Ce projet est noté, et à réaliser sur les dernières séances de TP en complément du temps de travail personnel. **Le projet sera réalisé en binôme**. Le code doit être en java 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 d'abord en version console, puis en version graphique, ces deux versions faisant l'objet de rendus successifs. Le développement suivra le concept MVC (Modèle, Vue, Contrôleur), avec la version console constituant le modèle, complété par une vue et un contrôleur dans la version graphique.

Un jeu de chasse au trésor

I. Le jeu

1/ But du jeu

Le but du jeu est de faire se déplacer plusieurs personnages (les chasseurs de trésor, *a minima*) sur une grille, tous à la recherche du trésor qui se trouve dans une case de la grille inconnue des personnages. La grille est délimitée par des bords, et comprend également des murs intérieurs qui font obstacle aux déplacements des personnages, ainsi que des cases spéciales pouvant ralentir les personnages, ou leur fournir les outils pour franchir les murs sans les contourner, etc. Chaque personnage se déplace tout seul, c'est à dire sans intervention de l'utilisateur, mais son parcours est conditionné par les cases qu'il rencontre.

Un personnage se déplace d'une case à la fois dans l'une des huit directions possibles autour de sa propre case ('P' dans la figure 1 ci-dessous). Initialement, la direction d'un personnage est choisie au hasard.

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

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

Si la case visée est libre alors le personnage s'y déplace. Si elle est occupée, alors il doit interagir avec l'objet qui l'occupe avant de pouvoir réaliser son déplacement.

Un tour du jeu consiste à faire se déplacer un à un chacun (quand cela leur est possible) des personnages. Le jeu se termine quand l'un des chasseurs a trouvé le trésor.

2/ Les différents types de pièces

Les pièces peuvent être soit mobiles (personnages), soit fixes (obstacle, cases spéciales).

Les pièces mobiles sont *a minima* les chasseurs (Hunter), mais on peut aussi prévoir des autres personnages, par exemple un sage (Wise) qui réoriente les chasseurs vers le trésor, et/ou un menteur

(Cheater) qui éloigne les chasseurs du trésor (ou simplement les désoriente au hasard) et les ralentit (i.e. le chasseur ne se déplacera pas au tour suivant).

Les pièces fixes sont :

- le trésor (Treasure)
- des cartes (RoadMap) qui orientent les chasseurs en direction du trésor
- les bords (Border) de la grille, sur lesquels les personnages rebondissent
- des pierres (Stone) organisées en murs : un mur (Wall) est un ensemble contigu de pierres
- des cases 'Glue' qui ralentissent le personnage (il ne se déplacera à nouveau qu'au bout de deux tours)
- des cases outils 'Tool' (par exemple une échelle) qui dotent le personnage d'un moyen de passer au-dessus des murs sans les contourner
- d'autres types de pièces sont possible, laissées à votre imagination…

Quand un personnage rencontre un mur, s'il est équipé d'un outil il peut se superposer à la case pierre, sinon il devra contourner le mur : une case pierre réoriente alors le personnage vers le bord du mur le plus proche.

3/ Déplacement d'un personnage

Un personnage se déplace en visant une case cible, déterminée par sa position actuelle et sa direction. Si la position de la case cible est libre, alors le personnage s'y déplace. Sinon, la case cible agit sur le personnage, dont le comportement est modifié.

Les différents cas sont les suivants.

- Il y'a un chasseur sur la case cible : le personnage ne l'occupe pas, il est réorienté au hasard.
- La case cible est une case 'Glue' : le personnage peut l'occuper, mais il y restera ensuite bloqué pendant un tour.
- La case cible est un bord : le personnage ne l'occupe pas mais rebondit dans la direction symétrique à la sienne. N.B. A vous de choisir, pour la fluidité, s'il tente de se déplacer immédiatement sur sa nouvelle case cible ou s'il attendra pour cela le tour suivant.
- La case cible est une pierre d'un mur : il y'a deux cas à considérer selon que le personnage dispose d'un outil (une échelle) pour passer sur le mur ou pas.
    1 *pas d'outil*. Le personnage ne l'occupe pas mais se met à le longer le mur en direction du bord le plus proche. Quand il aura fini de longer le mur, et s'il n'a pas été réorienté entre-temps, le personnage reprendra la direction qu'il avait avant de heurter le mur.
    2 *outil* : le personnage se rend sur la case pierre. Il pourra continuer à se déplacer sans changer de direction. Il conserve son outil tant qu'il est sur le mur mais il le perdra dès qu'il l'aura quitté.
- La case cible est une case carte : le personnage peut l'occuper, mais sa direction change : il se dirigera ensuite vers le trésor.
- La case cible est une case outil : le personnage peut l'occuper, et il se retrouve doté d'un outil. N.B. La case outil reste une case outil après cela. Ainsi, si un autre personnage s'y rend ensuite, il pourra à nouveau bénéficier d'un outil.
- La case cibe est la case trésor : le personnage peut l'occuper. Si c'est un chasseur, il a gagné, la partie s'arrête.

