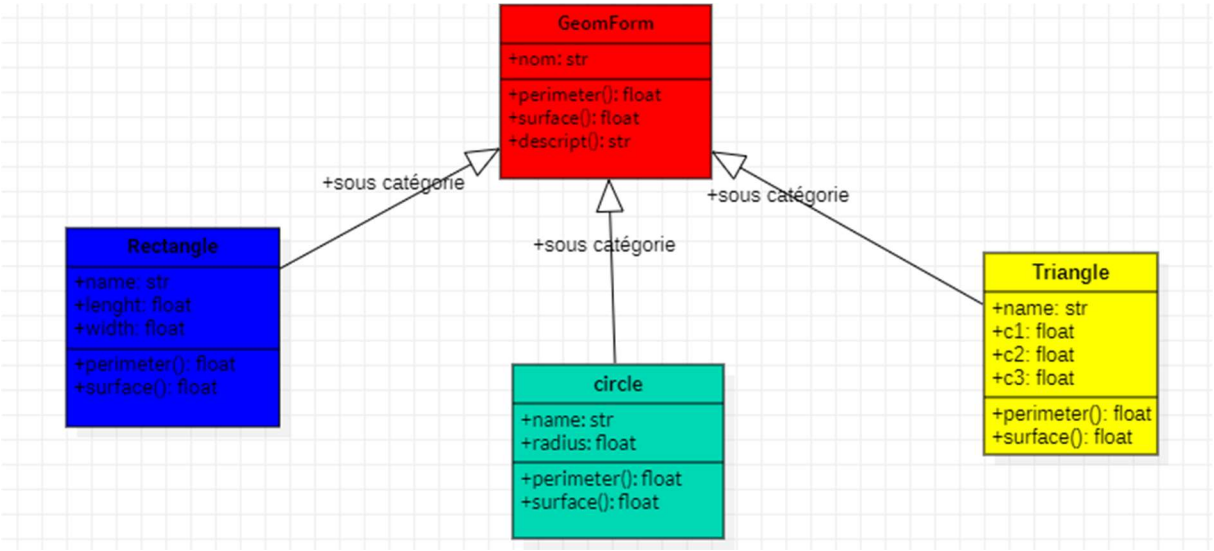
Solution du projet

Réalisation du diagramme UML au préalable



Question 1

Classe de base valable pour toute forme géométrique

```python
from abc import ABCMeta, abstractmethod
from math import pi, sqrt

""" QUESTION NUMERO 1 """

class GeomForm(metaclass = ABCMeta): # Creation de la classe de base abstraite
    @abstractmethod
    def perimeter(): # Definition de la fonction perimeter dont le code n'est pas defini, elle devient donc abstraite
        pass

    @abstractmethod
    def surface(): # Definition de la fonction surface dont le code n'est pas defini, elle devient donc abstraite
        pass

    def descript(self):
        print("Soit {}\nLe Perimetre est : {}\nLa Surface est : {}".format(self.name, self.perimeter(), self.surface()))
```

