GROUPE 7
BAZAYAMA MBOTI Abel (GC)
METENA M'NTEBA Yves (GEI)
YEKWA WAYANDIRI John (GC)

TP ALGORITHMIQUE ET PROGRAMMATION (**PROJET 1**) 2G POLYTECH UNIKIN/2021-2022

Réponse 1 :

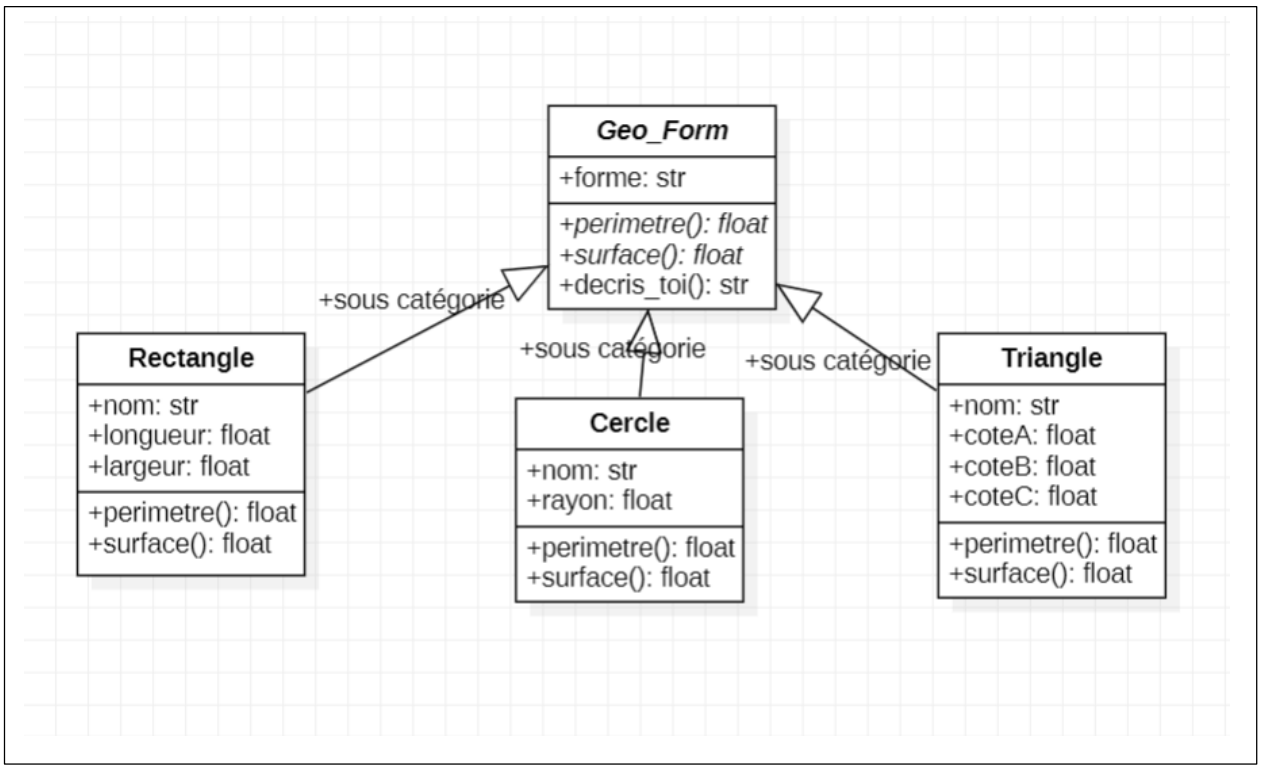
Pour créer la classe demandée nous avons utilisé le module ABC de python, module qui s'occupe des classes abstraites pour créer notre classe ***Geo_Form*** et nous avons défini 2 méthodes abstraites respectivement ***perimetre()*** pour le calcul du périmètre et ***surface()*** pour le calcul de la surface, en dehors de ça nous avons pensé ajouter une troisième méthode **decris_toi()** cette dernière nous facilite sur l'affichage de toutes les valeurs relatives au perimetre et surface de l'objet instancié Le code se trouve se trouve à la page de code (pro1_7.py).

Réponse 2 :

Ici nous avons écrit nos 3 classes : **Rectangle**, **Cercle**, **Triangle** qui sont des classes filles de la classe mère ***Geo_Form***, tout en surchargeant les deux méthodes de base en les affectant les formules dont ils ont besoin pour donner les résultats valables à chaque figure, il est important de signaler que chaque figure prend à des attributs ou propriétés à elle seule qui la définit, ces derniers sont souvent fonctions des côtés et de leurs grandeurs. Le code se trouve se trouve à la page de code (pro1_7.py).

