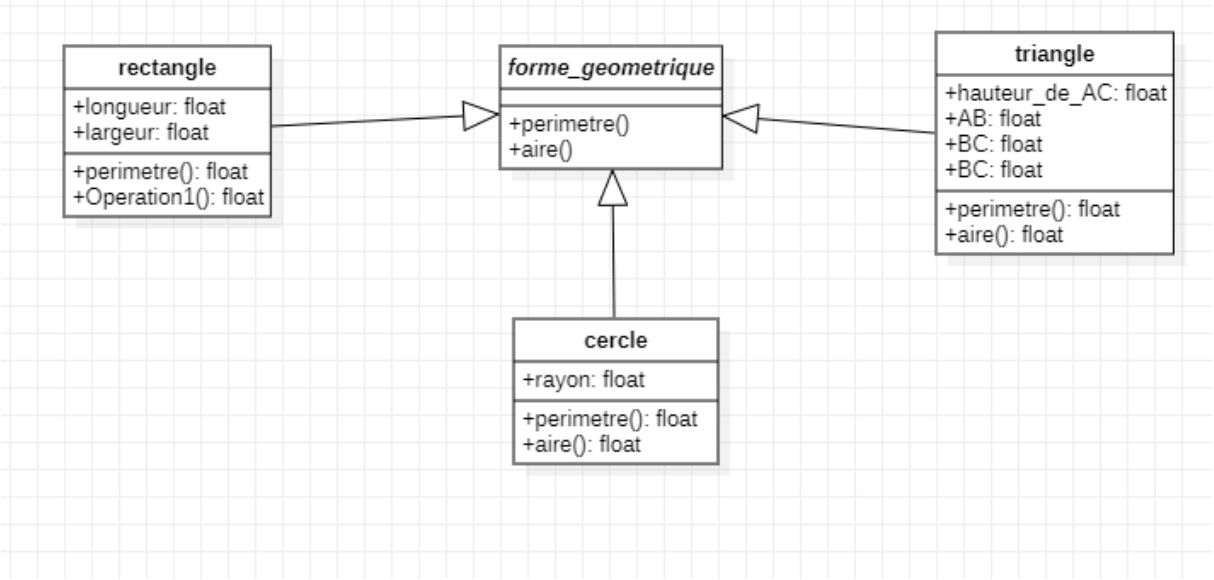
PROJET D'ALGO

ETUDIANTS(ES) :

DZAPILI MITANO JOSEPH(2GEI) , MBONGA MABIALA GLOIRE(2GC)

RESOLUTION PROJET 1 :

Commençons par représenter le diagramme UML de notre projet(REPONSE A LA QUESTION 3) :



Question1

Nous nous sommes décidés que la classe ***forme géométrique*** dans notre projet soit une classe abstraite avec l'implémentation qui suit :

```
from math import *
from abc import ABCMeta, abstractmethod

class forme_geometrique(metaclass=ABCMeta):

    @abstractmethod

    def aire(self):
        return
    def perimetre(self):
        return
class rectangle(forme_geometrique):
    def __init__(self,hauteur,largeur):# ce sont des attributs
        self.hauteur=hauteur
        self.largeur=largeur
    def aire(self):
        return self.hauteur*self.largeur

    def perimetre(self):
        return (self.hauteur+self.largeur)*2
```

QUESTION2

Nous avons dans le diagramme UML représenter plus haut l'organisation de nos classes et dans les images ci-dessous nous pouvons voir leur implémentation :

1. **Classe rectangle :**

```
class rectangle(forme_geometrique):
    def __init__(self,hauteur,largeur):# ce sont des attributs
        self.hauteur=hauteur
        self.largeur=largeur
    def aire(self):
        return self.hauteur*self.largeur

    def perimetre(self):
        return (self.hauteur+self.largeur)*2
```

2. **Classe cercle :**

```
class cercle(forme_geometrique):
    def __init__(self, rayon):
        self.rayon=rayon
    def aire(self):
        return pi*self.rayon**2
    def perimetre(self):
        return 2*pi*self.rayon
```

3. **Classe triangle :**

```
class triangle(forme_geometrique):
    def __init__(self,AB,BC,AC,hauteur_de_AC):
        self.AB=AB
        self.BC=BC
        self.AC=AC
        self.hauteur_de_AC=hauteur_de_AC
    def aire(self):
        return self.hauteur_de_AC*self.AC/2
    def perimetre (self):
        return self.AB+self.BC+self.AC
```

