Mery de Montigny Arnaud

Fouchet Pierre

B3336

Compte-rendu Tp Héritage - Geom

# Conception Préliminaire

Cette application implémente un éditeur de formes géométriques de manière basique. L’utilisateur, via la console, interagi avec le programme et lui demande l’exécution d’un certain nombre de commandes. Les commandes sont détaillées dans l’énoncé du tp néanmoins quelque éclaircissement concernant l’organisation de l’application est nécessaire.

La classe Controleur s’occupe de récuperer les requêtes du client puis de les exécuter. L’exécution de ces différentes requêtes repose sur le design pattern Command vu en cours. Ainsi, chaque requête du client abouti à la création d’une commande permettant d’uniformiser les traitements et de pouvoir choisir à quel moment exécuter les requêtes. L’uniformisation se matérialise par la présence de 3 méthodes Execute(), Undo() et Redo() et met en place une hiérarchie d’héritage au sein des commandes. En effet, les commandes permettant de créer un rectangle et un cercle sont toutes vues comme des *commandes*, mais également comme des *commandes de création*, une commande move sera seulement une classe dont l’ancêtre direct est *commande*. Cette hiérarchie permet de factoriser le code via les notions d’héritage et permet de mettre en place un polymorphisme efficace dans l’application. La contre-parti est le fractionnement des classes en de nombreuses sous-classes, qui rendent un peu plus complexe le suivi de l’application.

