Modélisation et Conception Objet

TD 3 : Design Patterns

Nous allons ici utiliser des *design pattern*. Ce sont sont des façons d'organiser ses classes en vue de résoudre des problèmes courants en développement.

1 Création d'objets

On considère le (maintenant connu) modèle UML d'un dé:

- attribut de classe : `MAX_FACE = 6`
- attribut : `_value`
- méthodes :
  - `__init__(value=1)`
  - `get_value()`
  - `set_value()`
  - `roll()`

1.1 Factory

Créer une méthode de classe (qu'est-ce ?) pour dé qui crée un dé avec un nombre de faces différent. Le fait de créer des objets via des méthodes de classes est un moyen pratique d'avoir de nombreuses possibilités de création sans avoir trop de paramètres dans la méthode `__init__`. Ce pattern d'appelle : *factory*.

1.2 Héritage

Proposez une modélisation UML d'une spécialisation de dé qui compte le nombre de fois (depuis sa création) qu'une valeur a été trouvée. Ecrire le code correspondant.

Lorsque l'on veut spécifier une classe et que de nombreuses fonctionnalités soit sont identiques soit partagent du code, on peut utiliser l'*héritage*. Cette technique est souvent utilisée via des bibliothèques (par exemple graphiques) qui créent des objets génériques (comme une fenêtre) que l'on spécifie pour ses besoins propres grâce à l'héritage.

2 Memento

Le pattern *memento* permet de sauvegarder une information et de la redonner via une méthode `restore` :

- proposez une modélisation UML de ce pattern.
- déclinez-le pour stocker/replacer la valeur d'un dé.

Ce pattern est souvent la première brique d'une méthode d'UNDO/REDO. Proposez un modèle UML et une façon de faire algorithmique du UNDO.

Que faudrait-il ajouter pour faire également un REDO ?

3 Composition

On va utiliser ici le pattern *composite* qui permet d'utiliser un ensemble d'objets comme un seul.

3.1 Expression arithmétique

On suppose que l'on a 2 opérations : $+$ et $*$. En représentant chaque opération par un nœud proposez une structure en arbre pour représenter l'expression : $2 * (3 * x + y + z) + t$.

Comment l'évaluer ?

3.2 Composition

Proposez une structuration UML pour représenter l'expression arithmétique de la question 3.1. Appliquez cette solution aux dés. On doit par exemple être en mesure de décrire 3d6 + 12

3.3 Spécificité de python

Implémentez la structure UML pour la classe `Dice` de python.