Question 2

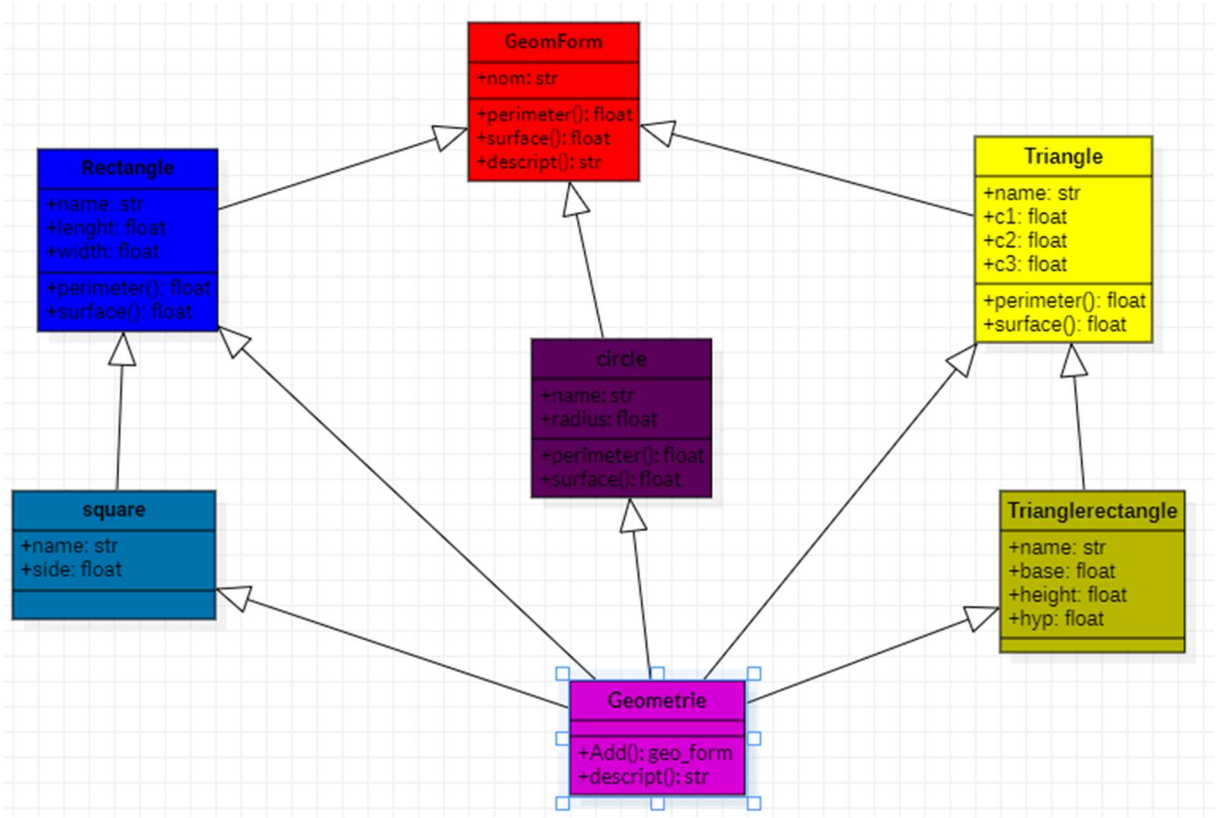
Trois classes qui héritent de la classe de base

```
class Rectangle(GeomForm): # Creation de la classe Rectangle heritant de la classe GeomForm
    try:
        def __init__(self,name, length, width):
            self.name = name
            self.length = length
            self.width = width
        def perimeter(self):
            return 2*self.length + 2*self.width

        def surface(self):
            return self.length*self.width
    except:
        print("ERREUR SUR LE PARAMETRE INSERER .")

class Circle(GeomForm):  #Creation de la classe Circle heritant de la classe GeomForm
    try:
        def __init__(self, name, radius):
            self.name = name
            self.radius = radius
        def perimeter(self):
            return 2*pi*self.radius
        def surface(self):
            return  pi*(self.radius**2)
    except:
        print("ERREUR SUR LE PARAMETRE INSERER . ")
```

```
class Triangle(GeomForm): # Creation de la classe Triangle heritant de la classe GeomForm
    try:
        def __init__(self,name, c1,c2,c3):
            self.name = name
            self.c2 = c2
            self.c1 = c1
            self.c3 = c3
        def perimeter(self):
            return self.c2 + self.c1 + self.c3

        def surface(self):
            k = self.perimeter()/2
            area = sqrt(k*(k - self.c1)*(k - self.c2)*(k - self.c3))
            area = area.real
            return area
    except:
        print("ERREUR SUR LE PARAMETRE INSERER .")
```

Question 3

Mise à jour du diagramme UML




```python
class Square(Rectangle): # Creation de la classe Carre heritant de la classe Rectangle
    try:
        def __init__(self,name, side):
            Rectangle.__init__(self, name, side, side)
    except:
        print(" ERREUR SUR LE PARAMETRE INSERER . ")

class TriangleRectangle(Triangle): # Creation de la classe TriangleRectangle heritant de la classe Triangle
    try:
        def __init__(self,name, base, height):
            hyp = sqrt(base**2+height**2)
            Triangle.__init__(self, name, base, height, hyp)
    except:
        print(" ERREUR SUR LE PARAMETRE INSERER . ")
```