Si vous avez prévu d'autres types de personnages, tels que sage ou menteur, à vous de définir les comportements correspondants.

N.B. Quand un personnage ne peut pas occuper sa case cible, à vous de choisir, pour la fluidité, s'il tente de se déplacer immédiatement sur sa case cible recalculée ou s'il attendra le tour suivant.

4/ Superpositions

On le voit, il est possible qu'un personnage se superpose à une case fixe. Par contre deux personnages ne devraient pas en principe se superposer. En cas de superposition dans la case cible, il faudra potentiellement prévoir d'interagir avec tous les occupants (exemple : pierre d'un mur surmontée d'un personnage). Les règles et les priorités d'action à appliquer sont laissées à votre appréciation.

5/ Contrainte concernant les murs

Les murs sont horizontaux ou verticaux, constitués de cases de type *Stone*. 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.

II. Indications et contraintes de codage

Ce projet sera à rendre en deux versions successives. La première version fonctionnera en mode console. La seconde fonctionnera en mode graphique avec une architecture MVC où la version console servira de modèle.

Les personnages à déplacer sont stockés dans une liste de personnages mobiles. Chaque personnage a une direction initialisée au hasard.

La grille n'est pas définie comme une matrice de cases. A la place, on impose que la grille « mappe » les positions occupées à leur contenu. Les positions libres ne sont pas « mappées ». Ainsi, le contenu de la grille doit être de type

```
Map<Position, List<Occupant>>
```

où :

- `Occupant` serait le type de tous les éléments, mobiles ou pas, pouvant occuper le jeu ;
- `Position` serait le type définissant une position par un numéro de ligne et un numéro de colonne.

N.B. Même si on ne peut pas en principe superposer plus de deux occupants, le type `List` permet de ne pas perdre en généralité, en laissant la porte ouverte à de futures extensions.

Pour le mode console, chaque occupant possède son propre affichage sous la forme d'un caractère (par exemple "#" pour une pierre d'un mur, "$" pour le trésor, "~" pour la glue, etc.) Les chasseurs sont symbolisés par une lettre majuscule **différente pour chacun d'eux**. N.B. En pratique cela limite le nombre de chasseurs à 26 mais ce n'est pas un problème.

Les occupants sont organisés par héritage, leur super-classe commune étant abstraite.

Les cases cibles **doivent pouvoir agir sur les personnages** qui tentent de s'y déplacer. Ce comportement sera obtenu en faisant **obligatoirement** implanter à chaque occupant 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
}
```

N.B. Si vous avez d'autres personnages en mouvement que les chasseurs, vous devrez certainement adapter le type du paramètre de la méthode `process`, pour ne pas le limiter à `Hunter`.

Ainsi, pour un occupant dont le comportement serait de réorienter au hasard le personnage qui tente de s'y rendre, la méthode `process` va affecter à `h` une nouvelle direction aléatoire. Si le comportement est de ralentir le personnage, alors la méthode va par exemple modifier le nombre de tours d'attente de `h`, etc.

Il est possible, mais ce n'est pas imposé, de définir d'autres interfaces. Par exemple, les personnages mobiles pourraient implanter une interface `Moveable` définissant des méthodes permettant de réaliser les mouvements (calcul de la case cible, déplacement effectif, etc.)

Si besoin, le calcul de la distance entre deux cases 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 $4^2 + 2^2 = 20$.

	X				
					X

III. Travail demandé

Les programmes doivent être codés en Java avec l'IDE Eclipse. Attention à régler dans les réglages du projet la compatibilité Java 1.8. Une mise en œuvre élégante des concepts objet étudiés au cours du semestre sera appréciée.

Les deux versions du projet seront à rendre successivement sur Moodle, qui précisera les dates des rendus. Un seul dépôt par binôme devra être réalisé pour chaque rendu. Les extensions potentielles au sujet sont à faire valider par votre encadrant de TP avant de les inclure dans vos développements.

Rendu 1 : jeu en mode console

Vous rendrez d'abord le futur modèle de votre architecture MVC sous la forme d'un projet Eclipse correspondant au jeu en mode console, accompagné de son analyse : diagramme de classe complet de tous les objets que vous avez développés, et décrivant les choix d'implantation retenus pour coder le projet.

Rendu 2 : jeu en mode graphique

Vous déposerez dans un deuxième temps la version graphique du jeu, en architecture MVC. Le modèle sera une reprise du jeu en mode console. Un bouton « tour suivant » permettra de faire du pas à pas. Ce rendu sera accompagné d'un rapport dont le contenu attendu sera indiqué sur Moodle.