ALGO2 – Algorithmique et Programmation 2

Fiche de TP numéro 2

Exercice 1 : Définissez une classe `CompteBancaire()`, qui permette d'instancier des objets tels que compte1, compte2, etc. Le constructeur de cette classe initialisera deux attributs *__nom* et *__solde*. Par défaut, le solde sera initialisé à 1000 euros.

Trois autres méthodes seront définies :

— `depot(somme)` permettra d'ajouter une certaine somme au solde.

— `retrait(somme)` permettra de retirer une certaine somme du solde. Un retrait supérieur au solde du compte ne sera pas effectué.

— `affiche()` permettra d'afficher le nom du titulaire et le solde de son compte.

Écrivez un programme principal qui vous permet de tester votre classe.

```
>>> compte1 = CompteBancaire('Michu', 800)
>>> compte1.depot(350)
>>> compte1.retrait(200)
>>> compte1.affiche()
Le solde du compte bancaire de Michu est de 950 euros.
>>> compte2 = CompteBancaire('Blanchard')
>>> compte2.depot(25)
>>> compte2.affiche()
Le solde du compte bancaire de Blanchard est de 1025 euros.
```

Exercice 2 : Définissez une classe `Voiture` qui permet de créer des objets représentants des voitures. Une `Voiture` est caractérisée par sa couleur, son nom et sa vitesse. Par défaut, à la création d'une `Voiture`, on doit donner un nom et une couleur, qui permettront de donner des valeurs aux attributs `__nom` et `__couleur`, mais la `__vitesse` est initialisée à zéro.

Quelques autres méthodes seront définies :

— la méthode `accelere(inc)` qui augmente de `inc` la vitesse de la voiture. Nos voitures sont limitées à 130 km/h, et ne peuvent pas être accélérées de plus de 10km/heure (pour augmenter de 20km/h, il faudra donc appeler deux fois la méthode `accelere()`). C'est à `accelere()` de contrôler tout cela.

— la méthode `freine(dec)` qui diminue de `dec` la vitesse de la voiture. Une voiture ne peut pas rouler à une vitesse négative (elle est à l'arrêt lorsque sa vitesse est à zéro).

— la méthode `str()` qui retourne une chaîne de caractères avec toutes les indications utiles sur la voiture (quel est son nom ? quelle est sa couleur ? est-elle à l'arrêt ? sinon quelle est sa vitesse ?).

— la méthode `affiche()` qui affiche sur la console toutes les indications utiles sur la voiture.

Travail à faire :

— vous devez écrire et spécifier chacune des méthodes demandées

— vous devez écrire des tests unitaires pour chacune des méthodes

— vous devez écrire un script principal qui illustre l'utilisation de chacune des méthodes.

Écrivez un programme principal qui vous permet de tester votre classe.

Exercice 3 : Les cartes (à la belote)

Le but de cet exercie est de construire des cartes pour un jeu de belote.

Q 1 . Nous nous intéressons à l'implémentation de la classe `Carte` qui modélise une carte à jouer. Une carte est définie par sa hauteur (7, 8, 9, Valet, Dame, Roi, ...), sa couleur (trèfle, carreau, coeur, pique), sa valeur et son nombre de points. La valeur d'une carte est donnée par sa place dans la hiérarchie des cartes. À la belote, hors atout, les cartes sont ordonnées (dans l'ordre croissant de valeur) par 7, 8, 9, Valet, Dame, Roi, 10, As. Les points des cartes sont respectivement 0, 0, 0, 2, 3, 4, 10, et 11. À l'atout, les cartes sont ordonnées, toujours dans l'ordre croissant par 7, 8, Dame, Roi, 10, As, 9, Valet. Le nombre de points du Valet est 20, le nombre de points du 9 est 14. Tous les autres nombres de points sont les mêmes qu'hors atout.

Proposez un constructeur pour la classe `Carte`. Par défaut, une carte n'est pas dans la couleur d'atout.

Q 2 . Une carte devra posséder les méthodes suivantes :

- une méthode pour afficher une carte
- une méthode qui retourne la couleur de la carte ; une autre pour la hauteur ; une autre pour le nombre de points ;
- une méthode qui indique si la carte est un atout
- une méthode qui teste si une carte est strictement plus grande qu'une autre en valeur (le test utilisera la valeur des cartes, sachant qu'une carte d'atout est plus forte qu'une carte hors atout)
- une méthode pour savoir si une carte est égale en valeur à une autre.

Exemples d'utilisation :

```
>>> c1 = Carte('Trèfle','Roi')
>>> c1.affiche()
Roi de Trèfle
>>> c1.nbrePoints()
4
>>> c1.estAtout()
False
>>> c2 = Carte('Carreau','7',True)
>>> c2.estAtout()
True
>>> c1.est_plus_grand(c2)
False
```

Q 3 . Un jeu de cartes sera représenté dans la suite par une liste de cartes. Spécifiez puis écrivez une fonction `affiche_jeu` qui, pour un jeu de cartes donné en paramètre, affiche la liste des cartes.

```
# TJeuDeCartes = list(Carte)
```

Q 4 . Spécifiez puis écrivez une fonction `creer_jeu_de_belote` qui crée un jeu de cartes pour la belote. L'atout sera choisi aléatoirement parmi les quatre couleurs habituelles.

```
>>> belote = creer_jeu_de_belote()
>>> affiche_jeu(belote)
['7 de Trèfle', '8 de Trèfle', '9 de Trèfle', '10 de Trèfle',
'Valet de Trèfle', 'Dame de Trèfle', 'Roi de Trèfle', 'As de Trèfle',
'7 de Carreau', ... 'Roi de Carreau', 'As de Carreau',
'7 de Coeur', ... 'Roi de Coeur', 'As de Coeur',
'7 de Pique', ... 'Roi de Pique', 'As de Pique']
>>> belote[7].str()
'As de Trèfle'
>>> belote[15].str()
'As de Carreau'
>>> belote[15].est_egal(belote[7])
True
```