# Diagramme de classes de l’application

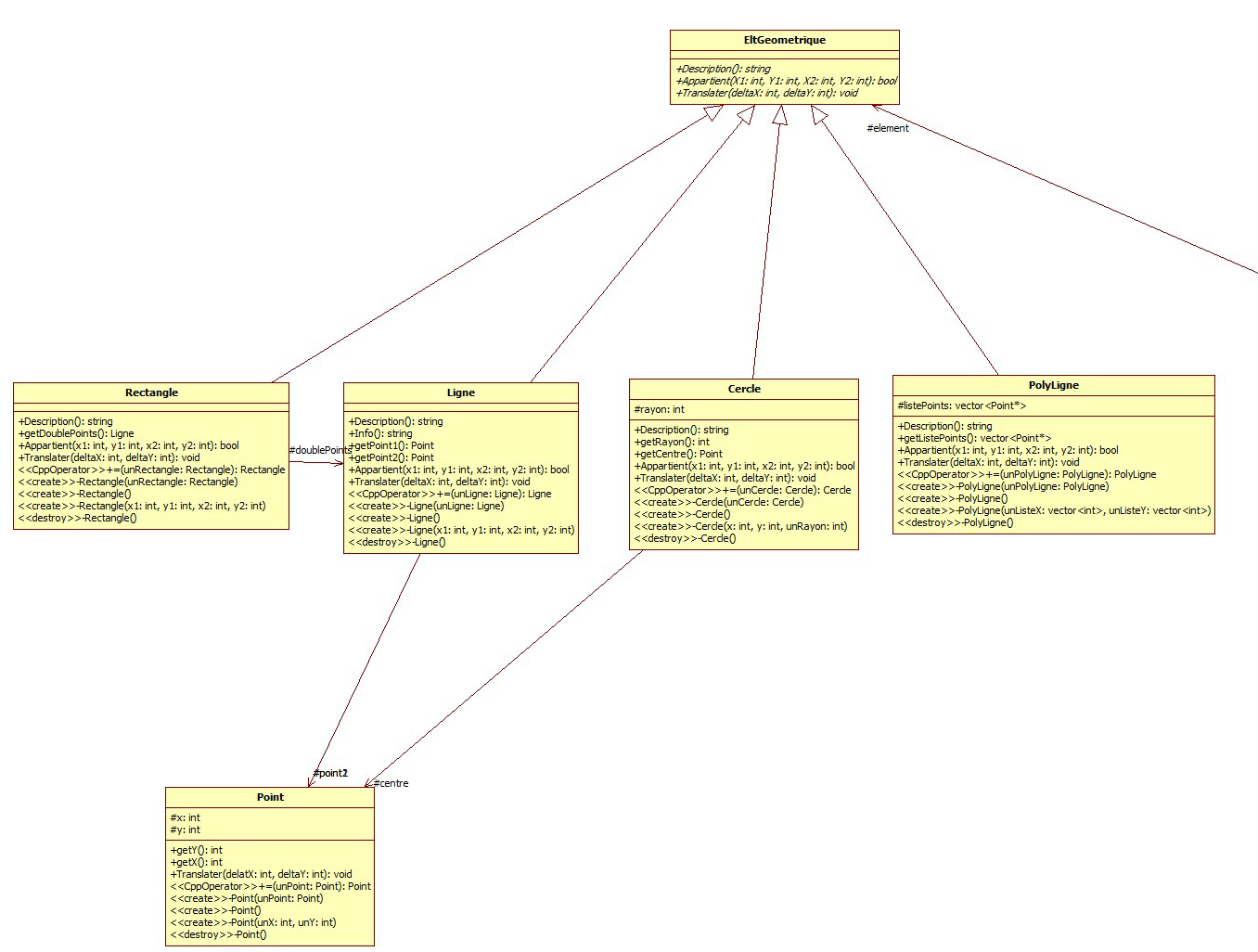


Figure Diagramme de classe partie modèle

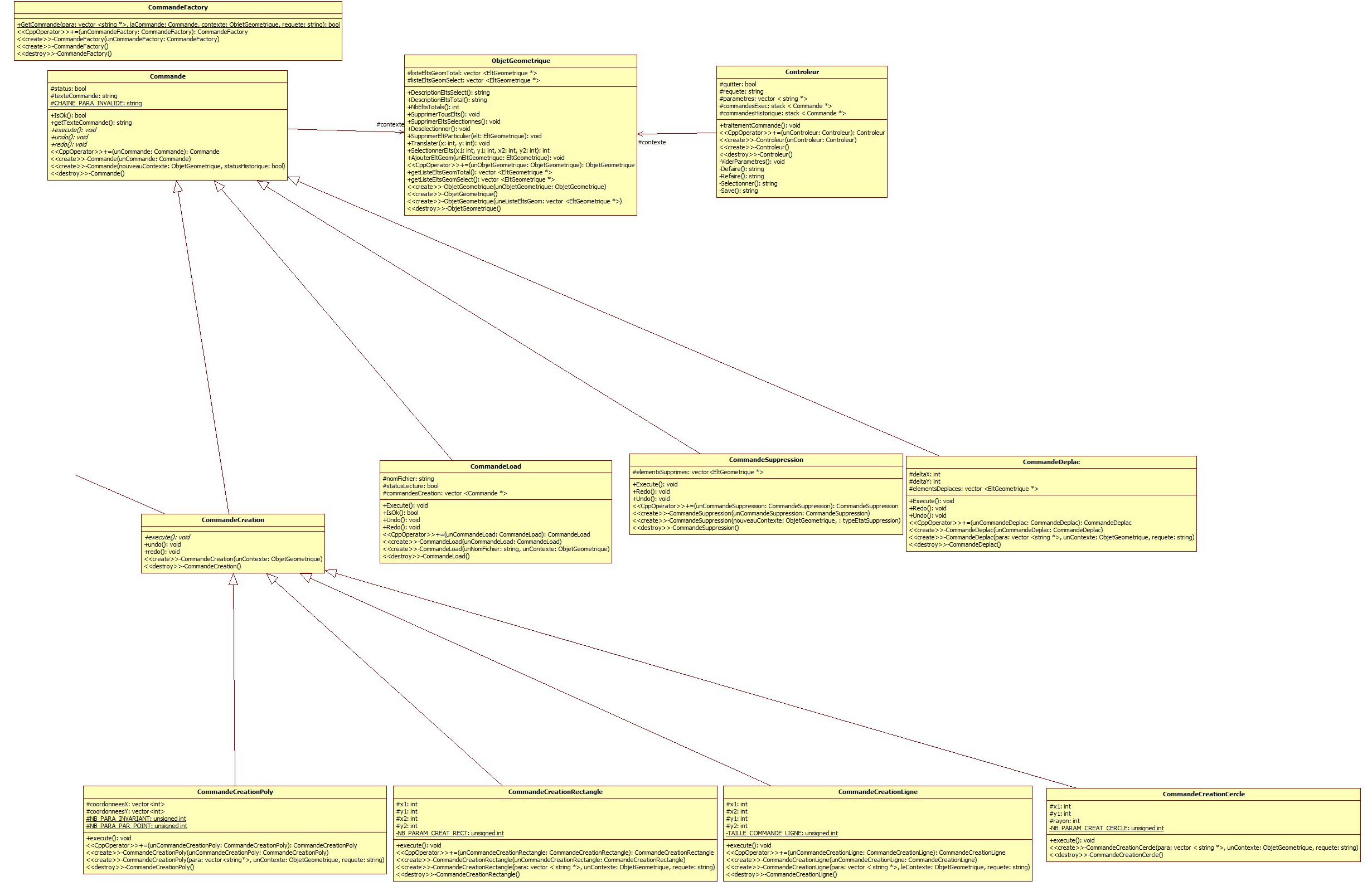


Figure Diagramme de classe partie Contrôleur

# Structure de données

Comme l’indique le diagramme de classes de l’application, le programme utilise de nombreuses listes. Les *stack* ont étés retenues pour stocker les listes de commandes car les fonctionnalités undo et redo ont seulement besoin d’accéder à la dernière commande exécutée (ou annulée, selon le cas). Insertion, accès et suppression se faisant en O(1).

Les *vector* ont été retenus pour stocker les autres ensembles tels que les ensembles d’éléments géométriques. En effet l’insertion se fait en O(1) si l’on fait en tête (ce qui est le cas dans notre application, car il n’y a pas d’ordre d’éléments géométriques) et l’accès peut se faire en O(1). Ce dernier point a soulevé une erreur de conception, trouvée très tardivement. En effet, dans les cas où le programme a besoin de vérifier l’existence d’un élément présent dans un *vector,* celui-ci est obligé à parcourir l’ensemble du conteneur.