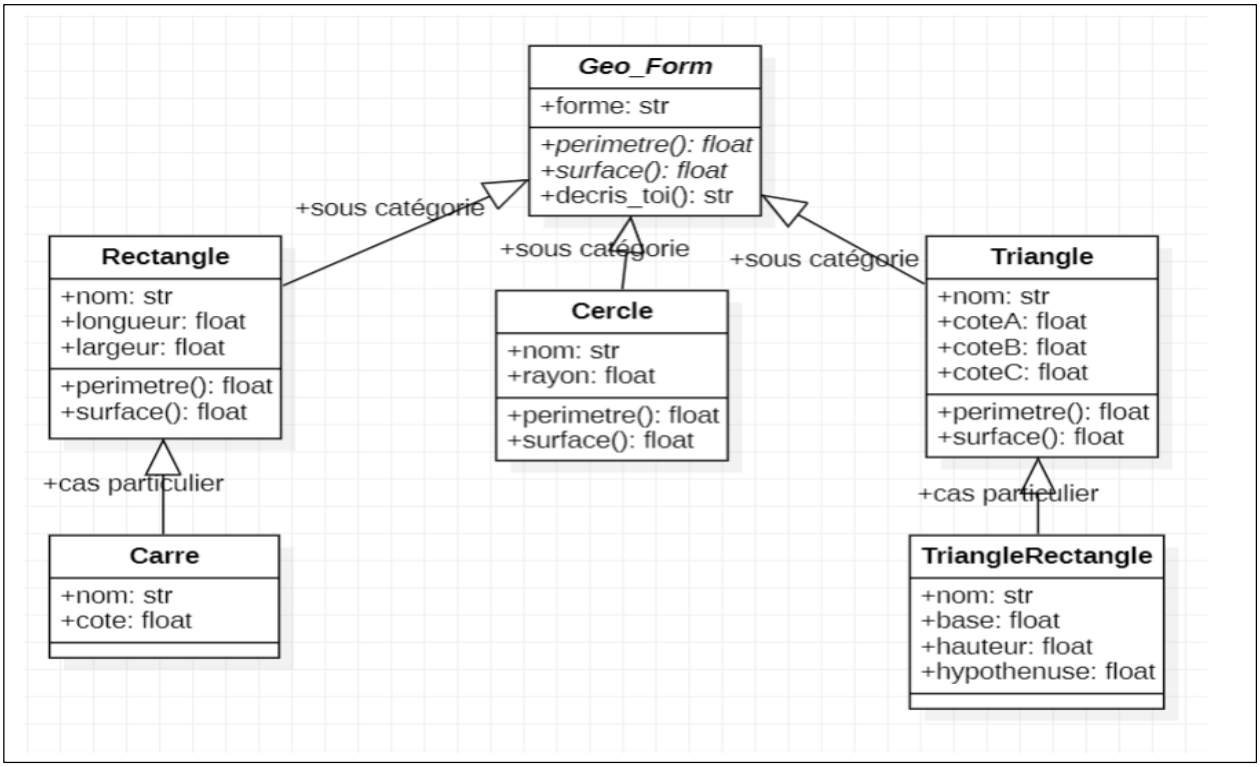
Réponse 3 :

Le diagramme UML pour les classes déjà développées se présente comme suit :



Réponse 4 :

Sachant que le carré est une forme particulière du rectangle déjà définit et ce qui en est aussi le triangle rectangle pour le triangle, alors en se basant sur la notion de l'héritage de la POO, notion qui nous facilite à implémenter des objets enfants partant de leurs parents, nous avons estimé bon, d'utiliser la notion d'héritage pour ces deux classes respectivement **Carre** qui hérite de la classe **Rectangle** ; et **TriangleRectangle** qui celle-ci hérite de la classe **Triangle** ; il est important de savoir que pour les deux classes filles nous avons modifier les attributs mais avons utilisés les méthodes de leurs classes mères. Le code se trouve à la page du code, Ci-dessous le diagramme UML mise à jour :



Réponse 5 :

Sur cette rubrique nous avons défini une class GeoFig, cette classe est reliée avec les autres classes définies ci haut par une relation d'agrégation, c'est aussi une notion de l'héritage multiple qui entre en jeu dans ce cas. Le code se trouve se trouve à la page de code.