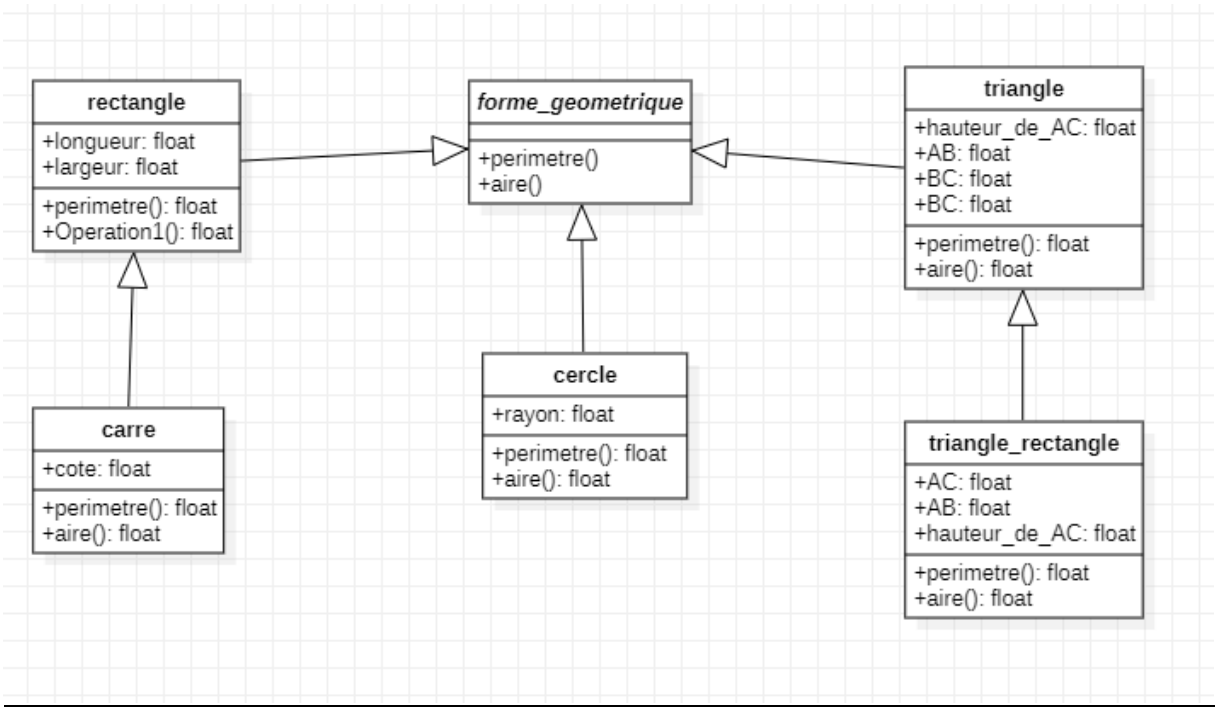
QUESTION3 (cfr le diagramme UML**)**

QUESTION4

Nous savons que le carré est un cas rectangle où la largeur équivaut à la longueur,

La classe ***carrée*** héritera donc de la classe ***rectangle*** et le ***triangle rectangle*** héritera de la classe ***triangle***.

LE DIAGRAMME UML DEVIENT :



QUESTION5

Nous avons crée une classe « NEW » et nous y avons crée un objet de type rectangle et avec celui-ci nous avons calculé la surface et le périmètre.

l'implémentation de la classe est représenté ci dessous :

```
from tp_algo import*

class NEW :
    Rect= rectangle(24,58)
    print("la surface vaut :",Rect.aire())
    print("le perimetre vaut :", Rect.perimetre())
```

Le résultat donne :

```
========== RESTART: C:/Users/Joseph Dzapili/Documents/monscript/NEW.py =========
la surface vaut : 1392
le perimetre vaut : 164
```

QUESTION6

Nous avons implémenté 6 classes, et le nombre de classes est le minimum car il est toujours possible d'utiliser encore d'autre classe pour avoir le même résultat.

