Programmation orientée objet

TD 3 : Héritage

1 Héritage et composition

1.1 Un point

On veut créer une classe `Point` représentant un point en deux dimensions qui pourra calculer sa distance à un autre point. Proposer une modélisation UML de cette classe.

1.2 Un polygone

En utilisant cette classe `Point`, proposer une modélisation UML d'une classe `Polygon` qui doit être capable de :

- créer un polygone vide
- ajouter un point au polygone
- calculer son aire
- calculer son périmètre.

Quel lien y a-t-il entre la classe `Point` et la classe `Polygon`

1.3 Un polygone régulier

On souhaite créer une version plus spécifique d'un polygone : un polygone régulier qu'on pourra créer à partir de son centre, son rayon et son nombre de sommets. On ne pourra plus par contre ajouter de points dans le polygone. Proposer une modélisation UML de cette classe. Quel est son lien avec la classe `Polygon`.

2 L'héritage dans la vraie vie

En pratique, on utilise le plus souvent l'héritage pour réutiliser des classes complexes et déjà définies dans des librairies. Par exemple, on peut utiliser des librairies d'UI pour définir facilement notre propre fenêtre qu'on pourra ainsi utiliser sans avoir à redéfinir chaque fois les paramètres qui seront toujours les mêmes dans notre programme. Proposer le code d'une classe `MaFenetre` héritant de `gui` de la librairie `appJar` que nous avons utilisée dans le TP précédent qui :

- a un fond orange (on pourra utiliser la méthode `setBg(color)` de `appJar`)
- a un titre "Notre programme" (avec `setTitle(title)`)
- est de taille 800x800 (`setGeometry(height, width)`)

3 Design pattern composite

On va utiliser ici le design pattern *composite* qui permet d'utiliser un ensemble d'objets comme on utiliserait un objet simple.

3.1 Expression arithmétique

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

3.2 Composite

Le design pattern composite comprend plusieurs modèles de classes :

- la *feuille* : l'objet de base,
- les *composites* qui représentent les composants qui ont des enfants et qui permettent la manipulation de ces enfants,
- le *composant* qui est l'abstraction de tous les composants (incluant les composites et les feuilles).

On souhaite pouvoir faire des opérations de dés et faire par exemple `3 * d6 + d6` en écrivant `d6.mult(3).add(d6)`. Proposer une modélisation UML permettant de faire de telles opérations et utilisant le design pattern composite.