1. Agrégation

La conception d'une classe a pour but généralement de pouvoir créer des objets qui suivent tous le même modèle de fabrication. Un objet dans la vraie vie, par exemple votre stylo, est composé d'autres objets : une pointe (ou plume), un réservoir d'encre, éventuellement un capuchon et un ressort....
Votre stylo est ce qu'on appelle un objet agrégat et son réservoir d'encre est donc un objet composant.

1.1. Exemple d'agrégation

En parlant de stylo, voici un exemple très simple.
Commençons par la définition de la classe composant **Reservoir** :

```
# Fichier reservoir.py

class Reservoir:
    '''classe permettant de construire un réservoir d'encre pour des stylos
    toutes marques, toutes dimensions'''

    def __init__(self, couleur):
        '''On se contente d'un seul paramètre pour l'exemple
        les dimensions ne seront donc pas incluses dans cette description '''
        # un seul attribut toujours par souci de clarté
        self.couleur = couleur

    # Accesseur de self.couleur
    def get_couleur(self):
        return self.couleur

    # Mutateur de self.couleur
    def set_couleur(self, couleur):
        self.couleur = couleur
```

Maintenant, voyons la classe agrégat **Stylo** et son utilisation de la classe **Reservoir** :

```
# Fichier stylo.py

from reservoir import*

class Stylo:
    ''' classe permettant de construire un stylo avec un réservoir d'encre.
    On ne s'occupe pas de ses autres caractéristiques

    '''def __init__(self, couleur):
        ''' On se contente d'un seul paramètre pour l'exemple
        les dimensions ou autres composants ne seront donc
        pas inclus dans cette description '''
        self.reservoir = Reservoir(couleur)

    # Accesseur du self.couleur de self.reservoir
    def get_couleur(self):
        return self.reservoir.get_couleur()

    # Mutateur du self.couleur de self.reservoir
    def set_couleur(self, couleur):
         self.reservoir.set_couleur(couleur)
```

Maintenant, voyons comment on créé simplement un stylo rouge dans l'éditeur python (programme) :

```
# Fichier main.py

from stylo import*

pen = Stylo("Rouge")
print(pen.get_couleur())

# Changeons la cartouche d'encre
pen.set_couleur("Bleu")
print(pen.get_couleur())
```

Le résultat donne ceci dans la console python :
Rouge
Bleu

A noter : dans le fichier principal de votre programme, ici main.py, vous n'avez pas importé le fichier reser-voir.py, vous n'avez même pas besoin de savoir qu'il existe et encore moins de savoir comment il est conçu. Et pourtant, vous l'utilisez indirectement : en instanciant la classe Stylo, vous instanciez également la classe Reservoir. Et vous obtenez un objet stylo un peu plus complexe qu'il n'y parait. Imaginez maintenant que vous vouliez un stylo quatre couleurs : oui, il vous faudra 4 instances de Reservoir dans Stylo. Ce qui entraînera une modification des accesseurs et mutateurs. Et certainement une méthode de sélection de la couleur. Les possibilités sont grandes. Cette architecture nécessite en général la création d'un fichier par classe. Elle permet de transformer une classe sans toucher aux autres. Où tout simplement, de se partager le travail dans une équipe.

2. Jeu de cartes

Voici la classe Carte qui permet de créer une carte à partir du rang du nom et de la couleur.

```
#fichier carte.py

class Carte: #définition de la classe
    """Affectation de l'attribut nom et de l'attribut couleur"""
    def __init__(self, nom, couleur): #constructeur
        noms = ['2','3','4','5','6','7','8','9','10','Valet','Dame','Roi','As']
        couleurs = ['CARREAU','COEUR','TREFLE','PIQUE']
        self.nom = noms[……] #premier attribut
        self.couleur = couleurs[……] #deuxième attribut

    def get_nom(self): #accesseurs
        """retourne le nom de la carte"""

    def get_couleur(self): #accesseurs
        """retourne,la couleur de la carte"""

    def get_attribut(self): #accesseurs
        """retourne le nom et la couleur de la carte"""
```

✍ Compléter le constructeur et les accesseurs de la classe Carte (s'aider du résultat de la console ci-dessous).

```
>>> c = Carte(1,1)
>>> c.get_attribut()
('3', 'COEUR')
```

