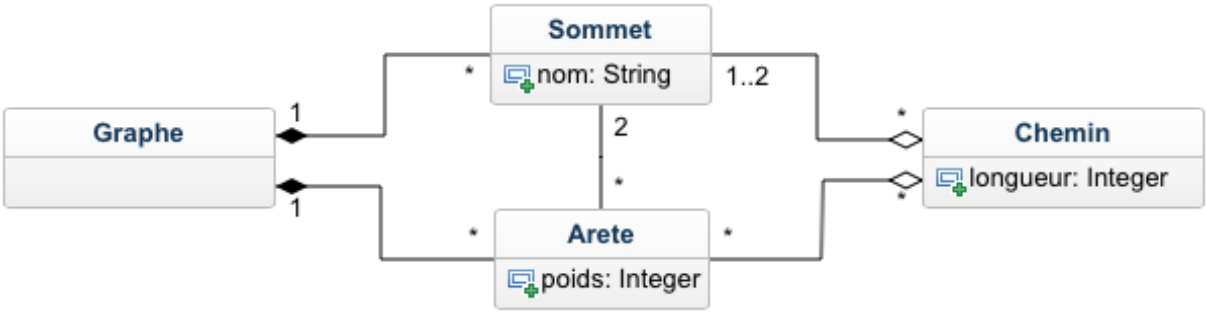
Graphes

POGL, TP4. Diagrammes de classes, conception, traduction en code.

On veut construire une bibliothèque Java permettant de manipuler des graphes non orientés. Une analyse des besoins et des éléments principaux a déjà été faite, et a produit le diagramme de classes suivant :



Les principaux services que doit fournir la bibliothèque sont :

- Création d'un graphe : la classe `Graphe` doit permettre de créer un graphe vide, des nœuds et des arêtes.
- Affichage d'une description d'un graphe : on veut pouvoir afficher dans un format textuel l'ensemble des sommets, chacun accompagné par l'ensemble des arêtes qui lui sont incidentes (la description correspond donc à un dictionnaire d'adjacence).
- Distance entre deux sommets : étant donnés deux sommets on veut pouvoir connaître la longueur du chemin le plus court de l'un à l'autre (la longueur d'un chemin étant la somme des poids de ses arêtes).
- Routage d'un sommet à un autre : étant donnés deux sommets on veut pouvoir obtenir un chemin le plus court de l'un à l'autre.

L'analyse a établi en outre les possibilités et contraintes suivantes :

- Chaque sommet peut stocker une quantité d'information proportionnelle à la taille du graphe.
- La réponse aux requêtes de distance doit se faire en temps constant.
- La réponse aux requêtes de routage doit se faire au plus en temps proportionnel au nombre d'arêtes dans le chemin renvoyé.
- Des calculs plus coûteux sont admissibles lors de la création du graphe.

Pour la modélisation, vous utiliserez l'outil en ligne suivant :
`https://www.genmymodel.com/`
Créez-y un compte gratuit, qui sera suffisant pour les besoins du jour.

Travail demandé

1. On s'intéresse pour l'instant uniquement à la création et à l'affichage des graphes.
   (a) Conception : créez un diagramme de classes plus précis, dans lequel apparaissent notamment
   - les méthodes de chaque classe,
   - la visibilité de chaque attribut ou méthode,
   - la navigabilité de chaque association

   Ceci vous demandera de réfléchir à la manière dont chaque fonctionnalité est décomposée en une séquence de méthodes associées à différentes classes.
   (b) Génération : générez un squelette de code à partir du diagramme de classes avec `GenMyModel`.
   (c) Programmation : corrigez et complétez le squelette pour que les fonctionnalités de création et d'affichage soient opérationnelles.
2. Intégration des requêtes de distance : mettez à jour votre diagramme et votre code pour permettre la réponse à de telles requêtes. Cela vous demandera d'introduire de nouvelles méthodes, mais aussi peut-être de nouveaux attributs ou associations. Vous pourrez également avoir à enrichir certains constructeurs. *N'utilisez plus la génération automatique : vous n'avez pas les outils pour synchroniser avec le code que vous avez ajouté à la main.*
3. Intégrez de même les requêtes de routage.
4. Pour aller plus loin : ajoutez une fonctionnalité de coloration des sommets, de sorte que deux sommets adjacents aient des couleurs différentes.
5. Pour aller plus loin : ajoutez une fonctionnalité d'identification des composantes connexes.
   Mots-clés : *Union-Find*.