Jeu de bataille

I. Introduction et objectif

L'objectif de ce projet est de réaliser un jeu de bataille comprenant 4 classes dans des 4 fichiers distincts et un programme principal utilisant ces classes.

Vous devrez rendre :

- Un fichier python pour chaque classe contenant la définition de la classe, des commentaires adaptés et la fonction de test de la classe.
- Un fichier contenant le programme principal du projet avec vos noms et prénoms.

Exemple de sortie en cours d'exécution :

```
# Dans la console PYTHON
Debut: 26 26
La carte du joueur Toto est: Dame de TREFLE
La carte du joueur Dupont est: 3 de COEUR
Le joueur Toto a gagne ce tour
Nombre de cartes en sortie du dernier coup:
NbCartes de Toto : 27
NbCartes de Dupont : 25

La carte du joueur Toto est: 2 de TREFLE
La carte du joueur Dupont est: As de TREFLE
Le joueur Dupont a gagne ce tour
Nombre de cartes en sortie du dernier coup:
NbCartes de Toto : 26
NbCartes de Dupont : 26

La carte du joueur Toto est: 10 de COEUR
La carte du joueur Dupont est: Roi de TREFLE
Le joueur Dupont a gagne ce tour
Nombre de cartes en sortie du dernier coup:
NbCartes de Toto : 25
NbCartes de Dupont : 27
```

II. Description

1) Description de la classe Carte

```
# Variables globales
couleurs = ('CARREAU', 'COEUR', 'TREFLE', 'PIQUE')
noms = ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'Valet', 'Dame', 'Roi','As']
valeurs = {'2': 2, '3': 3, '4': 4, '5': 5, '6': 6, '7': 7, '8': 8, \
'9': 9, '10': 10, 'Valet': 11, 'Dame': 12, 'Roi': 13, 'As': 14}

# Classe Carte
class Carte:
    def __init__(self, nom, couleur):
        # Affectation des attributs nom et couleur avec contrôle.
        self.nom = nom
        self.couleur = couleur
        self.valeur = ...

######### Définition des méthodes d'instances avec contrôle #######
```

```
    def setNom(self, nom): # setter
        ...

    def getNom(self): # getter
        ...

    def getCouleur(self): # getterDescription
        ...

    def getValeur(self): # getter
        ...

    def egalite(self, carte):
        ''' Renvoie True si les cartes ont même valeur, False sinon
        carte: Objet de type Carte '''

    def estSuperieureA(self, carte):
        ''' Renvoie True si la valeur de self est supérieure à celle de carte,
        False sinon
        carte: Objet de type Carte '''

    def estInferieureeA(self, carte):
        ''' Renvoie True si la valeur de self est inferieure à celle de carte,
        False sinon
        carte: Objet de type Carte '''
```

Fonction test et vérification de la classe Carte à compléter.

```
def testCarte():
    valetCoeur = Carte('Valet', 'COEUR')
    print('Nom:', valetCoeur.getNom())
    print('Couleur:', valetCoeur.getCouleur())
    print('Valeur:', valetCoeur.getValeur())
    valetCoeur.setNom('Dame')
    print('Nom modifie:', valetCoeur.getNom())
    print('Valeur modifiee:', valetCoeur.getValeur())
    # Essai des exceptions: cette instruction conduit à une erreur
    dameCarreau = Carte('Dame', 'COooEUR')
```

Ce qui doit nous donner :

```
Nom: Valet
Couleur: COEUR
Valeur: 11
Nom de fichier image: COEURValet.png
Nom modifie: Dame
Valeur modifiee: 12
Le couleur de la carte est incorrect: COooEUR
```

2) Description de la classe JeuCartes et le détail de l'implémentation de la classe JeuCartes

```
from carte import * # Il faut importer la classe Carte et les variables globales
import random # Nécessaire pour mélanger le jeu
class JeuCartes():
    def __init__(self, nbCartes=52):
        # Le jeu doit comporter 32 ou 52 cartes, effectuer un contrôle
        self.jeu = [] # self.jeu est une liste des self.nbCartes
        ... # à compléter

    ################## Définition des méthodes d'instances ####################
    def getTailleJeu(self):
        ''' Fonction qui retourne le nombre de cartes du jeu
        Valeur retournée: type int '''
```

```
    def creerJeu(self): # utilise des objet
        '''Créée la liste des cartes de l'attribut self.jeu '''

    def getJeu(self):
        '''Renvoie la liste des cartes correspondant à l'attribut self.jeu'''
    def melanger(self): # utiliser le module random ...
        '''Mélange sur place les cartes de la liste des cartes associée
        au champ self.jeu'''

    def distribuerCarte(self):
        ''' Cette fonction permet de distribuer une carte à un joueur.
         Elle retourne la carte Valeur retournée: Objet de type Carte '''

    def distribuerJeu(self, nbJoueurs, nbCartes):
        ''' Cette méthode distribue nbCartes à chacun des nbJoueurs, ... '''
```