QUESTION7 :

Notre projet Forme géométrique est enregistré au nom de << tp_algo>> , créons ensuite une batterie test :

```
# -*- coding: utf-8 -*-
"""
Created on Tue Dec 27 16:14:28 2022

@author: Gloire MBONGA AND JOSEPH DZAPILI
"""
#orienter objet = heritage
from tp_algo import *
fre=triangle(5,5,6,4)
print("la surface vaut :", fre.aire())
print("le perimetre vaut :", fre.perimetre())
RECT=rectangle(28,60)
print("la surface vaut : ", RECT.aire())
print("le perimetre vaut :", RECT.perimetre())
CAR=carre(25)
print("la surface du carre vaut :", CAR.aire())
print("le perimetre du carre vaut :",CAR.perimetre())
CIRCLE=cercle(48)
print("la surface du cercle vaut {:.2f}".format(CIRCLE.aire()))
print("le perimetre du cercle vaut {:.2f}:".format(CIRCLE.perimetre()))
```

Le resultat vaut :

```
==== RESTART: C:/Users/Joseph Dzapili/Documents/monscript/batterie test 1.py ===
la surface vaut : 12.0
le perimetre vaut : 16
la surface vaut :  1680
le perimetre vaut : 176
la surface du carre vaut : 625
le perimetre du carre vaut : 100
la surface du cercle vaut 7238.23
le perimetre du cercle vaut 301.59:
>>>
```

QUESTION 8 :

le problème est de refaire le même programme mais avec une technologie différentes, « le polymorphisme » nous pouvons montrer le diagramme UML basé sur cette technologie :

DIAGRAMME UML :

Et l'implémentation vaut :

```
import math
class rectangle:
    def __init__(self,longueur,largeur):
        self.longueur=longueur
        self.largeur=largeur
    def aire(self):
        return self.longueur*self.largeur
    def perimetre (self):
        return (self.longueur+self.largeur)*2

class cercle:
    def __init__(self,rayon):
        self.rayon=rayon
    def aire(self):
        return math.pi*self.rayon**2
    def perimetre (self):
        return 2*math.pi*self.rayon
class triangle:
    def __init__(self,AB,BC,AC,hauteur_de_AC):
        self.AB=AB
        self.BC=BC
        self.AC=AC
        self.hauteur_de_AC=hauteur_de_AC
    def aire(self):
        return self.hauteur_de_AC*self.AC/2
    def perimetre (self):
        return self.AB+self.BC+self.AC
class carre:
     def __init__(self,cote):
        self.cote=cote
     def aire(self):
        return self.cote**2
     def perimetre(self):
        return self.cote*4
class triangle_rectangle:
    def __init__(self,AB,AC,hauteur_de_AC):
        self.AB=AB
        self.AC=AC
        self.hauteur_de_AC=hauteur_de_AC
    def aire(self):
        return self.AC*self.hauteur_de_AC/2
    def perimetre (self):
        return self.AB+self.hauteur_de_AC+self.AC
```

La batterie de test donne :

```
from TP_algo2 import*

class forme_geometrique:


    print("menu : ")
    print(" 1° rectangle \n 2° triangle \n 3° cercle \n 4° carré \n 5° triangle_rectangle")
    print("veuillez effectuer votre choix pour le test du calcul du perimetre et de la surface ")
    try:

        choix=int(input("choix : "))

        while (choix !=1 and choix !=2 and choix !=3 and choix !=4 and choix !=5 ):
            print("veuillez faire le bon choix")
            choix=int(input("choix : "))
    except:
        print("entrer des valeurs numerique")



    if choix==1:
        print("le perimetre et la aire du rectangle donne")
        rect=rectangle(5,7)
        print("la aire du rectangle : ",rect.aire())
        print("le perimetree du rectangle : ",rect.perimetre())

    elif choix==2  :
        print("le perimetre et la aire du triangle donne")

        tri=triangle(6,8,5,4)
        print("la aire du triangle est :  ",tri.aire())
        print("le perimetre du triangle est : ",tri.perimetre())
    elif choix==3  :
        print("le perimetre et la surface du cercle donne")
        cer=cercle(7)
        print("la aire du cercle : ",cer.aire())
        print("le perimetre du cercle : ",cer.perimetre())

    elif choix==4:
        print("la aire et du perimetre d'un carré donne")
        car=carre(5)
        print("la aire du carré est :  ",car.aire())
        print("le perimetre du carré est : ",car.perimetre())
```

```
    elif choix==4:
        print("la aire et du perimetre d'un carré donne")
        car=carre(5)
        print("la aire du carré est :  ",car.aire())
        print("le perimetre du carré est : ",car.perimetre())

    elif choix==5  :
        print("le perimetre et la aire du triangle_rectangle donne")

        triRe=triangle_rectangle(7,4,5)
        print("la aire du triangle est :  ",triRe.aire())
        print("le perimetre du rectangle est : ",triRe.perimetre())
```

