Ce TP noté est à réaliser sur les dernières séances de TP.
Le projet est à réaliser sous Eclipse, en Java (version 1.8). Une programmation objet mettant en œuvre les concepts d'héritage, de classe abstraite, d'interface et de polymorphisme est exigée.
Ce projet consiste à programmer un jeu en version console puis en version graphique.

Un jeu de chasse au trésor

I. Le jeu

But du jeu :

Le but du jeu est de faire se déplacer plusieurs personnages (des aventuriers) sur un damier, tous à la recherche d'un trésor qui se trouve dans une case du damier inconnue des personnages. Le damier comprend également des murs qui feront obstacle aux déplacements des personnages. Chaque personnage se déplace tout seul, c'est à dire sans intervention de l'utilisateur. Son parcours est conditionné par les cases qu'il rencontre.
Les personnages se déplacent d'une case à la fois dans une des huit directions possibles autour de sa case.

4	3	2
5	**P**	1
6	7	8

Fig. 1 *Les huit directions possibles du personnage P*

Lorsqu'un personnage veut se déplacer sur une case, la case visée peut être :
- libre auquel cas le personnage peut l'occuper ;
- déjà occupée par un autre personnage auquel cas le personnage devra changer de direction ;
- une pierre faisant partie d'un mur que le personnage devra contourner ;
- un bord du damier auquel cas le personnage devra changer de direction.

Déplacement d'un personnage :

Quand un personnage se déplace (voir Fig.2), il vise la case cible selon sa propre direction.

4	3	2
5	**P**	1
6	7	8

Fig. 2 *Le personnage P de direction1 vise la case grisée (case cible)*

La case cible « réagit » au déplacement prévu par le personnage, selon son type.
Réaction de la case cible selon son type :
1. Type *Libre* : le personnage peut l'occuper.
2. Type *Libre* mais occupée par un autre personnage : le personnage ne bouge pas, il est réorienté au hasard dans une des huit directions possibles afin de l'éloigner du trésor.

3. Type *Pierre* (faisant partie d'un mur) : le personnage ne bouge pas, il est réorienté dans une direction (vers un des deux bouts du mur) qui lui permette à la fois de contourner le mur et aussi de se rapprocher le plus du trésor.
4. Type *Bord* du damier : le personnage ne bouge pas, il est réorienté dans la direction symétrique à la sienne. Par exemple, la direction symétrique à la direction 1 est la direction 5
5. Type *Trésor :* le personnage a gagné. C'est la fin de la partie

Autrement dit, faire évoluer le jeu d'un tour consistera à déplacer chaque personnage un par un, en gérant leurs déplacements selon la case cible visée (ou case cible) par chaque personnage. La partie se termine dès qu'un personnage a comme case cible : la case Trésor.

Les éléments du jeu :
Le damier de dimension *dim* comprend *dim* X *dim* cases (de type *Libre*, *Pierre* ou *Trésor*), et aussi les cases pour les bords.
Les murs sont horizontaux ou verticaux, constitués de cases de type *Pierre*. Pour ne pas bloquer un personnage, les murs ne seront jamais collés les uns aux autres. De plus, ils ne sont jamais collés non plus aux bords du damier. Autrement dit, il y a toujours un point de passage à chaque bout du mur pour permettre de le contourner.
Les personnages sont stockés dans une liste de personnages. Chaque personnage a une direction initialisée au hasard.
Il n'y a pas de superposition sur une case : une case occupée est ou bien une pierre, ou bien la case trésor ou encore une case libre, occupée ou pas par un personnage.

II. Guide pour le codage du jeu :

Dans ce paragraphe quand il est dit qu'une classe comprend (ou inclut) des attributs, ce n'est pas exhaustif. La classe peut éventuellement comprendre d'autres attributs supplémentaires.

1. La classe **Damier** (Board en anglais) comprend une collection de cases et la liste des personnages.

2. Chaque **personnage** est orienté dans une direction.
Ainsi, La classe Personnage (Hunter en anglais) comprend une Direction.
L'affichage de chaque personnage se fera par une lettre majuscule. Le premier personnage est affiché A, la second B, etc.

3. La case **Tresor** (Treasure, symbole 'T') .
Le personnage qui vise la case Trésor a gagné. C'est la fin de la partie.

4. Les **cases** (Cell en anglais) sont organisées par héritage. Coder une classe abstraite (*Cell*), les classes modélisant des cases hériteront de celles-ci.
Les cases quand elles deviennent la cible d'un personnage, doivent réagir à cette sollicitation. Ainsi, chaque case devra implémenter l'interface Questionnable ci-dessous :

```
public interface Questionnable {
      public void process (Hunter h);
      // la case modifie le personnage h
      // un message est affiché à la console pour expliciter
      // la(les) modification(s) du personnage
}
```

Chaque case inclut un symbole qui sera affiché (en mode console) pour représenter la case à l'écran.
La description du *process* de chaque case est décrit ci-dessous

4.1. **Case libre** (Free, de symbole '.')