Fonction test et vérification de la classe JeuCartes.

```
def testJeuCartes():
    jeu52 = JeuCartes(52)
    jeu52.melanger()
    L=jeu52.getJeu()
    carte= L[2] # le 3e carte
    print('Nom:', carte.getNom())
    print('Couleur:', carte.getCouleur())
    print('Valeur:', carte.getValeur())
    # Distribution de 4 cartes à 3 joueurs
    distribution_3j_4c = jeu52.distribuerJeu(3, 4)
    for i in range(3):
        print('Joueur', i+1, ':')
        listeCartes = distribution_3j_4c[i]
        for c in listeCartes:
            print(c.getNom(), 'de', c.getCouleur())

   # Distribution de 10 cartes à 6 joueurs pour générer une exception (6X10 > 52)
   distribution_6_joueurs_10_cartes_par_joueur = jeu52.distribuerJeu(6, 10)
```

Ce qui doit nous donner :

```
Joueur 1 :
6 de TREFLE
Dame de CARREAU
6 de CARREAU
Dame de TREFLE
Joueur 2 :
3 de COEUR
8 de TREFLE
Valet de COEUR
5 de TREFLE
Joueur 3 :
Roi de TREFLE
As de COEUR
10 de PIQUE
9 de PIQUE
Pas assez de cartes dans le jeu.
```

3) Description de la classe Joueur et détail de l'implémentation de la classe Joueur

Créer une classe Joueur ayant les attributs suivants :

1. **nom** : Nom du joueur;

2. **nbCartes** : Correspond au nombre de cartes dans la main du joueur;
3. **mainJoueur** : Liste des cartes (objets de type Carte) dans la main du joueur.

Cette classe devra implémenter les méthodes suivantes :
1. **setMain()** : Définit la main du joueur, donc la liste de ses cartes au début du jeu;
2. **getNom()** : Accesseur de l'attribut nom;
3. **getNbCartes()** : Accesseur du champ nbCartes;
4. **jouerCarte()** : Enlève et renvoie la dernière carte (objet de type Carte) de la main du joueur pour la jouer, ou retourne None s'il n'y a plus de cartes dans la main du joueur;
5. **insererMain()** : Fonction qui insère les cartes de la liste des cartes gagnées dans la main du Joueur.

Fonction test et vérification. A faire.

4) Description de la classe Bataille

La classe bataille doit instancier un jeu de cartes, deux joueurs et implémenter la méthode jouer. Tester votre classe. On utilisera l'algorithme suivant pour la bataille.

Algorithme de la bataille

tant que le nb de cartes du joueur1 est non nul ou le nb de cartes du joueur2 est non nul faire
    Prendre la carte du dessus du joueur 1 et la retourner.
    Mettre cette carte sur le dessus de la pile de bataille1
    Prendre la carte du dessus du joueur 2 et la retourner
    Mettre cette carte sur le dessus de la pile de bataille 2
    tant que les 2 cartes du dessus des piles de bataille sont égales faire
        Prendre la carte du dessus du joueur 1
        Mettre cette carte face cachée sur le dessus de la pile de bataille 1
        Prendre la carte du dessus du joueur et la retourner face visible
        Mettre cette carte sur le dessus de la pile de bataille 1
        Prendre la carte du dessus du joueur 2
        Mettre cette carte face cachée sur le dessus de la pile de bataille 2
        Prendre la carte du dessus du joueur 2 et la retourner face visible
        Mettre cette carte sur le dessus de la pile de bataille 2
    fin tant que
    Joueur gagnant ce tour : celui qui a la plus forte carte en sommet de pile de bataille
    tant que la pile de bataille 1 n'est pas vide faire
        Prendre la carte sur le dessus de la pile de bataille 1
        La glisser face cachée sous le paquet du gagnant
    fin tant que
    tant que la pile de bataille 2 n'est pas vide faire
        Prendre la carte sur le dessus de la pile de bataille 2
        La glisser sous le paquet du gagnant
    fin tant que
fin tant que

Le gagnant est celui qui a toutes les cartes dans son jeu.

5) A vous de faire le programme principal