Question 4

Classe qui exploite les classes des formes géométriques

```python
class Geometrie(): # Creation de la classe qui exploite les classes des diffentes formes geo pour fournir le perimetre et
    try:
        def __init__(self):
            self.gGeo_rep = []
        def add(self, fig):
            self.gGeo_rep.append(fig)
        def descript(self):
            for g in self.gGeo_rep:
                print("Soit {}\nLe Perimetre est : {}\nLa Surface est : {}".format(g.name, g.perimeter(), g.surface()))
    except:
        print(" ERREUR SUR LE PARAMETRE INSERER . ")
```

Question 5

Diagramme des classes UML

Cfr question 3

Question 6

On a développé 7 classes au total.

7 n'est pas le nombre de classe minimum possible car on peut se passer de la classe de TriangleRectangle et de carré et avoir le résultat escompté avec la classe rectangle et triangle.

Question 7

```
# BATTERIE DE TEST IMPLEMENTATION 1

from module_impl1 import Rectangle, Circle, Triangle, Square, TriangleRectangle, Geometrie
if __name__ == '__main__':
    print(" A PARTIR DES DIFFERENTES CLASSES DES FORMES GEOMETRIQUES :")
    print()
    try:
        rectangle = Rectangle(" Un Rectangle_1 donné dont :", 14, 7)
        cercle = Circle("Un Cercle_1 donné dont :", 14)
        triangle = Triangle("Un Triangle_1 donné dont :", 9, 6, 7)
        carre = Square("Un Carré_1 donné dont :", 6)
        t_rect = TriangleRectangle("Un Triangle Rectangle donné dont :", 5, 7)

        rectangle.descript()
        print("")
        cercle.descript()
        print("")
        triangle.descript()
        print("")
        carre.descript()
        print("")
        t_rect.descript()
        print("")
    except Exception:
        print("ERREUR SUR LE PARAMETRE INSERER .")

    print()
    print()
    print("A PARTIR DE LA CLASSE DE BASE : ")
    print()
    figA = Geometrie()
    figB = Geometrie()
    figC = Geometrie()
    figD = Geometrie()
    figE = Geometrie()
```

```
figA.add(Rectangle("Un Rectangle_2 donné dont :", 14, 5))
figB.add(Circle("Cercle_2 donné dont :", 6))
figC.add(Triangle("Triangle_2 donné dont :", 11, 6, 7))
figD.add(Square("Carré_2 donné dont :", 13))
figE.add(TriangleRectangle("Triangle Rectangle_2 donné dont :", 15, 7))

figA.descript()
print("")
figB.descript()
print("")
figC.descript()
print("")
figD.descript()
print("")
figE.descript()


try:
    pass

except Exception:
    print("ERREUR SUR LE PARAMETRE INSERER .")
```

