Membres du groupe 25 :

- ASIMWE NDRUUDJO GLOIRE 2GC
- IMANI UKURANGO CHRISTIAN 2GC
- HESHIMA MASIMIKE DAVID 2GM

FICHES DES REPONSES AUX DIFFERENTES QUESTIONS DU PROJET 1

1)

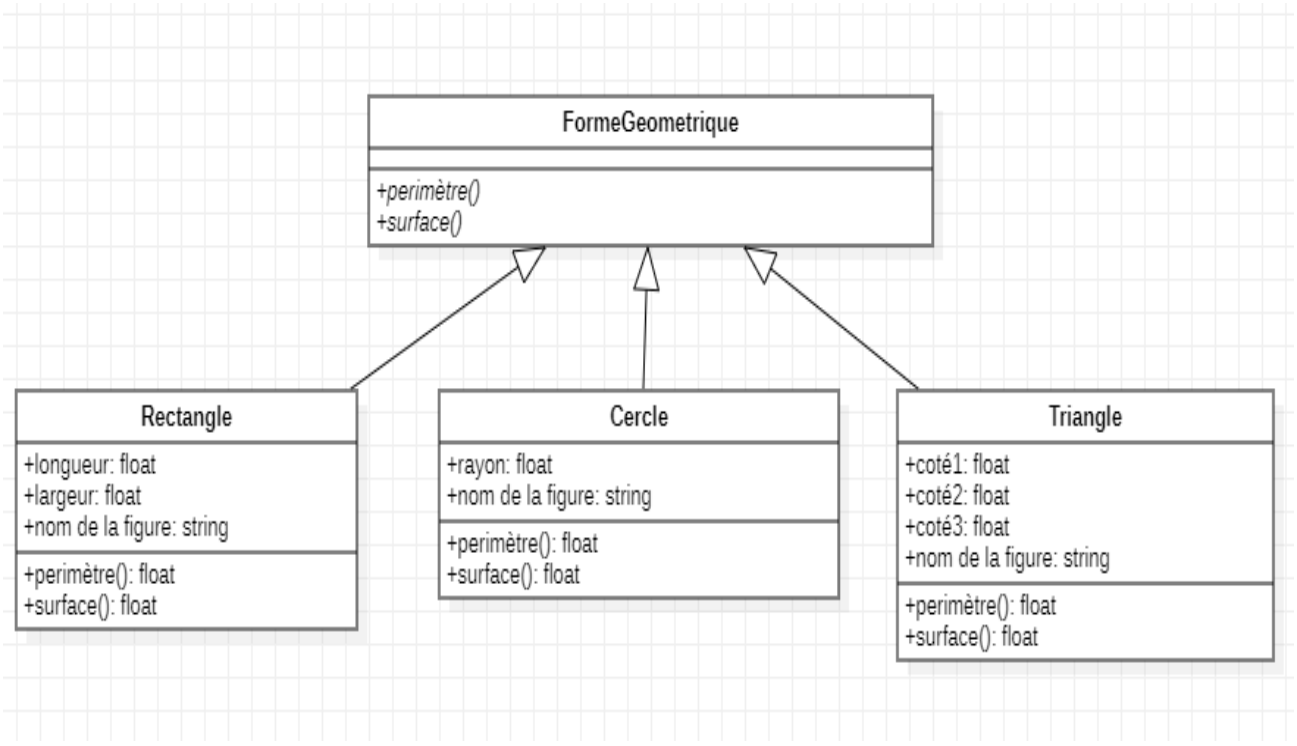
```python
from abc import ABC, abstractmethod
from math import pi, sqrt

class FormeGeometrique(ABC):

    @abstractmethod
    def perimetre(self):
        pass

    @abstractmethod
    def surface(self):
        pass
```

2)

