LMI 1 - Semestre 2

Université d'Artois
Faculté des Sciences Jean Perrin

ALGO2 – Algorithmique et Programmation 2

Fiche de TD numéro 7

Pour les exercices suivants,
- on utilisera la classe `Pile` créée en cours
- on utilisera la classe `Carte` que certains d'entre vous ont déjà créée en TP ;

On rappelle qu'une instance de la classe `Carte` possède les méthodes suivantes :
- `couleur(self)` : retourne la couleur de la carte
- `hauteur(self)` : retourne la hauteur de la carte
- `est_plus_grande_que(self,c)` : retourne vrai si `self` est plus grande que la carte `c`

Exercice 1 : Nous nous intéressons à l'implémentation de la classe `Carte` qui modélise une carte à jouer. Une carte est définie par sa hauteur (2, 3, ...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. Généralement, les cartes sont ordonnées (dans l'ordre croissant de valeur) par 2, 3, ...7, 8, 9, 10, Valet, Dame, Roi, As.

Une carte devra posséder les méthodes suivantes :
- un constructeur
- une méthode pour afficher une carte
- une méthode qui retourne la couleur de la carte ;
- une méthode qui retourne la hauteur de la carte ;
- une méthode qui teste si une carte est strictement plus grande qu'une autre **uniquement** en fonction de sa hauteur ;

Exemples d'utlisation :

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

Exercice 2 : Spécifiez puis écrivez la fonction `vider_couleur` qui, pour une couleur donnée et deux piles, fait passer les cartes de la première pile à la seconde, si elles sont de la couleur spécifiée. Dès qu'on rencontre une carte d'une couleur différente, on arrête de changer les cartes de pile.

Exercice 3 : Spécifiez puis écrivez la fonction `separer_couleurs` qui prend quatre piles en entrée telles que la première contient toutes les cartes d'un jeu dans un ordre aléatoire, et les trois autres sont vides. À la fin de la fonction, la première pile contient tous les trèfles, la deuxième tous les coeurs, la troisième tous les carreaux, et la dernière tous les piques.

Exercice 4 : Spécifiez puis écrivez la fonction `vider_plus_petits_que` qui, pour une carte donnée et deux piles, fait passer les cartes de la première pile à la seconde, si elles sont de la même couleur que la carte donnée, et de valeur inférieure. Dès qu'on rencontre une carte d'une couleur différente ou d'une valeur supérieure, on arrête de changer les cartes de pile.

Exercice 5 : Spécifiez puis écrivez la fonction `trier` qui, à partir de trois piles passées en paramètre, trie les cartes qui sont sur la première pile. À l'appel de la fonction `trier`, la première pile passée en paramètre contient des cartes qui sont toutes de la même couleur mais dans un ordre aléatoire. Les deux autres piles ne contiennent pas de carte de la même couleur (vous pouvez même considérer qu'elles sont vides, ça ne change rien). À la fin de la fonction, la première pile contient les mêmes cartes qu'au début mais cette fois-ci triées dans l'ordre décroissant (l'as au sommet de la pile, puis le roi, etc etc) ; et les deux autres piles sont dans le même état qu'au départ.

Exercice 6 : Spécifiez puis écrivez la fonction `trier_4_piles` qui, à partir des quatre piles passées en paramètre, trie leurs cartes dans l'ordre décroissant. À l'appel de la fonction `trier_4_piles`, chaque pile contient des cartes d'une seule et même couleur (par exemple, les trèfles sur la première, les coeurs sur la deuxième, etc etc), mais dans un ordre aléatoire, et d'une couleur différente pour chaque pile. À la fin de la fonction, les quatre piles sont triées dans l'ordre décroissant.