Modélisation et Conception Objet

TD 1 : Introduction

1 Un Dé

1.1 Modèle

On veut créer une classe `Dé`. Elle doit être capable de :

- créer un objet sans paramètre,
- créer un objet avec sa valeur initiale,
- connaitre et donner la valeur du dé (avec les méthodes `get_valeur` et `set_valeur`),
- lancer un dé,
- connaitre le nombre maximum de faces (`NUMBER_FACES`), commun à tous les dés,
- s'afficher sous forme de chaine de caractères.

Proposez une modélisation UML de la classe `Dé` et écrivez des tests (en python) permettant de vérifier que tout se passe comme prévu.

1.2 Namespaces

Exécutez le code suivant en montrant tous les *Namespaces* utilisés.

```
from dice import Dice

dice = Dice()
dice.roll()
print(dice.get_valeur())
print(dice.NUMBER_FACES)
```


2 Des Dés

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

- 5 dés comme attribut,
- pouvoir lancer les 5 dés simultanément ou individuellement,
- Connaître la valeur d'un dé spécifique.

3 Arbres et objets

Nous voulons pouvoir rapidement connaître le dés ayant la valeur la plus grande. Plutôt que de vérifier la valeur de chaque dé, on va utiliser une structure en *tas* (`https://en.wikipedia.org/wiki/Heap_(data_structure)`)

3.1 Tas

Dessinez un tas à 5 nœuds avec les valeurs 1, 3, 3, 6 et 4.

Proposez un modèle UML de nœud et de tas. On doit pouvoir :

- accéder au nœud racine d'un tas,
- pour un nœud donné on doit pouvoir accéder à :
  - son père (ou `None` si c'est la racine),
  - ses fils gauche et droit (ou `None`),
  - sa valeur.

En prenant en compte ce modèle, proposez un algorithme permettant de maintenir la structure de tas si un de ses nœuds est modifié.

3.2 Dés et Tas

Proposez un moyen de lier tas et dés.

Regardez ce que cela donne pour le scénario suivant :

1. on a 5 dés avec comme valeur initiale 1,
2. on lance 1 dé, qui fait 3 avec le quatrième dé du tas.