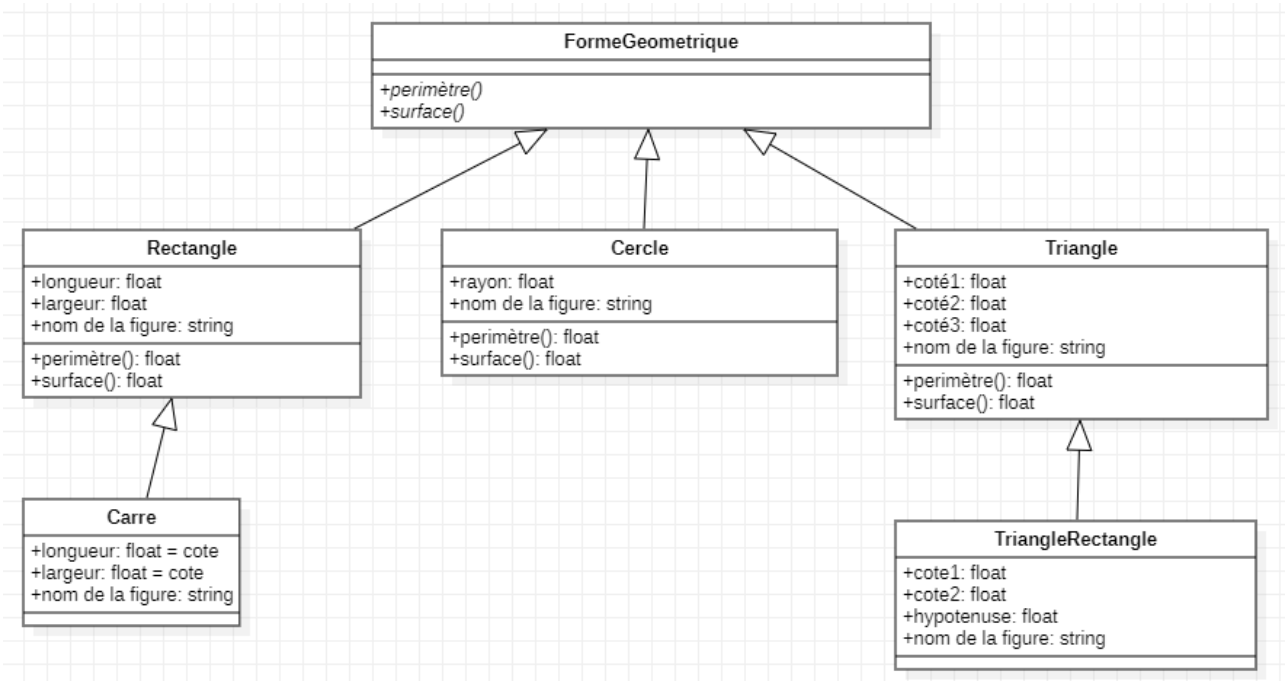
```python
class Rectangle(FormeGeometrique):
    """Permet de calculer le périmetre et la surface d'un rectangle"""
    def __init__(self, longueur, largeur):
        self._longueur = longueur
        self._larguer = largeur
        self._nom = "Rectangle"

    def perimetre(self):
        return 2*(self._longueur + self._larguer)

    def surface(self):
        return self._longueur*self._larguer

class Cercle(FormeGeometrique):
    """Permet de calculer le périmètre et la surface du cercle"""
    def __init__(self, rayon):
        self._rayon = rayon
        self._nom = "Cercle"


    def perimetre(self):
        return 2*(pi*self._rayon)
```

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

class Triangle(FormeGeometrique):
    """Permet de calculer le périmètre et la surface d'un triangle"""
    def __init__(self, cote_1, cote_2, cote_3):
        self._cote1 = cote_1
        self._cote2 = cote_2
        self._cote3 = cote_3
        self._nom = "Triangle"

    def perimetre(self):
        return self._cote1 + self._cote2 + self._cote3

    def surface(self):
        perimetre = self._cote1 + self._cote2 + self._cote3
        return sqrt((perimetre/2)*((perimetre/2) - self._cote1)
        *((perimetre/2) - self._cote2)*((perimetre/2) - self._cote3))
```

3)



4)




```python
class Carre(Rectangle):
    """Permet de calculer le périmètre et la surface d'un Cercle"""
    def __init__(self, cote):
        self._longueur = cote
        self._larguer = cote
        self._nom = "Carré"

class TriangleRectangle(Triangle):
    """Permet de calculer le périmètre et la surface d'un triangle
rectangle"""
    def __init__(self, cote_1, cote_2):
        self._cote1 = cote_1
        self._cote2 = cote_2
        self._cote3 = sqrt((self._cote1**2) + (self._cote2**2))
        self._nom = "Triangle Réctangle"
```

5)