Réponse 6 :

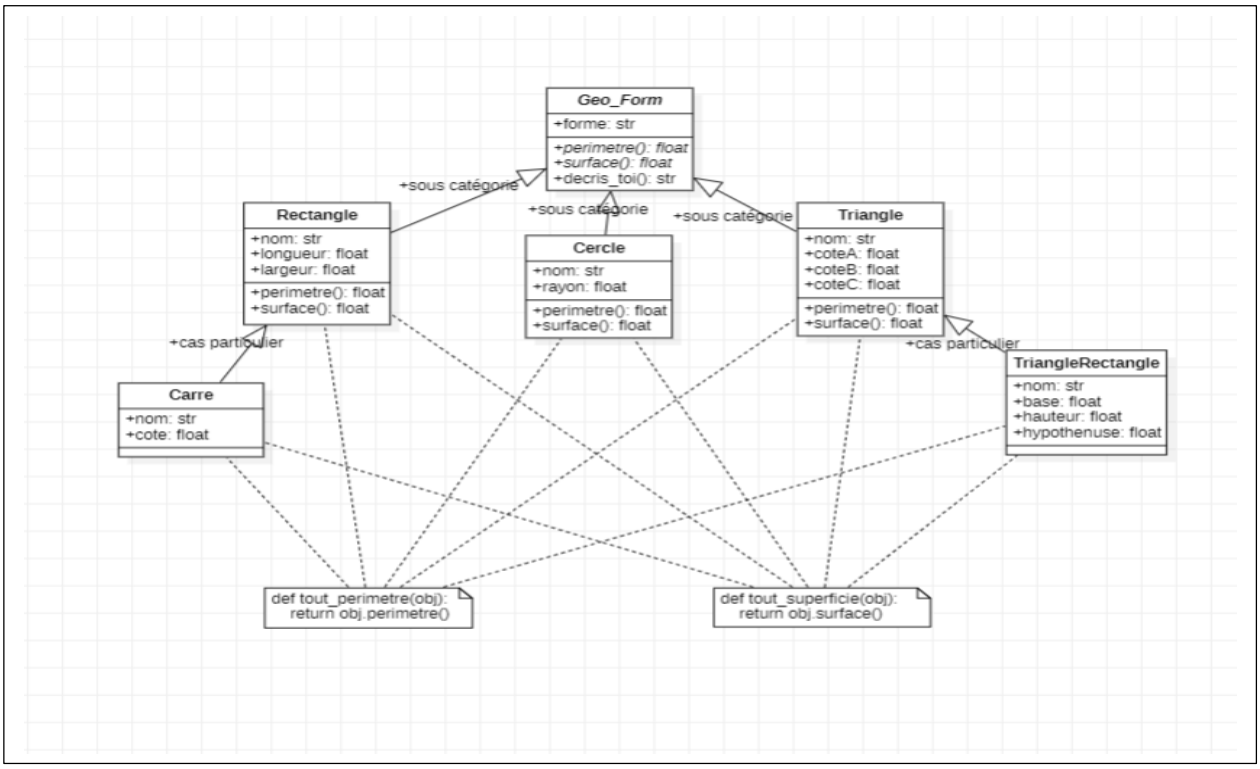
Dans le cadre de ce projet, nous avons développés 7 classes, le nombre minimum des classes est de 4, parce que avec les 4 classe je cite : **Geo_Form**, **Rectangle**, **Cercle**, **Triangle,** on peut implémenter les carrés et les triangles rectangles, bref ces 4 classes sont suffisantes pour répondre à tous nos problèmes se basant sur les figures traitées dans notre projet.

Réponse 7 :

Notre module si situe dans un même dossier avec la batterie tests, le module porte l'identifiant **pro1_7** ceci est exactement le fichier **pro1_7.py** et la batterie tests se trouve dans le fichier **bat_test1_7.py** Le code se trouve se trouve à la page de code (pro1_7.py & bat_test1_7.py).

Réponse 8 :

En dehors de l'héritage, on peut utiliser le **polymorphisme** pour réaliser ce projet, ci-dessous le diagramme UML et le code se trouve se trouve à la page de code (pro1_8.py & bat_test1_8.py).



Réponse 9 :

Après avoir comparé le temps d'exécution par annotation asymptotique, on a trouvé que le temps d'exécution pour les classes avec héritage est une fonction logarithmique log(n) et pour les classes avec polymorphisme ce dernier est une fonction linéaire (n).

Réponse 10 :

Pour un choix judicieux, on prendra l'algorithme des classes avec héritage parce que ce dernier à un temps d'exécution tellement court par rapport à l'autre algorithme basé sur le polymorphisme. Un temps d'exécution logarithmique est meilleur par rapport au temps linéaire.