Voici la classe JeuDeCartes qui à 52 instances (objet) de la classe Carte.

```
#fichier jeu_carte.py

import random
……

class JeuDeCartes:
    def __init__(self): #constructeur de la classe
        self.paquet_carte = []

    def créer_paquet(self): #mutateurs
        """crée un paquet de carte"""
        for couleur in range(4):
            for nom in range(13):
                ……#instance d'une carte
                ……#ajout de la carte au paquet

    def distribuer_une_carte(self): #mutateurs
        """distribue la première carte du paquet"""
        carte = …… #première carte du paquet
        …… # on enlève la carte distribuée du paquet
        …… # retourne les attributs de la carte

    def get_paquet(self): #accesseurs
        """renvoie toutes les cartes restantes dans le paquet"""
        paquet_restant = []
        …… # remplissage de paquet_restant
                :
        ……

    def battre_jeu(self): #mutateurs
        """mélange le jeu de cartes"""
        …… #utiliser la méthode shuffle de random

    def affiche_paquet(self): #accesseurs
        """affiche les cartes restantes dans le paquet"""
        ……
```

✍ Compléter le fichier jeu_carte.py ainsi que les méthodes de la classe JeuDeCartes (s'aider du résultat de la console ci-dessous).

```
>>> p = JeuDeCartes()
>>> p.creer_paquet()
>>> p.distribuer_une_carte()
('2', 'CARREAU')
>>> p.battre_jeu()
>>> p.distribuer_une_carte()
('3', 'COEUR')
>>> p.get_paquet()
[('8', 'PIQUE'), ('Dame', 'COEUR'), ('6', 'COEUR'), ('6', 'TREFLE'), ('8', 'TREFLE'), ('As',
'COEUR'), ('7', 'PIQUE'), . . . ('3', 'CARREAU'), ('Roi', 'PIQUE')]
>>> p.affiche_paquet()
 8 de PIQUE
 Dame de COEUR
 6 de COEUR
 6 de TREFLE
  . . .
 Roi de PIQUE
```

✍ Créer un fichier qui (ci-dessous, en aide un exemple de résultat obtenu à la console) :

- Instancie un jeu de carte
- Crée un paquet
- Affiche deux cartes
- Mélange le paquet
- Affiche deux cartes
- Affiche le reste du paquet

Un exemple de résultat à la console :

```
('2', 'CARREAU')
('3', 'CARREAU')
('6', 'CARREAU')
('Roi', 'PIQUE')
[('As', 'CARREAU'), ('6', 'PIQUE'), ('2', 'TREFLE'), ('7', 'COEUR'), ('5', 'CARREAU'), ('8', 'COEUR'), ('4', 'COEUR'), ('10', 'PIQUE'), ('7', 'TREFLE'), ('As', 'COEUR'), ('Dame', 'COEUR'), ('6', 'TREFLE'), ('7', 'PIQUE'), ('Valet', 'CARREAU'), ('Valet', 'PIQUE'), ('Dame', 'TREFLE'), ('Dame', 'PIQUE'), ('Valet', 'COEUR'), ('Roi', 'COEUR'), ('5', 'TREFLE'), ('2', 'COEUR'), ('9', 'PIQUE'), ('4', 'TREFLE'), ('8', 'TREFLE'), ('9', 'TREFLE'), ('Roi', 'TREFLE'), ('6', 'COEUR'), ('As', 'PIQUE'), ('10', 'COEUR'), ('10', 'TREFLE'), ('7', 'CARREAU'), ('3', 'TREFLE'), ('3', 'PIQUE'), ('5', 'COEUR'), ('Valet', 'TREFLE'), ('3', 'COEUR'), ('Dame', 'CARREAU'), ('8', 'PIQUE'), ('10', 'CARREAU'), ('4', 'PIQUE'), ('9', 'CARREAU'), ('4', 'CARREAU'), ('5', 'PIQUE'), ('Roi', 'CARREAU'), ('8', 'CARREAU'), ('9', 'COEUR'), ('2', 'PIQUE'), ('As', 'TREFLE')]
```

Source : http://www.math93.com/images/pdf/NSI/terminale/NSI_Classes.pdf