```python
class PerimetreSurface:
    """Permet de récupérer le périmètre et la surface d'une forme
géométrique"""
    def __init__(self, *formeGeometrique:FormeGeometrique):
        self._formeGeo = formeGeometrique
    def get_perimetreSurface(self):
        return [(i.perimetre(),i.surface(), i._nom )for i in self._formeGeo]
```

6) Dans ce projet, nous avons utilisé 7 classes.

Oui, parce que chaque figure géométrique traité dans le cadre de ce projet est représentée par une classe bien précise en dehors de la classe mère et celle qui affiche les périmètres et les surfaces des différentes figures géométriques.

7)

Le module des classes développées est :

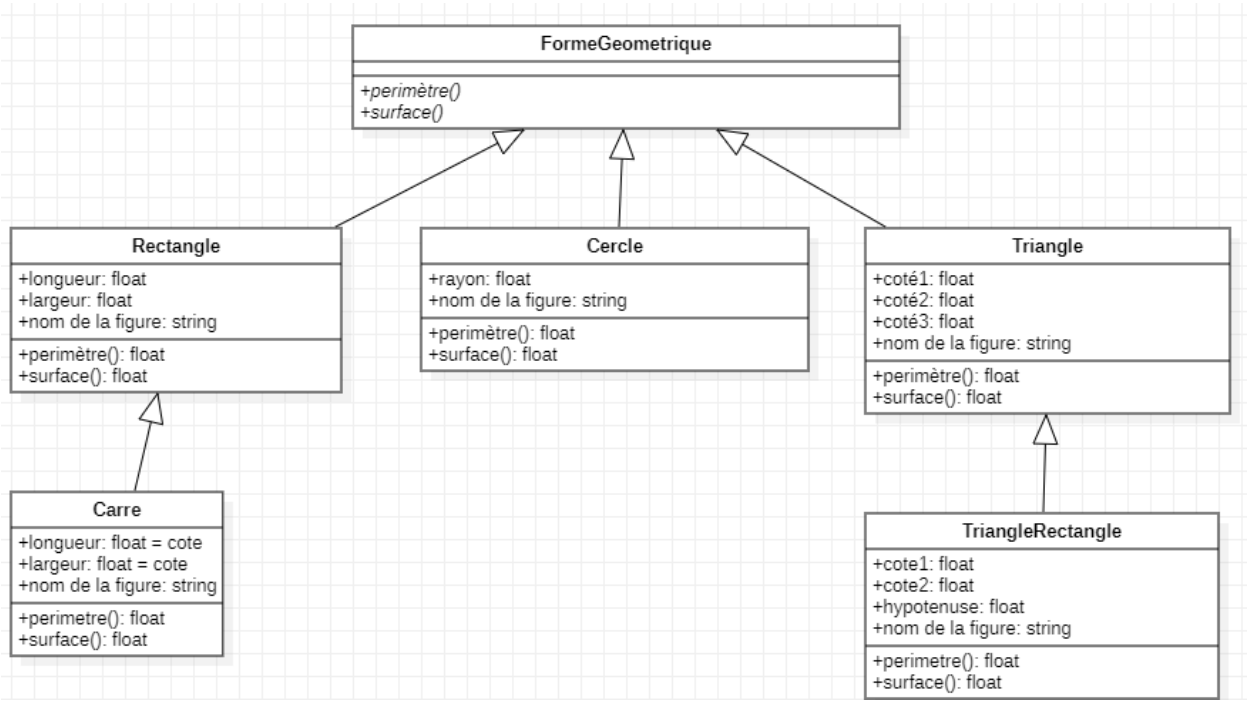
```
from abc import ABC, abstractmethod
from math import pi, sqrt

class FormeGeometrique(ABC):

    @abstractmethod
    def perimetre(self):
        pass

    @abstractmethod
    def surface(self):
        pass

class Rectangle(FormeGeometrique):
    """Permet de calculer le périmetre et la surface d'un rectangle"""
    def __init__(self, longueur, largeur):
        self._longueur = longueur
        self._larguer = largeur
        self._nom = "Rectangle"

    def perimetre(self):
        return 2*(self._longueur + self._larguer)

    def surface(self):
        return self._longueur*self._larguer

class Cercle(FormeGeometrique):
    """Permet de calculer le périmètre ou la circonférence et la surface du
cercle"""
    def __init__(self, rayon):
        self._rayon = rayon
        self._nom = "Cercle"

    def perimetre(self):
        return 2*(pi*self._rayon)

