# Projet - Programmation orientée objet : Un jeu de cartes

### 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 :

1. un fichier présentant votre projet (sous Word, ou Openoffice) ;
2. un fichier python pour chaque classe contenant la définition de la classe, des commentaires adaptés et la fonction de test de la classe ;
3. le fichier principal du projet avec vos noms et prénoms ;
4. présentation : un support de présentation orale sous Powerpoint.

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
```

## 1 Description de la classe Carte

### 1.1 Détail de l'implémentation

```
# Dans l'éditeur PYTHON : fichier \textit{carte.py}

# 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 = valeur
    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
    ...
Projet - Programmation orientée objet
  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    '''
```

## 1.2 Fonction test et vérification de la classe Carte.

```
# Dans l'éditeur PYTHON : fichier  carte.py

#########################################################
################### Test de la classe Carte ###########
#########################################################

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 :

```
# Dans la console PYTHON

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

# 2 Description de la classe JeuCartes

## 2.1 Détail de l'implémentation de la classe JeuCartes

Description

```
# Dans l'éditeur PYTHON : fichier jeucartes.py

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, ... '''
```

## 2.2 Fonction test et vérification de la classe JeuCartes.

```
# Dans l'éditeur PYTHON : fichier jeucartes.py

################### Test 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 :

```
# Dans la console PYTHON
Nom: Dame
Couleur: COEUR
Valeur: 12
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

## 3.1 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

## 3.2 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. A vous de faire le programme principal.