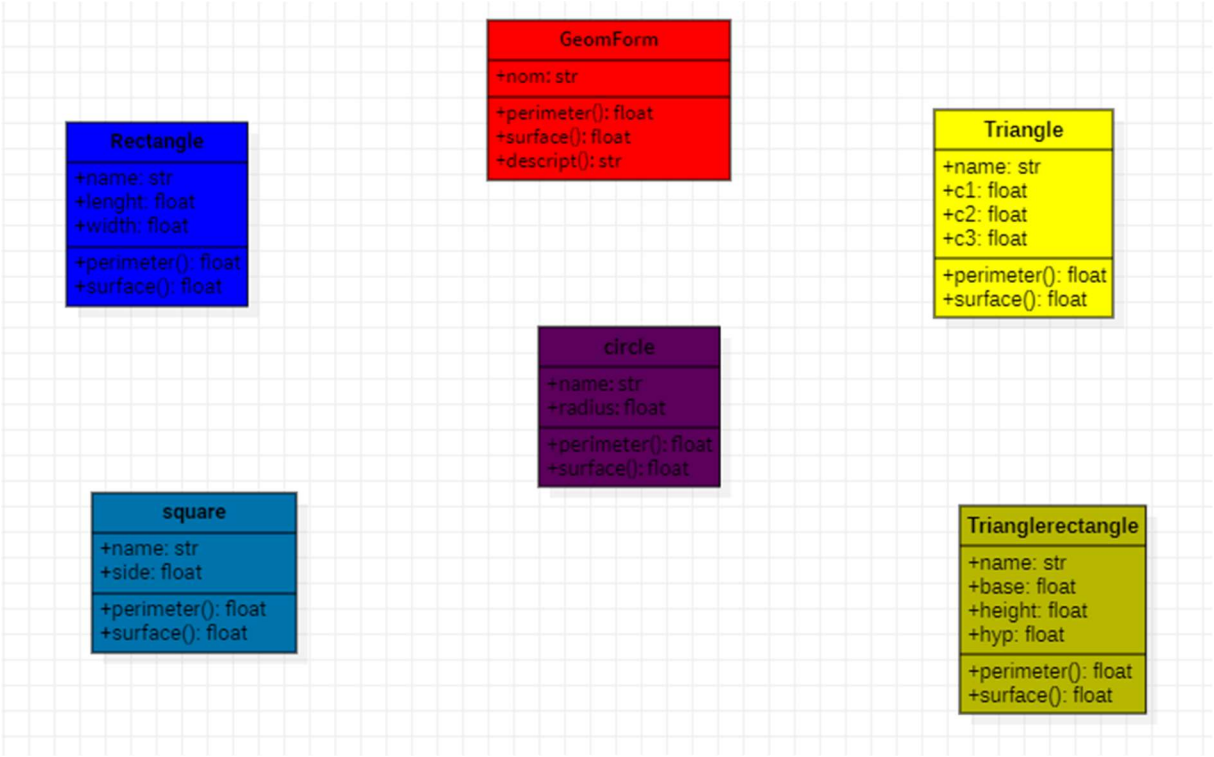
Son résultat

```
 A PARTIR DES DIFFERENTES CLASSES DES FORMES GEOMETRIQUES

Soit  Un Rectangle_1 donné dont :
Le Perimetre est : 42
La Surface est : 98

Soit Un Cercle_1 donné dont :
Le Perimetre est : 87.96459430051421
La Surface est : 615.7521601035994

Soit Un Triangle_1 donné dont :
Le Perimetre est : 22
La Surface est : 20.97617696340303

Soit Un Carré_1 donné dont :
Le Perimetre est : 24
La Surface est : 36

Soit Un Triangle Rectangle donné dont :
Le Perimetre est : 20.602325267042627
La Surface est : 17.5



A PARTIR DE LA CLASSE DE BASE :
```

```
A PARTIR DE LA CLASSE DE BASE :

Soit Un Rectangle_2 donné dont :
Le Perimetre est : 38
La Surface est : 70

Soit Cercle_2 donné dont :
Le Perimetre est : 37.69911184307752
La Surface est : 113.09733552923255

Soit Triangle_2 donné dont :
Le Perimetre est : 24
La Surface est : 18.973665961010276

Soit Carré_2 donné dont :
Le Perimetre est : 52
La Surface est : 169

Soit Triangle Rectangle_2 donné dont :
Le Perimetre est : 38.55294535724685
La Surface est : 52.50000000000001
```

