrivez la fonction *aGagneHor* prenant en paramètre un numéro de joueur, un numéro de ligne y, un numéro de colonne x et qui retourne 'vrai' si il y a un alignement horizontal de 4 cases dans *grille*, qui débute à la case (y,x). **ATTENTION**, on vérifiera qu'on peut faire un alignement sans sortir du tableau.



2. En utilisant la même méthode, écrivez les 3 autres fonctions *aGagneVer*, *aGagneDiagMont* et *aGagneDiagDesc* prenant en paramètre un numéro de joueur, un numero de ligne y, un numéro de colonne x et retournant vrai si il y a un alignement vertical, diagonal montant ou diagonal descendant de 4 cases débutant à la case (y,x) dans *grille*. De même que pour la fonction précédente, on vérifiera que les alignements ne sortent pas du tableau avant de les tester.

Maintenant que nous pouvons vérifier tous les types d'alignement à partir d'une case, il reste à tester ces fonctions sur toutes les cases de *grille*.

3. Écrivez la fonction *aGagne* prenant en paramètre un numéro de joueur et qui retourne vrai si il existe un alignement de 4 cases du joueur dans *grille*.

Il se peut que les deux joueurs remplissent la grille sans qu'il y ait de gagnant. Dans ce cas, il y a 'match nul'.

4. Comment détecter un match nul à partir de *grille* ?

5. Écrivez une fonction *matchNul* ne prenant aucun paramètre et retournant 'vrai' si il y a match nul.

3. Boucle de jeu

Exercice .5 Boucle de jeu

On considère que le jeu possède 2 paramètres internes :
— la grille,
— le joueur courant.

Le jeu part de la situation initiale suivante :
— le joueur courant : le joueur 1,

— la grille : un tableau bidimentionnel de 6 lignes par 7 colonnes rempli de 0.

Ensuite le joueur courant joue son coup, et modifie la grille puis le joueur courant change. Et ainsi de suite jusqu'à ce que la partie soit finie.

il y a au moins un zero -> pas terminé

1. Quel est le test de continuation de cette boucle de jeu ?

Pour jouer un coup, on souhaite avoir une visualisation de la grille, suivi du texte *"Quel coup pour le joueur . . . ?"*. Où "..." représente le numéro du joueur courant. Ensuite, on souhaite récupérer dans une variable de type entier, le nombre (entre 0 et 6) qui sera tapé au clavier par le joueur correspondant.

2. Écrivez en JAVA la séquence d'opération qui affiche la grille, affiche le texte puis récupère dans une variable un entier taper au clavier.

3. Écrivez en JAVA l'opération qui permet de jouer le coup correspondant (en utilisant la fonction que vous avez déjà définie).

Une fois que le coup est joué, on change de joueur. *Remarque : la fonction* $x \to 3-x$ *peut vous être utile.*

4. Écrivez en JAVA la boucle de jeu.

Quand on sort de la boucle, c'est soit que l'un des joueurs à gagné, soit que le match est nul. On souhaite donc afficher à l'écran la grille finale, ainsi que le texte "Le joueur . . . a gagné" ou "Match nul".

5. Écrivez en JAVA la fonction *jeu* ne prenant pas de paramètre et ne renvoyant rien. Cette fonction doit initialiser la grille, permettre de jouer la partie et afficher à l'écran la grille finale suivi du texte correspondant.

6. Écrivez une fonction *main* pour tester votre fonction *jeu*.

4. IA

Exercice .6 ia

On souhaite pouvoir jouer contre l'ordinateur et nous allons donc implémenter un semblant d'ia.

1. Écrivez une fonction *joueCoupRandom* qui choisit une colonne libre au hasard et qui place une pièce du joueur courant dans la colonne.

2. Modifiez le *main* pour inclure la possibilité de jouer contre l'ordinateur, ce dernier utilisant la fonction *joueCoupRandom* à chacun de ses tours.

Pour améliorer le comportement de l'ia, on souhaite qu'avant de choisir une colonne au hasard elle regarde si elle peut gagner en 1 coup et si oui qu'elle joue ce coup.

3. Écrivez une fonction *joueCoupRandom2* qui implémente le comportement décrit ci-dessus.

On souhaite aussi que dans la mesure du possible, l'ordinateur ne joue pas de coups qui permette au joueur suivant de gagner de suite.

4. Écrivez une fonction *joueCoupRandom3* qui implémente le comportement décrit ci-dessus.

5. Essayez de battre l'ia.