Diplôme : Licence SPI 3ème année	2022-2023

UE : Ateliers de programmation

Programmation Orientée Objet

Atelier 4 : Bilan

- classes attributs méthodes
- attributs et méthodes statiques
- hiérarchie d'héritage
- classes abstraites, polymorphisme

Enseignants : Paul-Antoine BISGAMBIGLIA, Marie-Laure NIVET,
Evelyne VITTORI, David ARAUJO, Rodolpho BABATOUNDE

Jeu d'aventures

On souhaite développer un simulateur très simplifié de jeu d'aventures.
Le jeu comporte des personnages qui se déplacent dans les cases d'un tableau à une dimension.
Chaque personnage appartient à un joueur et lui procure des points ou des pénalités (diminution de points) lors de ses déplacements.
Le jeu comporte deux types de personnages, les taurens et les humains qui se déplacent à des rythmes différents.
Le lancement du jeu consiste à déplacer successivement chacun des personnages. Ce déplacement est répété autant de fois que le jeu comporte d'étapes.
Le jeu comporte 50 cases numérotées de 0 à 49. Lors du lancement du jeu, le tableau des cases est initialisé. Un gain (nombre de points gagnés) est ainsi attribué à chaque case et des obstacles sont placés sur certaines cases. Un obstacle est caractérisé par une pénalité (nombre de points à soustraire).
Lorsque l'on demande à un personnage de se déplacer, il détermine la case sur laquelle il souhaite aller. Si la case est vide, le personnage s'y place et cela procure une augmentation de points à son joueur propriétaire (égale au montant du gain associé à la case). Si la case comporte un obstacle, le personnage ne se déplace pas et cela entraine une diminution de points à son joueur propriétaire (égale au montant de la pénalité associée à l'obstacle). Enfin, si la case souhaitée est déjà occupée par un autre personnage, le personnage ne se déplace pas et cela entraine une diminution de points à son joueur propriétaire (égale au montant du gain associé à la case).

A la fin du jeu, la liste des joueurs est affichée avec leur nombre de points respectifs et le (ou les) gagnants sont identifiés (joueurs ayant obtenu le nombre maximum de points).
Le logiciel doit permettre de créer un jeu, de créer des joueurs, de créer des personnages appartenant à des joueurs, d'inscrire des joueurs au jeu et de lancer le jeu.

Pour chacune des questions, vous prendrez soin d'écrire un programme de test de chaque classe sans attendre la question finale.

Remarque importante : Pour chacune des classes des différentes questions, les méthodes get et set (getters et setters) à définir ne sont pas obligatoirement mentionnées dans les indications données. Vous devrez les ajouter si elles sont nécessaires et non de manière systématique.

Question 1 : Classe Obstacle

Définissez la programmation Java d'une classe Obstacle comportant un seul attribut *pénalité* de type int représentant la penalité qui caractérise l'obstacle c'est-à-dire le nombre de points à retrancher au joueur qui le rencontre.
Le constructeur de cette classe possède un paramètre de type int représentant la valeur de l'attribut pénalité. La classe doit aussi fournir une méthode get permettant d'accéder à l'attribut pénalite.

Question 2 : Classe Joueur

Définissez la programmation Java de la classe Joueur conformément aux indications données ci-dessous :

- La classe possède 5 attributs :
  - un attribut **nom** de type String
  - un attribut **code** de type String de la forme J suivi d'un numéro correspondant au numéro d'ordre de création du joueur (par exemple : J1 pour le premier joueur, J2 , ..ect…). Il est généré automatiquement lors de la création d'un joueur.
  - un attribut **nbJoueurs** de type int représentant le nombre de joueurs créés.
  - un attribut **nbPoints** est un entier **positif** qui représente le nombre de points gagnés par le joueur au cours du jeu. Lors de la création du joueur, cet attribut est initialisé à 0.
  - un attribut **listePersos** de type ArrayList<Personnage> représentant la liste des personnages appartenant au joueur.
- Le constructeur possède un paramètre de type String représentant le nom du joueur.
- La méthode **ajouterPersonnage**(p : Personnage) ajoute le personnage p à la liste des personnages du joueur.
- La méthode **modifierPoints**(nb : int) ajoute la valeur du paramètre nb au nombre de points du joueur (*remarque : nb peut être un nombre négatif*).
- La méthode **peutJouer()** renvoie la valeur vrai si le joueur possède au moins un personnage.
- La méthode **toString**() doit avoir pour résultat une chaîne de la forme suivante :

```
J1 Paul(15 points) avec 2 personnages
```

