Programmation Orientée Objet

TD 2 : Composition et agrégation

1 Des Dés

Nous allons réutiliser les dés de la séance 1 pour les composer à une autre classe qui permettra de jouer à un jeu comportant plusieurs dés.

1.1 Un dé

Rappelez le modèle UML du dé de la séance 1 et donnez un petit exemple de programme lançant 5 dés différents.

1.2 Tapis Vert

Pour pouvoir jouer à des jeux de dés, implémentons une classe `TapisVert`. Cette classe doit avoir :

- 5 dés comme attribut (une liste de 5 dés nommée `dices`),
- pouvoir lancer les 5 dés simultanément (méthode `roll`),
- Connaître la somme des valeurs des dés (méthode `sum`).

1. retranscrivez le code la question précédente en utilisant ma classe `TapisVert`
2. Proposez un modèle UML pour cette classe. Quel lien la classe `TapisVert` a-t-elle avec la classe `Dice` ?
3. Ecrire le code python de la classe.
4. comment est-il possible d'avoir à la fois une méthode `roll` pour `Dice` et pour `TapisVert` sans que python s'embrouille ?
5. exécutez le code précédent en explicitant tous les namespaces utilisées (namespace de classe, d'objet, de fichier et de fonctions)

2 Des Cartes

2.1 Une carte

Donnez le diagramme UML d'une classe `Card` définie par une couleur et une valeur.

2.2 Un tas de cartes

Un `Deck` est un tas de cartes, initialement vide auquel on peut ajouter une carte, dont on peut voir la carte du dessus et dont on peut prendre la carte du dessus (la piocher i.e la récupérer et l'enlever du paquet). Proposez un diagramme UML pour cette classe. Quel est son lien avec la classe `Card` ?

2.3 Les couleurs

On veut que les couleurs des cartes soient communes à toutes les cartes. Une façon de faire est de mettre les différentes possibilités *dans* la classe carte. Proposez une modélisation UML/python de ceci.

3 Fonctions et namespaces

3.1 Les fonctions sont des variables comme les autres

```
une_liste = []
une_liste.append("premier")
truc = une_liste.append
truc("?")

print(une_liste)
```

Dans le code ci-dessus, que représente `append` ? Exécutez-le en montrant toutes les lignes de codes et les *namespaces* utilisés.

3.2 Fonctions de fonctions

On souhaite créer une fonction `ajoute` avec un paramètre (entier) `x`. Le retour de cette fonction doit être une fonction à un paramètre `y` qui rend `x + y`

3.2.1 Implémentation

Codez cette fonction et vérifiez (à la main) qu'elle passe bien vos tests en notant toutes les variables et les namespaces rencontrés.