EXERCICE 1 (6 points)

Cet exercice porte sur la programmation orientée objet, l'algorithmique et la programmation en Python.

Marc souhaite coder le jeu de cartes "6 qui prend" afin d'y jouer contre l'ordinateur ou en ligne avec ses amis. Il va réaliser le code en Python en utilisant la programmation orientée objet, sans se préoccuper de l'interface graphique.

Ce jeu est composé de 104 cartes numérotées de 1 à 104. Chacune possède une ou plusieurs "têtes de bœuf" (notée TdB) qui représentent des points de pénalité. Le but est d'avoir le moins de "têtes de bœuf" possible à la fin de la partie.

Partie A : La classe `Carte`

On commence par créer la classe `Carte` pour modéliser les 104 cartes.

Le constructeur de cette classe prend en paramètre une valeur (de type `int`) comprise entre 1 et 104 et l'affecte à l'attribut `valeur` (de type `int`).

On ajoute l'attribut `TdB` (de type `int`) qui contient le nombre de "têtes de bœuf" qui sera calculé à l'aide de la méthode de classe `calcul_TdB` (définie à la question suivante) qui n'a pas de paramètre.

1. Écrire le code du constructeur de la classe `Carte`.

Chaque carte possède une ou plusieurs "têtes de bœuf".
Ce nombre est déterminé de la façon suivante :

- si la valeur de la carte est divisible par 11 alors la carte reçoit 5 "têtes de bœuf";
- si la valeur de la carte se termine par 0 alors la carte reçoit 3 "têtes de bœuf";
- si la valeur de la carte se termine par 5 alors la carte reçoit 2 "têtes de bœuf";
- si la valeur de la carte ne remplit aucune de ces conditions alors elle ne reçoit qu'1 "tête de bœuf".

Attention, ces règles se cumulent.

Par exemple, 55 est divisible par 11 et se termine par 5 donc la carte 55 comporte 7 "têtes de bœuf".

2. Écrire le code de la méthode `calcul_TdB` de la classe `Carte` afin de calculer le nombre de "têtes de bœuf" pour une carte donnée.

On veut ajouter à la classe `Carte` une méthode `est_superieure_a` qui prend en paramètre une carte `autre` (de type `Carte`) et qui renvoie `True` si la valeur de la carte considérée est strictement supérieure à la valeur de la carte `autre`.

3. Écrire le code de cette méthode.

La classe `Carte` possède également la méthode `difference` qui prend en paramètre une carte `autre` (de type `Carte`) et qui renvoie la valeur absolue de la différence entre les valeurs des deux cartes considérées.

Partie B : La classe `Paquet`

Maintenant que les cartes sont modélisées, on crée une classe `Paquet` pour gérer les différents paquets de cartes durant ce jeu.

Le constructeur de la classe `Paquet` prend en paramètre une liste (de type `list`) contenant des cartes (de type `Carte`) que l'on affecte à l'attribut `contenu`.

On ajoute deux méthodes à cette classe `Paquet` :

- la méthode `afficher` permet d'afficher la valeur de toutes les cartes du paquet,
- la méthode `ajouter_carte` prend en paramètre une carte (de type `Carte`) et l'ajoute au contenu du paquet considéré.

4. Compléter le code suivant en recopiant les lignes 6, 7 et 10 sur votre copie :

```
1 class Paquet:
2     def __init__(self, L):
3         self.contenu = L
4
5     def afficher(self):
6         ...
7         ...
8
9     def ajouter_carte(self, carte):
10        ...
```

5. Écrire le code de la méthode `nombre_TdB` de la classe `Paquet` qui renvoie le nombre total de "têtes de bœuf" contenus sur les cartes de ce paquet.

La classe `Paquet` contient aussi une méthode `distribuer` qui prend en paramètre un entier naturel `nbr` correspondant au nombre de joueurs (au maximum 10) à servir et qui renvoie la liste contenant les `nbr` paquets.

Cette liste est initialisée avec `nbr` instances de classe `Paquet` dont le contenu est vide.

Chaque joueur reçoit 10 cartes. La première carte du paquet est donnée au premier joueur, la deuxième au deuxième joueur et ainsi de suite.

Ainsi chacun des `nbr` paquets contiendra 10 cartes prises du paquet sur lequel on appelle la méthode (donc enlevées de ce paquet initial).

6. Écrire le code de la méthode `distribuer` de la classe `Paquet`.

La classe `Paquet` possède également les méthodes suivantes :

- la méthode `prendre_carte` n'a pas de paramètre et renvoie la carte choisie par l'utilisateur dont il saisit la valeur lors de l'appel de cette méthode (on demande

une valeur tant que celle-ci n'est pas valable, ainsi la carte est obligatoirement une carte du paquet).
Cette carte est supprimée du paquet ;

- la méthode `trier` n'a pas de paramètre et permet de trier les cartes du paquet dans l'ordre croissant de leur valeur.

Partie C : La classe `Joueur` et modélisation du plateau de jeu -> classe `Plateau`

Il reste à gérer les joueurs : leur nom, leur paquet de cartes, etc.
C'est le rôle de la classe `Joueur`.

Le constructeur de la classe `Joueur` prend en paramètres :

- le nom du joueur (de type `str`) que l'on affecte à l'attribut `nom` ;
- un paquet de cartes (de type `Paquet`) que l'on affecte à l'attribut `main`.

On ajoute également les attributs `cartes_ramassees` qui est initialisé avec un paquet de cartes vide et `penalite` initialisé à 0.

```
1 class Joueur():
2     def __init__(self, nom, main):
3         self.nom = nom
4         self.main = main
5         self.cartes_ramassees = Paquet([])
6         self.penalite = 0
```

7. Écrire une commande Python permettant d'instancier le joueur nommé `Joueur 1` ayant pour paquet de cartes `L[0]` et qui affecte l'objet créé à la variable `J1`.

La classe `Joueur` possède également la méthode `ramasser_paquet` qui prend en paramètre un paquet (de type `Paquet`), l'ajoute à l'attribut `cartes_ramassees`, et met à jour l'attribut `penalite` en lui ajoutant le nombre total de "têtes de bœuf" contenus dans le paquet ramassé.

Maintenant que les trois classes sont prêtes, on initialise le jeu pour deux joueurs dans le programme principal.

8. Compléter le script ci-dessous en recopiant les lignes 3, 7 et 9 sur votre copie :

```
1   from random import *
2   # créer les 104 cartes du jeu initial grâce à une liste par
compréhension
```

```
3   jeu = [... for i in range (..., ...)]
4   # mélanger cette liste de cartes
5   shuffle(jeu)
6   # instancier le paquet de cartes avec cette liste de cartes
7   jeu_initial = ...
8   # distribuer 10 cartes aux deux joueurs que l'on intancie en les
nommant `J1` et `Ordi`.
9   distri = ...
10  Ordi = Joueur("Ordi", distri[0])
11  J1 = Joueur("J1", distri[1])
```