Ou, dans le cas où le joueur n'a encore aucun personnage :

```
J1 Paul (0 point) aucun personnage
```

Question 3 : Classe Personnage

Définissez la programmation Java de la classe **abstraite** Personnage conformément aux indications données suivantes :

- La classe comporte quatre attributs :
  - Un attribut **nom** de type String
  - Un attribut **age** de type int
  - Un attribut **position** est un entier qui représente la position (numéro de case) du personnage dans le tableau des cases du jeu.
  - Un attribut **proprietaire** de type Joueur représentant le propriétaire du personnage.
- Le constructeur possède deux paramètres nom de type String et age de type int.
- La méthode **deplacer**(destination : int , gain : int) modifie la position du personnage en lui attribuant la valeur du paramètre destination et augmente le nombre de points de son joueur propriétaire de la valeur du paramètre gain.
- La méthode **penaliser**(penalite : int) diminue le nombre de points de son joueur propriétaire de la valeur du paramètre penalite.
- La méthode **toString**() renvoie simplement le nom du personnage.
- La méthode **positionSouhaitee()** est abstraite et renvoie un int.

MCF informatique : Marie-Laure Nivet - ☎ : +33 (0)4 95 45 02 09 - 🖷 : +33 (0)4 95 61 05 51 - 🖃: marie-laure.nivet@univ-corse.fr
Faculté des Sciences et Techniques - Campus Grossetti - BP 52 - 20250 Corte
http://fst.univ-corse.fr

Question 4 : Classe Tauren

Définissez la programmation Java de la classe Tauren qui **hérite de la classe personnage** conformément aux indications suivantes :

- L'attribut **taille** représente la taille du Tauren exprimée en mètres.
- Le constructeur possède trois paramètres nom de type String, age de type int et taille de type int.
- La méthode **positionSouhaitee**() renvoie un entier représentant la position que souhaite atteindre le personnage. Cette position est calculée en ajoutant à la position actuelle du personnage un nombre aléatoire compris entre 1 et la valeur de l'attribut taille.

  Pour effectuer ce calcul, vous pourrez utiliser la méthode statique *random* de la classe Math :

  `public static double random()` qui renvoie un nombre réel x tel que $0 \leq x < 1$.
- La méthode **toString**() doit avoir pour résultat une chaîne de la forme suivante :

  `Tauren Hector`

Question 5 : Classe Humain

Définissez la programmation Java de la classe Humain qui **hérite de la classe personnage** conformément aux indications suivantes :

- L'attribut **nbDeplacements** représente le nombre de déplacements effectués par l'humain. Il est initialisé à 0 lors de la création de l'humain et incrémenté dans la méthode deplacer.
- L'attribut **niveau** représente le niveau de l'humain qui est compris entre 1 et 3. Il est initialisé à 1 lors de la création de l'humain, il prend la valeur 2 lorsque le nombre de déplacements arrive 4 et il prend la valeur 3 lorsque le nombre de déplacements arrive à 6 (modification dans la méthode deplacer).
- Le constructeur possède deux paramètres: nom de type String et age de type int.
- La méthode **deplacer**(destination : int , gain : int) redéfinit la méthode deplacer de Personnage en la complétant pour mettre à jour les attributs niveau et nbDeplacements.
- La méthode **positionSouhaitee**() renvoie un entier représentant la position que souhaite atteindre le personnage. Cette position est calculée en ajoutant à la position actuelle du personnage, le niveau multiplié par le nombre de déplacements.
- La méthode **toString**() doit avoir pour résultat une chaîne de la forme suivante :

  `Humain Jean`

Question 6 : Classe Case

Définissez la programmation Java de la classe Case conformément aux indications données ci-dessous.

