Programmation orientée objet

Cours 3 – Introduction à UML

UML

- Pour représenter graphiquement les conceptions orientées objet, il existe le UML (Unified Modeling Language).
- Le UML fournit une foule de standards pour représenter un programme de divers angles, parfois structurels, parfois comportementaux, à diverses étapes du développement de l'application.
- Diagrammes comportementaux :
    - Ils représentent les différents comportements que doit supporter l'application.
    - Diagrammes de cas d'utilisation. Ils sont utiles très tôt, lors de la conception d'une application pour documenter les besoins des usagers.
    - Diagrammes de séquences. Ils sont utiles plus tard, pour découper plus en détails chacune des interactions entre les différentes composantes d'une application.
    - etc.
- Diagrammes structurels :
    - Ils représentent la structure de l'application :
        - **Diagrammes de classes. C'est eux qui nous intéresseront le plus dans ce cours.**
        - Diagrammes d'objets ou d'instances.
            - Équivalent du diagramme de classe, mais avec des exemples concrets d'utilisation.
        - Diagrammes de composantes, de déploiement, de packages, etc. Utiles dans le développement d'applications de très grande taille.

Diagrammes comportementaux

- Diagramme de cas d'utilisation :

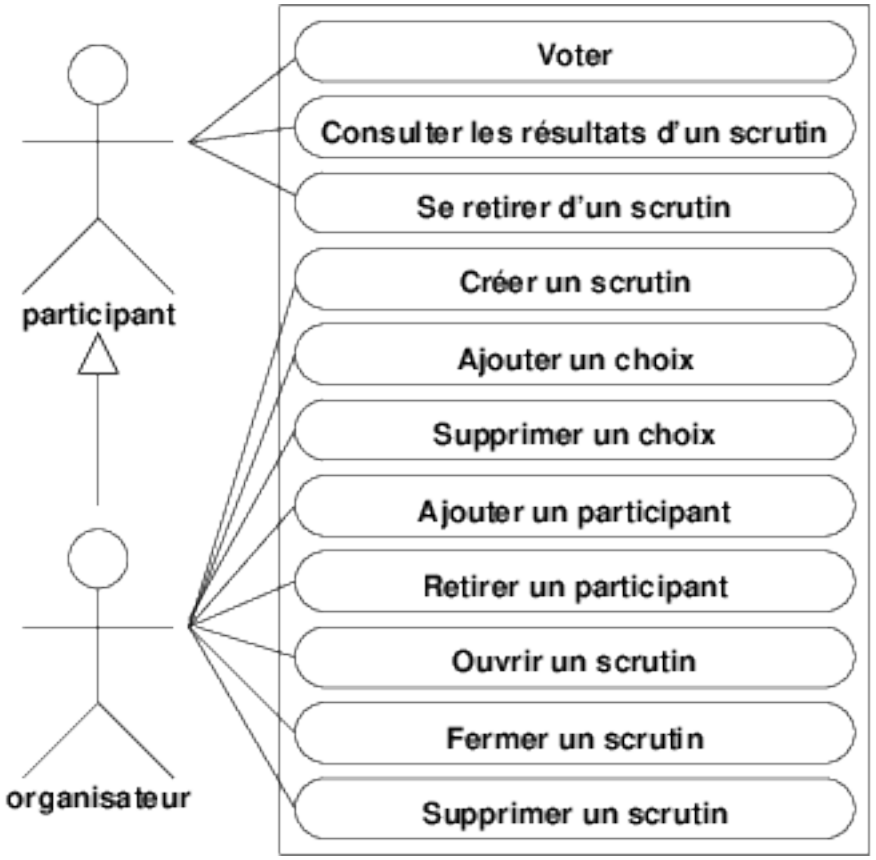
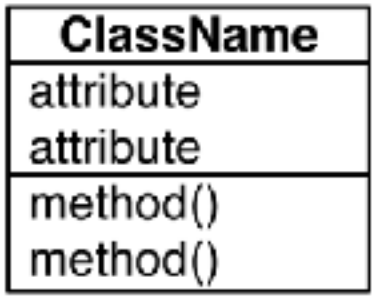
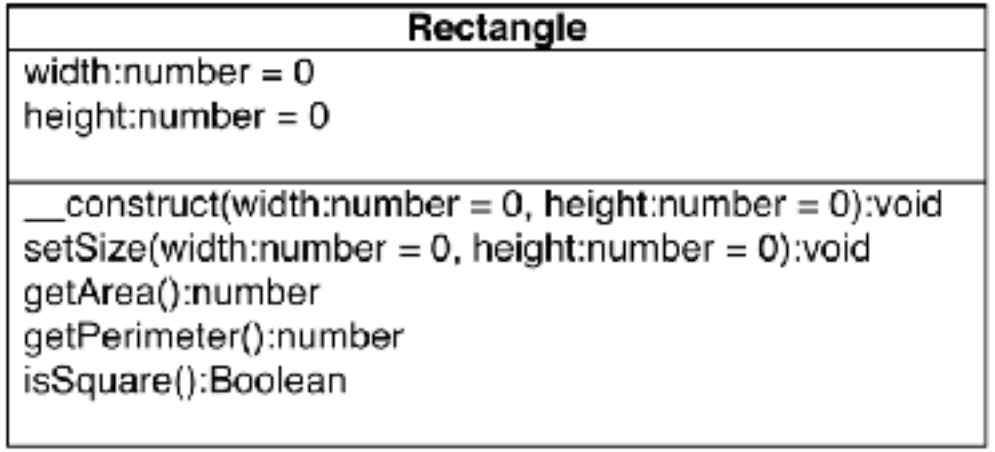


Diagramme de classes

- Le diagramme de classes est un peu l'équivalent du MCD de la méthode Merise que nous avons utilisé pour représenter les données.
  - Cette fois-ci, le diagramme sert à représenter les classes qui composent notre programme et les liens entre elles.
  - Chaque classe est représentée par un carré séparé en trois. On met le nom de la classe dans l'entête, les attributs dans la deuxième section et les méthodes en bas. Généralement, on spécifie aussi le type et les valeurs par défaut pour les attributs et le type de retour des méthodes.





Diagrammes de classes

- Les classes peuvent avoir des relations assez complexes. Nous verrons plus tard les relations d'héritage, entre autres, mais nous avons déjà vu qu'une classe peut en contenir une autre.
  - Chaque type de relation a sa propre nomenclature au niveau de la représentation graphique UML.
  - Nous verrons la représentation des autres types de relation lorsque nous verrons ces types.
- La relation « contient », en conception orientée objet, s'appelle une « composition », et elle s'exprime à l'aide d'une flèche pleine à bout mince qui débute par un losange noir. Dans cet exemple, la classe A1 contient *une* instance de la classe B1.