Question 8

Réalisation du diagramme UML au préalable



module_impl2 (méthode de classe classique**)**

```python
from abc import ABCMeta, abstractmethod
from math import pi, sqrt
class GeomForm(metaclass = ABCMeta): # Creation de la classe de base abstraite
    @abstractmethod
    def perimeter(): # Definition de la fonction perimeter dont le code n'est pas defini, elle devient donc abstraite
        pass

    @abstractmethod
    def surface(): # Definition de la fonction surface dont le code n'est pas defini, elle devient donc abstraite
        pass

    def descript(self):
        print("Soit {}\nLe Perimetre est : {}\nLa Surface est : {}".format(self.name, self.perimeter(), self.surface()))
class rectangle:
    def __init__(self,height,width):
        self.height=height
        self.width=width

    def surface(self):
        return self.height*self.width

    def perimetre(self):

        return (self.height+self.width)*2
import math
class cercle:
    def __init__(self,radius):
        self.radius=radius


    def surface(self):
        return math.pi*self.radius**2
```

```python
class triangle:
    def __init__(self,height,c1,c2,c3):
        self.height=height
        self.c1=c1
        self.c2=c2
        self.c3=c3

    def surface(self):
        return self.height*self.c1/2

    def perimetre(self):
        return (self.c1+self.c2+self.c3)
class carre:
    def __init__(self,side):
        self.side=side


    def surface(self):
        return self.side**2

    def perimetre(self):
        return self.side*4



class triangleRectangle:
    def __init__(self,height,c1,hypotenus):
        self.height=height
        self.c1=c1
        self.hypotenus=hypotenus
```

```
  def surface(self):
      return self.height*self.c1/2

  def perimetre(self):
      return (self.c1+self.height+self.hypotenus)
```

Battérie de test 2

```
from module_impl2 import*
import random

import time as tm
class formegeometrique:

    start=tm.time()

    print("1) CALCUL DU PERIMETRE ET DE LA SURFACE DU RECTANGLE. ")
    print("-----------------------------------------------------")
    a=float(random.randint(1,50))
    b=float(random.randint(1,50))

    rect=rectangle(a,b)
    print(" - LA SURFACE DU RECTANGLE EST : ",rect.surface())
    print(" - LE PERIMETRE DU RECTANGLE EST : ",rect.perimetre())
    print("")


    print("2) CALCUL DU PERIMETRE ET DE LA SURFACE DU TRIANGLE. ")
    print("-----------------------------------------------------")
    a=float(random.randint(1,50))
    b=float(random.randint(1,50))
    c=float(random.randint(1,50))
    d=float(random.randint(1,50))
    tri=triangle(a,b,c,d)
    print(" - LA SURFACE DU TRIANGLE EST :  ",tri.surface())
    print(" - LE PERIMETRE DU TRIANGLE EST : ",tri.perimetre())
    print("")
```