Une case libre non occupée par un personnage peut recevoir un personnage. Quand un personnage arrive sur une case libre non occupée (case cible), alors cette case lui attribue une direction qui oriente le personnage sur une des huit cases contiguës à la case cible. Cette direction oriente alors le personnage vers la case la plus proche du trésor parmi les huit. Le personnage prend la position de la case cible.

Calcul de la distance entre deux cases :

C'est un nombre entier de cases, égal au carré de la distance euclidienne en nombre de cases.
Par exemple, la «distance» calculée de cette manière entre les deux cases marquées d'une croix dans l'exemple suivant, est égale à : :20=($4^2 + 2^2$)

	X				
					X

4.2. **Case libre occupée par un personnage.**

Lorsque le personnage a pour case cible une case libre mais occupée par un personnage, alors il ne bouge pas mais sa direction est modifiée. Il prend alors une autre direction définie au hasard.

4.3. **Case Pierre** (Stone, de symbole '#')

Lorsque le personnage a pour case cible une pierre, alors il ne bouge pas. Il est réorienté dans une direction (vers un des deux bouts du mur) qui lui permette à la fois de contourner le mur et aussi de se rapprocher le plus du trésor.
Exemple :

T					
	#	#	#	#	
			P		

Le personnage p s'il vise une des trois cases en grisé sur le schéma ci-dessus, est redirigé dans la direction 5, c'est-à-dire à gauche., pour contourner le mur tout en se rapprochant du trésor T.
Exemple :

T					
	#	#	#	#	
					P

Le personnage p s'il vise la case en grisé sur le schéma ci-dessus, peut-être redirigé dans la direction 3 ou 5 pour contourner le mur. Il sera redirigé vers la direction 3 (en haut car le rediriger dans la direction 5 l'amènerai à réinterroger toutes les cases du mur et serait un moins bon choix. Autrement dit, quand un personnage est vers une voie de passage pour contourner un mur, alors il prend la direction vers cette voie de passage.

4.4. **Case coté** (Side, de symbole '+'), c'est une case située sur le bord du damier.

Lorsque le personnage vise une case située sur le bord du damier, alors il ne bouge pas et est redirigé dans une direction symétrique à sa direction actuelle. Il repartira donc dans la direction opposée.

III. Travail demandé

Les programmes doivent être codés en Java (version 1.8) avec l'IDE Eclipse.

Partie 1 : jeu en mode console

Analyse : Vous rendrez d'abord l'analyse de votre projet pour la version en mode console. Cette analyse prendra la forme d'un diagramme de classes complet de tous les objets que vous envisagez de développer, et décrivant les choix principaux d'implantation retenus pour coder le projet. Cette analyse est à déposer sur Moodle, avant le **lundi 13 avril à midi.**

Développement :

Le jeu en mode console initialisera le damier selon l'exemple ci-dessous (Fig. 3).

Exemple :

Damier de dimension 12 avec 2 murs et 3 personnages (A,B,C). Le trésor est à la position (4,2).

```
++++++++++++++
+............+
+............+
+...#........+
+.T.#.....A..+
+...#........+
+...#........+
+...#..###...+
+...#...C....+
+...#....B...+
+............+
+............+
+............+
++++++++++++++
```

Fig 3. *Affichage (mode console) d'un damier*

A l'issue de chaque tour une trace est affichée à l'écran.

Exemple : Trace du premier tour en mode console :

```
Personnage A :
Hunter [4 10] dir 3
Case cible : 3 10
Best dir 6
 -> Hunter [3 10] dir 6

Personnage B :
Hunter [9 9] dir 4
Case cible : 8 8
Conflit de personnages
 -> Hunter [9 9] dir 1

Personnage C :
Hunter [8 8] dir 7
Case cible : 9 8
Best dir  4
 -> Hunter [9 8] dir 4
```

Le jeu en mode console est à rendre pour : **date à définir.**

Partie 2 : jeu en mode graphique

Coder une version graphique du jeu.
Le modèle sera une reprise du jeu en mode console, la vue comprendra une fenêtre graphique principale. Un bouton « Tour suivant » permettra de faire du pas à pas.
Le jeu en mode graphique est à rendre pour : **date à définir.**

Exemple d'affichage de la fenêtre principale alu démarrage du jeu :