Le résultat donne :

```
==== RESTART: C:\Users\Joseph Dzapili\Documents\monscript\batterie test2.py ====
menu :
 1° rectangle
 2° triangle
 3° cercle
 4° carré
 5° triangle_rectangle
veuillez effectuer votre choix pour le test du calcul du perimetre et de la surf
ace
choix : 1
le perimetre et la aire du rectangle donne
la aire du rectangle :  35
le perimetree du rectangle :  24
```

QUESTION 9 :

Calculons le temps d'exécution de nos deux programmes fait avec les deux technologies différentes, l'héritage dans le premier, et le polymorphisme dans le deuxième :

1) Pour le test avec héritage :

```
# -*- coding: utf-8 -*-
"""
Created on Tue Dec 27 16:14:28 2022

@author: Gloire MBONGA AND JOSEPH DZAPILI
"""
#orienter objet = heritage
from time import*
from tp_algo import *
debut=time()
fre=triangle(5,5,6,4)
print("la surface vaut :", fre.aire())
print("le perimetre vaut :", fre.perimetre())
RECT=rectangle(28,60)
print("la surface vaut : ", RECT.aire())
print("le perimetre vaut :", RECT.perimetre())
CAR=carre(25)
print("la surface du carre vaut :", CAR.aire())
print("le perimetre du carre vaut :",CAR.perimetre())
CIRCLE=cercle(48)
print("la surface du cercle vaut {:.2f}".format(CIRCLE.aire()))
print("le perimetre du cercle vaut {:.2f}:".format(CIRCLE.perimetre()))
fin=time()
temps=fin-debut

print("le temps d'execution est {:.2f}".format(temps))
```

Le résultat est le suivant :

```
==== RESTART: C:\Users\Joseph Dzapili\Documents\monscript\batterie test 1.py ===
la surface vaut : 12.0
le perimetre vaut : 16
la surface vaut :  1680
le perimetre vaut : 176
la surface du carre vaut : 625
le perimetre du carre vaut : 100
la surface du cercle vaut 7238.23
le perimetre du cercle vaut 301.59:
le temps d'execution est 0.21
```

On a donc un temps de 0.21 sec dans le cas de l'héritage.

2) Pour le test avec le polymorphisme :

```
from TP_algo2 import*

class forme_geometrique:
        debut=time()

        print("le perimetre et la aire du rectangle donne")
        rect=rectangle(5,7)
        print("la aire du rectangle : ",rect.aire())
        print("le perimetree du rectangle : ",rect.perimetre())


        print("le perimetre et la aire du triangle donne")

        tri=triangle(6,8,5,4)
        print("la aire du triangle est :  ",tri.aire())
        print("le perimetre du triangle est : ",tri.perimetre())

        print("le perimetre et la surface du cercle donne")
        cer=cercle(7)
        print("la aire du cercle : ",cer.aire())
        print("le perimetre du cercle : ",cer.perimetre())


        print("la aire et du perimetre d'un carré donne")
        car=carre(5)
        print("la aire du carré est :  ",car.aire())
        print("le perimetre du carré est : ",car.perimetre())


        print("le perimetre et la aire du triangle_rectangle donne")

        triRe=triangle_rectangle(7,4,5)
        print("la aire du triangle est :  ",triRe.aire())
        print("le perimetre du rectangle est : ",triRe.perimetre())

        fin=time()
        temps= fin-debut
        print("le temps  d'execution est {:.2f}".format(temps))
```

Le résultat vaut :

```
== RESTART: C:\Users\Joseph Dzapili\Documents\monscript\batterie test2 tmps.py =
le perimetre et la aire du rectangle donne
la aire du rectangle :  35
le perimetree du rectangle :  24
le perimetre et la aire du triangle donne
la aire du triangle est :   10.0
le perimetre du triangle est :  19
le perimetre et la surface du cercle donne
la aire du cercle :  153.93804002589985
le perimetre du cercle :  43.982297150257104
la aire et du perimetre d'un carré donne
la aire du carré est :   25
le perimetre du carré est :  20
le perimetre et la aire du triangle_rectangle donne
la aire du triangle est :   10.0
le perimetre du rectangle est :  16
le temps  d'execution est 0.36
>>>
```

Ceci nous donne un temps d'exécution de 0.36 sec.

QUESTION 10 :

Nous voyons donc qu'il est mieux d'utiliser l'héritage car le temps d'exécution est moindre