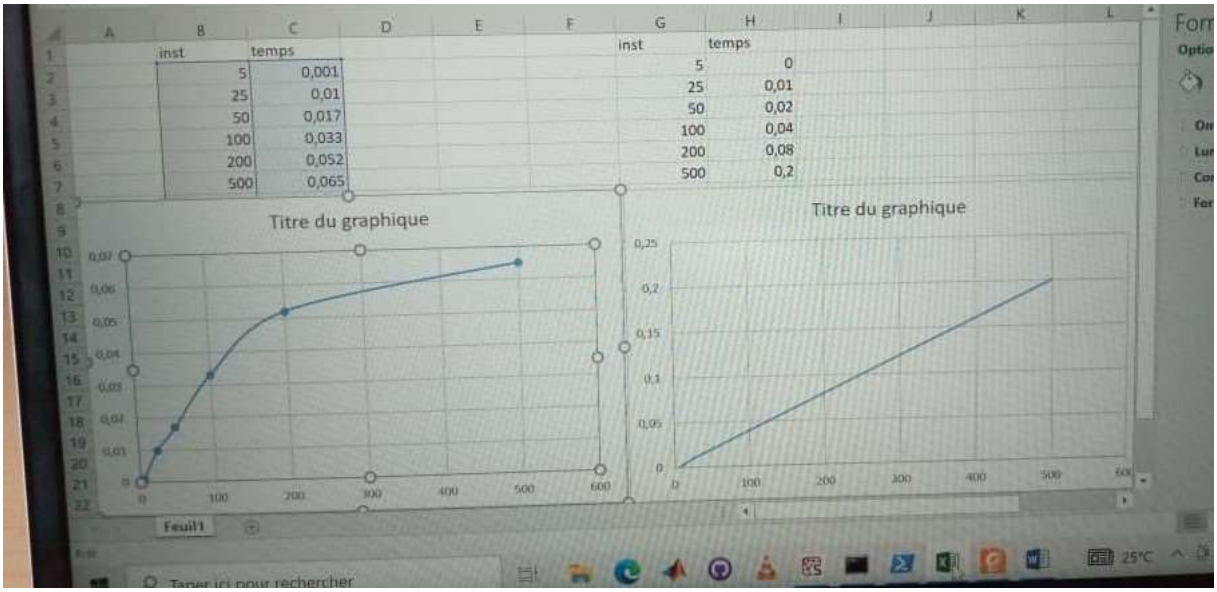
```
print("3) CALCUL DE LA CIRCONFERENCE ET DE LA SURFACE DU CERCLE. ")
print("-------------------------------------------------------------")
a=float(random.randint(1,50))


cer=cercle(a)
print(" - LA SURFACE DU CERCLE EST  : ",cer.surface())
print(" - LA CIRCONFERENCE DU CERCLE EST : ",cer.perimetre())
print("")




print("4) CALCUL DE LA SURFACE ET DU PERIMETRE DU CARRE. ")
print("-----------------------------------------------------")
a=float(random.randint(1,50))


car=carre(a)
print(" - LA SURFACE DU CARREE EST  :  ",car.surface())
print(" - LE PERIMETRE DU CARRE EST : ",car.perimetre())
print("")

print("5) CALCUL DU PERIMETRE ET DE LA SURFACE DU TRIANGLE RECTANGLE. ")
print("-------------------------------------------------------------")
a=float(random.randint(1,50))
b=float(random.randint(1,50))
c=float(random.randint(1,50))


tri=triangleRectangle(a,b,c)
print(" - LA SURFACE DU TRIANGLE RECTANGLE EST :  ",tri.surface())
print(" - LE PERIMETRE DU TRIANGLE RECTANGLE EST : ",tri.perimetre())
```

```
 tri=triangleRectangle(a,b,c)
 print(" - LA SURFACE DU TRIANGLE RECTANGLE EST :  ",tri.surface())
 print(" - LE PERIMETRE DU TRIANGLE RECTANGLE EST : ",tri.perimetre())
 print("")

 end=tm.time()

 temps= end-start


 print(" LE TEMPS D'EXECUTION EST : ", temps)
```

Son résultat

```
1) CALCUL DU PERIMETRE ET DE LA SURFACE DU RECTANGLE.
-----------------------------------------------------
 - LA SURFACE DU RECTANGLE EST :  296.0
 - LE PERIMETRE DU RECTANGLE EST :  90.0

2) CALCUL DU PERIMETRE ET DE LA SURFACE DU TRIANGLE.
-----------------------------------------------------
 - LA SURFACE DU TRIANGLE EST :   192.0
 - LE PERIMETRE DU TRIANGLE EST :  77.0

3) CALCUL DE LA CIRCONFERENCE ET DE LA SURFACE DU CERCLE.
---------------------------------------------------------
 - LA SURFACE DU CERCLE EST  :  2827.4333882308138
 - LA CIRCONFERENCE DU CERCLE EST :  188.49555921538757

4) CALCUL DE LA SURFACE ET DU PERIMETRE DU CARRE.
-------------------------------------------------
 - LA SURFACE DU CARREE EST  :   289.0
 - LE PERIMETRE DU CARRE EST :  68.0

5) CALCUL DU PERIMETRE ET DE LA SURFACE DU TRIANGLE RECTANGLE.
--------------------------------------------------------------
 - LA SURFACE DU TRIANGLE RECTANGLE EST :   252.0
 - LE PERIMETRE DU TRIANGLE RECTANGLE EST :  68.0

 LE TEMPS D'EXECUTION EST :  0.015594005584716797
```

Question 9



Le temps d'exécution avec le concept d'**héritage** est inférieur au temps d'exécution de la méthode de **classe** classique

Question 10

Le concept de l'héritage permet d'avoir moins de ligne de code et un temps d'exécution plus optimal par rapport à la méthode de classe classique, autrement dit c'est un choix judicieux.