- La classe comporte trois attributs :
  - Un attribut **gain** correspondant au nombre de points à ajouter au joueur propriétaire du personnage lorsque celui-ci se place sur la case. Ce nombre représente également le nombre de points à retrancher au joueur si son personnage tente de se placer sur la case alors qu'elle est déjà occupée par un autre personnage.
  - Un attribut **perso** de type Personnage représentant le personnage présent sur la case. Cet attribut a la valeur null si aucun personnage n'est présent sur la case.
  - Un attribut **obs** de type Obstacle représentant l'obstacle présent sur la case. Cet attribut a la valeur null si aucun obstacle n'est présent sur la case.
- La classe possède deux constructeurs:
  - un constructeur à deux paramètres: obs de type Obstacle et gain de type int,
  - un constructeur à un paramètre: gain de type int qui permet de créer une case ne comportant pas d'obstacle.
- La méthode **getPenalite()** renvoie 0 si la case ne comporte pas d'obstacle et si la case comporte un obstacle, elle renvoie la penalité caractérisant l'obstacle considéré.

- Les méthodes **placerPersonnage**(perso : Personnage) et **placerObstacle**(obs : Obstacle) positionnent respectivement le personnage perso et l'obstacle obs sur la case.
- La méthode **enleverPersonnage**() affecte la valeur null au personnage perso associé à la case.
- La méthode **estLibre**() renvoie un booléen égal à vrai si la case ne comporte ni obstacle ni personnage
- Les méthodes **sansObstacle()** et **sansPerso**() renvoient un booléen indiquant respectivement l'absence d'obstacle et l'absence de personnage sur la case.
- La méthode **toString**() doit avoir pour résultat une chaîne de la forme suivante :
  `Libre (gain = 28)` dans le cas où la case est libre
  `Obstacle (penalité = -30)` dans le cas où la case est occupée par un obstacle
  `Humain Jean(penalité = -34)` ou `Tauren Hercule(penalité = -34)` dans le cas où la case est occupée par un personnage (Tauren ou Humain) (la pénalité correspond dans ce cas au montant du gain de la case transformé en nombre négatif)

Question 7 : Classe Jeu

Définissez la programmation Java de la classe Jeu conformément aux indications données ci-dessous:

- La classe comporte 7 attributs :
  - un attribut **titre** de type String.
  - une constante **NB_JOUEUR_MAX** égale à 6 qui correspond au nombre maximum de joueurs pouvant être inscrits à un jeu.
  - une constante **NB_CASES** égale à 50 qui correspond au nombre de cases du tableau sur lequel se déplacent les personnages.
  - Un attribut **listeJoueurs** de type ArrayList<Joueur> représentant la liste des joueurs inscrits au jeu.
  - Un attribut **cases** de type tableau de Case représentant la liste des cases du jeu.
  - Un attribut **nbEtapes** de type int correspondant au nombre de déplacements à réaliser par chacun des personnages au cours du déroulement du jeu.
  - Un attribut **nbObstacles** de type int correspondant au nombre maximum d'obstacles présents dans le tableau des cases (il s'agit d'un nombre maximum car comme les obstacles sont générés de manière aléatoire, il n'est pas certain que ce nombre soit atteint lors de l'initialisation des cases).
  - Un attribut **scoreMax** de type int représentant le score maximum obtenu sur l'ensemble des lancements de jeux.
- Le constructeur possède trois paramètres: String titre de type String, nbEtapes de type int, nbObstacles de type int.
- La méthode **ajouterJoueur**(j : Joueur) ajoute le joueur j à la liste des joueurs inscrits au jeu.
- La méthode **tousLesPersos**() renvoie un ArrayList de Personnage contenant la liste de tous les personnages de tous les joueurs. Cet arrayList est construit en parcourant les listes de personnages de chacun des joueurs du jeu.
- La méthode **initialiserCases**() initialise le tableau des cases en attribuant à chaque case un montant de gains défini par un nombre aléatoire compris entre 1 et le nombre de cases du tableau. Un obstacle est également créé chaque fois que le nombre aléatoire généré est un multiple de 5 (vous utiliserez l'opérateur modulo (reste de la division entière) qui est exprimé en java par l'opérateur % : 10%3=1 par exemple). Chaque obstacle est créé avec une pénalité égale au double du nombre aléatoire généré et il est placé dans la case considérée. Le nombre effectif d'obstacles ne devra toutefois pas dépasser le nombre maximum d'obstacles attribué au jeu (attribut nbObstacles).

