TP2 - Héritage et Polymorphisme

Contraintes et Remise du TP

- ➢ La correction tiendra compte de la brièveté des fonctions que vous écrivez (évitez les fonctions de plus de 25 lignes) - n'hésitez pas à découper une fonction en plusieurs sous-fonctions plus courtes.
- ➢ Vos classes et méthodes doivent être documentées
- ➢ Pensez à utiliser pylint pour avoir un code qui soit le plus qualitatif possible.
- ➢ Les noms de classe commencent par une majuscule.
- ➢ Les noms de méthodes et d'attributs commencent par une minuscule.
- ➢ Les TP sont à réalisés en binômes, et à rendre au travers d'un dépôt **git.** Vous avez 4h pour faire les TP, et le commit pris en compte sera le dernier réalisé avant minuit.

Contexte

Vous allez développer un jeu de cartes simple où chaque carte possède des comportements spécifiques. Le but du TP est de mettre en œuvre les concepts d'héritage et de polymorphisme en créant des classes de cartes avec des effets différents. Le jeu se déroule dans un terminal, avec une interaction texte.

Objectifs pédagogiques

1. Comprendre l'héritage et sa mise en place en POO (Programmation Orientée Objet).
2. Utiliser le polymorphisme pour redéfinir des méthodes dans des classes dérivées.
3. Travailler avec des classes abstraites ou interfaces pour structurer les comportements communs.
4. Mettre en place une interaction simple via le terminal.

Règles du jeu

Le jeu se déroule en plusieurs tours, où chaque joueur tire une carte d'un deck. Chaque type de carte a un effet particulier qui modifie le score du joueur. Le but est d'obtenir le score le plus élevé après un nombre défini de tours.

Types de cartes

Vous devez implémenter plusieurs types de cartes, chacune héritant d'une classe abstraite commune **Carte**. Voici les types de cartes à créer :

1. **Carte Normale** : Donne un nombre de points aléatoire compris entre 1 et 10.

2. **Carte Bonus** : Double le score du joueur pour ce tour.
3. **Carte Malus** : Réduit le score du joueur de 5 points.
4. **Carte Chance** : Ajoute un nombre de points aléatoire entre -5 et 15.

A vous de décider de l'architecture, tout en évitant de dupliquer du code.

Par la suite vous créerez également une classe Joueur qui possède un nom et un score. Le joueur devra également posséder une méthode **jouerCarte** qui prend en paramètre une carte et applique son effet sur le score du joueur.

.

Interaction via le terminal

- Créez un deck contenant des instances de différentes cartes. (56 cartes - 30 cartes Normales, 6 cartes Bonus, 5 cartes Malus, et 15 cartes Chances)
- À chaque tour, un joueur tire une carte aléatoirement du deck.
- Affichez le type de la carte tirée, son effet, et le score du joueur après chaque tour.
- Après un nombre défini de tours (par exemple 5), affichez le score final.

Voici la sortie attendue:

```
Tour 1 : Joueur 1 tire une Carte Chance.
Effet : Ajoute 10 points.
Score actuel de Joueur 1 : 10

Tour 2 : Joueur 1 tire une Carte Malus.
Effet : Perd 5 points.
Score actuel de Joueur 1 : 5

...

Score final de Joueur 1 : 25
```

Bonus

Implémentez une classe tricheur qui modifiera le comportement de **jouerCarte**, en faisant en sorte que le joueur ne perde pas des points à chaque carte malus qu'il tire.