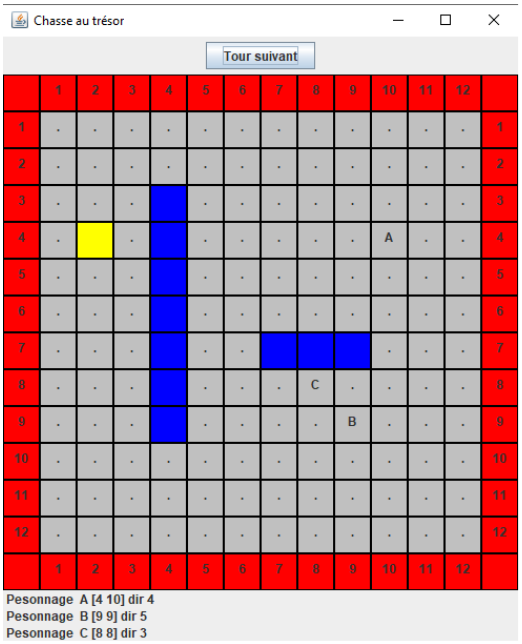

Fig. 4 : *le jeu au démarrage d'une partie.*

Le damier est initialisé de la même façon que celui présenté Fig.3 en mode console.
La fenêtre principale comprend trois panels :

- Panel en haut pour les commandes. Ici, seulement le bouton « Tour suivant »
- Panel au centre : le damier avec les numéros de ligne et de colonne, les bords en rouge, les murs en bleu et le trésor en jaune.
- Panel en bas qui indique des informations sur le « mouvement » de chaque personnage. Soit, une ligne par personnage.

Puis l'utilisateur appuie sur le bouton « Tour suivant ». El la fenêtre principale montre par exemple :

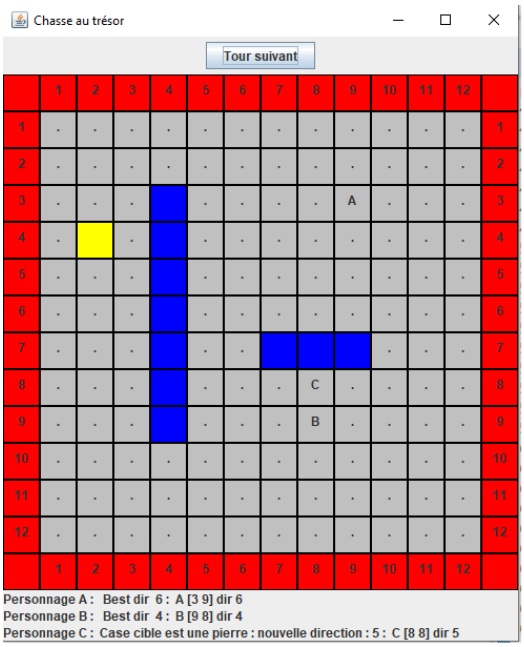

Fig. 5 : *Le jeu à l'issue du premier tour.*

A l'issue des tours la partie peut se terminer ainsi :

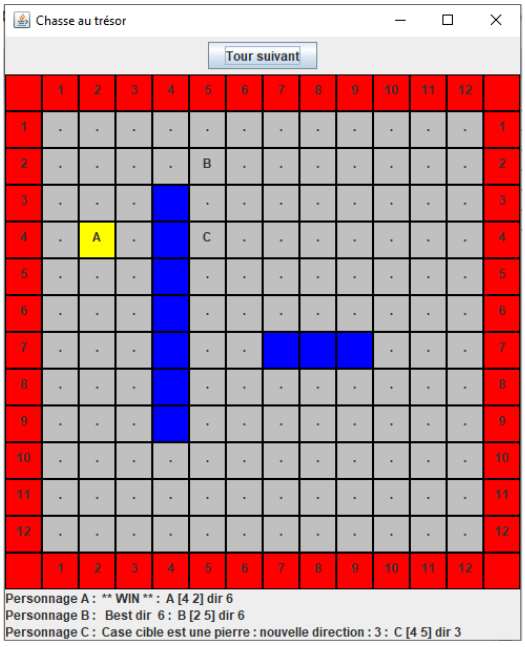

Fig. 6 : *le jeu en fin de partie*

Votre application devra donner des affichages conformes aux Fig. 4,5,6.

Bonus sur la version graphique : (2 points maximum)

Voici quelques idées de bonus pour la version graphique :
Initialisation du damier : ajouter une fenêtre qui permet par le jeu d'interactions de placer les éléments du jeu (murs, trésor, personnages…) avant de commencer une partie.
Introduire d'autres types de cases : des miroirs cassés qui renvoient dans la direction symétrique, des trappes qui téléportent sur une case(libre) aléatoire, des trous qui font passer des tours au personnage avant d'en ressortir.

Si vous avez des idées de bonus, *faites-en part à votre chargé de TP pour en discuter*.