- La méthode **lancerJeu()** commence par positionner l'ensemble des personnages (renvoyé par la méthode **tousLesPersos**) dans le tableau des cases. Ainsi, le premier personnage est placé sur la case 0 si celle-ci ne comporte pas d'obstacle, si elle comporte un obstacle, on le place dans la case 1, ect… On passe ensuite au personnage suivant et ainsi de suite.

Le jeu démarre alors.

Lors de chaque étape, chacun des personnages tente de se déplacer. Si la position souhaitée par le personnage correspond à une case libre, le personnage est effectivement déplacé et le joueur associé gagne des points. Si la position souhaitée correspond à une case avec obstacle, le personnage n'est pas déplacé et le joueur associé se voit attribuer une pénalité. De même si la case est déjà occupée par un autre personnage, le joueur associé se voit attribuer une pénalité égale au montant des gains associé à la case. Si la position souhaitée dépasse le numéro de la dernière case du tableau, on considère que le joueur souhaite atteindre la dernière case du tableau.

La méthode se termine lorsque toutes les étapes se sont déroulées. Elle affiche alors les résultats.

- La méthode **afficherCases()** affiche le tableau des cases sous la forme suivante

```
Case 0 : Humain Jean (penalité = -7)
Case 1 : Obstacle (penalité = -70)
Case 2 : Obstacle (penalité = -20)
Case 3 : Libre (gain = 37)
Case 4 : Libre (gain = 31)
Case 5 : Libre (gain = 37)
Case 6 : Libre (gain = 33)
Case 7 : Libre (gain = 49)
Case 8 : Tauren Hercule (penalité = -43)
...
```

- La méthode **afficherParticipants()** affiche la liste des joueurs inscrits au jeu sous la forme suivante

```
LISTE DES JOUEURS
-------------------------
J1 Paul(253 points) avec 2 personnages
-------------------------
J2 Lucien(240 points) avec 2 personnages
```

- La méthode **afficherResultats()** affiche la liste des joueurs inscrits au jeu en invoquant la méthode **afficherParticipants**() puis affiche les résultats du jeu sous la forme suivante :

```
JEU AtelierPOO
*********************************************
RESULTATS
Le gagnant est Paul avec 253 points
Record battu : Ancien score maximum 134
```

Question 8 : Classe TestJeu

Définissez la programmation Java de la classe **TestJeu** comportant une méthode main réalisant la répétition (5 fois) les actions suivantes :

- Créer un jeu ayant pour titre *AtelierPOO* comportant 4 étapes et 10 obstacles maximum.
- Créer un joueur nommé Paul et lui attribuer deux personnages :
  - un tauren de nom Hector agé de 15 ans et de taille 10
  - un humain nommé Jean agé de 10 ans.
- Créer un joueur nommé Lucien et lui attribuer deux personnages :
  - un humain de nom Marie agé de 10 ans
  - un tauren nommé Hercule agé de 20 ans et de taille 5.
- Inscrire les deux joueurs au jeu AtelierPOO

- Lancer le jeu
- Afficher la liste des participants et le résultat comportant l'affichage du score max.

MCF informatique : Marie-Laure Nivet - ☎ : +33 (0)4 95 45 02 09 - 🖷 : +33 (0)4 95 61 05 51 - 🖃: marie-laure.nivet@univ-corse.fr
Faculté des Sciences et Techniques - Campus Grossetti - BP 52 - 20250 Corte
http://fst.univ-corse.fr