	Université de Corse - Pasquale PAOLI
	Diplôme : MASTER1 Informatique DFS et DE — ***2024-2025***
	Cours Design Patterns
	TD N°1 : Design Pattern Strategy
	Enseignant : Evelyne VITTORI

Exercice 1 - Football

Dans le cadre d'un jeu de football, on s'intéresse à la modélisation du jeu des joueurs implémenté par la méthode jouer() de la classe Joueur (cf. figure 1).

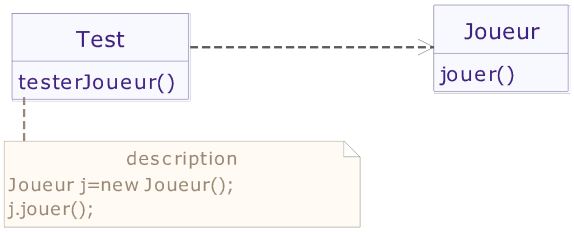

Figure 1 : Diagramme de classe TestJoueur

L'algorithme de cette méthode va varier en fonction de la stratégie de jeu attribuée au joueur. On considère en effet que plusieurs stratégies de jeu sont envisageables : stratégie de défense et stratégie d'attaque. Lors de sa création, un joueur possèdera par défaut une stratégie de défense mais la modélisation doit permettre de modifier cette stratégie en lui affectant par exemple une stratégie d'attaque au cours du déroulement du jeu.
La conception doit permettre d'envisager l'ajout de nouvelles stratégies de jeu sans avoir à modifier la classe Joueur.

1. Proposez un diagramme de classe prenant en compte ces différentes stratégies de jeu en utilisant le **pattern Strategy**
2. Implémentez votre solution en Java (ou un autre langage OO de votre choix)
3. Définissez un diagramme de séquence illustrant votre solution en décrivant le scénario de test suivant (méthode testerJoueur) :
   - Créer un joueur
   - Le faire jouer
   - Modifier sa stratégie en lui attribuant une stratégie d'attaque
   - Le faire jouer
4. On suppose à présent que le cahier des charges du jeu a changé. D'une part, le jeu est une version définitive, nous sommes certains qu'aucune nouvelle stratégie ne devra être ajoutée. Et d'autre part, la stratégie attribuée aux joueurs lors de leur création doit être immuable et elle ne doit pas pouvoir être modifiée au cours du jeu.

Dans ce cas, expliquez pourquoi l'utilisation du pattern Strategy n'est plus indispensable et proposez une solution plus simple.

Exercice2 - Jeu d'aventures

On souhaite modéliser le fonctionnement d'un jeu d'aventures comportant différents types de personnages : des humains, des trolls, des orcs et des taurens.
Les personnages savent se battre en utilisant des armes qui peuvent être des dagues, des épées, des boucliers et des bâtons.
A un moment donné du jeu, un personnage ne manipule qu'une seule arme. Lors de sa création, une arme par défaut lui est attribuée en fonction de son type (Orcs : baton ; Tauren : bouclier, Troll : dague ; Humain : épée). Par la suite, il peut changer d'arme plusieurs fois au cours du déroulement du jeu.
L'utilisateur du jeu doit avoir la possibilité d'ordonner à un personnage de se battre. Selon l'arme manipulée par le personnage lorsqu'il reçoit cet ordre, un comportement spécifique sera déclenché.
Une première ébauche de programme Java a été réalisée pour implémenter ce jeu. Récupérez les classes Personnage.java et Jeu.java sur l'ENT, et importez les dans un environnement de programmation de votre choix (Eclipse, NetBeans, IntelliJ,...) afin de tester leur fonctionnement.

Question 1.1 : On souhaite modifier le programme existant afin d'ajouter un nouveau type de personnage, les « Monstres » et un nouveau type d'arme, les « Couteaux ».

- **a.** Modifiez la classe Personnage afin de prendre en compte ces changements.
- **b.** En vous appuyant sur les modifications réalisées, identifiez les inconvénients de la conception de la classe Personnage.

Question 1.2 : Définissez un diagramme de classe permettant de concevoir l'application de manière plus satisfaisante en définissant d'une part une hiérarchie d'héritage au niveau des types de personnages et en appliquant le pattern Strategy (cf. cours) pour modéliser l'utilisation des armes.

Question 1.3 : Expliquez en quoi votre solution permet de palier les inconvénients identifiés en 1.1.a.

Question 1.4 : Implémentez en java votre nouvelle conception.

Question 1.5 : On souhaite à présent implémenter le déplacement des personnages. Les Taurens, les Trolls et les Orcs ne se déplacent qu'en courant alors que les Humains se contentent de marcher. Cependant s'ils n'ont pas de chaussures, les humains ne peuvent pas se déplacer.

- **a.** Complétez votre diagramme de classe de la question 1.2 afin de prendre en compte cette nouvelle fonctionnalité en utilisant à nouveau le pattern Strategy.
- **b.** Expliquez l'intérêt de l'utilisation du pattern Strategy dans cette situation.
- **c.** Implémentez votre solution en Java et testez la.

Question 1.6 : Définissez un diagramme de séquence modélisant l'exécution du scénario suivant :

- L'utilisateur crée un personnage nommé « diablo » de type « Tauren ».
- L'utilisateur demande au personnage de se battre.
- L'utilisateur change l'arme de diablo et lui confie une épée.
- L'utilisateur demande à diablo de se battre.
- L'utilisateur demande à diablo de se déplacer.