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

```
class Triangle(FormeGeometrique):
    """Permet de calculer le périmètre et la surface d'un triangle"""
    def __init__(self, cote_1, cote_2, cote_3):
        self._cote1 = cote_1
        self._cote2 = cote_2
        self._cote3 = cote_3
        self._nom = "Triangle"

    def perimetre(self):
        return self._cote1 + self._cote2 + self._cote3

    def surface(self):
        perimetre = self._cote1 + self._cote2 + self._cote3
        return sqrt((perimetre/2)*((perimetre/2) - self._cote1)
        *((perimetre/2) - self._cote2)*((perimetre/2) - self._cote3))

class Carre(Rectangle):
    """Permet de calculer le périmètre et la surface d'un Cercle"""
    def __init__(self, cote):
        self._longueur = cote
        self._larguer = cote
        self._nom = "Carré"

class TriangleRectangle(Triangle):
    """Permet de calculer le périmètre et la surface d'un triangle
rectangle"""
    def __init__(self, cote_1, cote_2):
        self._cote1 = cote_1
        self._cote2 = cote_2
        self._cote3 = sqrt((self._cote1**2) + (self._cote2**2))
        self._nom = "Triangle Réctangle"

class PerimetreSurface:
    """Permet de récupérer le périmètre et la surface d'une forme
géométrique"""
    def __init__(self, *formeGeometrique:FormeGeometrique):
        self._formeGeo = formeGeometrique
    def get_perimetreSurface(self):
        return [(i.perimetre(),i.surface(), i._nom )for i in self._formeGeo]
```

La batterie de test est :

```
from formGeometric import*

# Test de la classe Rectangle
print('===Classe Rectangle===')
rectangle = Rectangle(7, 5)
print("Longueur =",rectangle._longueur, ";", "Largeur =", rectangle._larguer)
print("Périmètre: ",rectangle.perimetre(), "mètres")
print("Surface: ",rectangle.surface(),"mètres carrés", "\n")

# Test de la classe Cercle
print('===Classe Cercle===')
cercle = Cercle(5)
print("Rayon =", cercle._rayon)
print("Périmètre: ",cercle.perimetre(), "mètres")
print("Surface: ",cercle.surface(), "mètres carrés", "\n")

# Test de la classe Triangle
print('===Classe Triangle===')
triangle = Triangle(5, 6, 5)
print("Coté_1 =",triangle._cote1,";","Coté_2 =",triangle._cote2,";","Coté_3
=",triangle._cote3)
print("Périmètre: ",triangle.perimetre(), "mètres")
print("Surface: ",triangle.surface(), "mètres carrés", "\n")

# Test de la classe Carre
print('===Classe Carre===')
carre = Carre(4)
print("Coté =",carre._longueur)
print("Périmètre: ",carre.perimetre(), "mètres")
print("Surface: ",carre.surface(), "mètres carrés", "\n")

# Test de la classe TriangleRectangle
print('===Classe TriangleRectangle===')
triangleRectangle = TriangleRectangle(4,3)
print("Coté_1 =",triangleRectangle._cote1,";","Coté_2
=",triangleRectangle._cote2,";","Hypoténuse =",triangleRectangle._cote3)
print("Périmètre: ",triangleRectangle.perimetre(), "mètres")
print("Surface: ",triangleRectangle.surface(), "mètres carrés", "\n \n")

# Test de la classe PerimetreSurface
print("===Classe PerimetreSurface===")
perimetre = PerimetreSurface(rectangle, cercle, triangle, carre,
triangleRectangle).get_perimetreSurface()
for i in perimetre:
    print("-->Figure: ", i[2],'\n',"Périmètre: ",i[0],"mètres", '\n',"Surface:
", i[1],"mètres carrés", '\n')
```

8) En dehors de l'héritage nous pouvons utiliser **le polymorphisme** qui un concept de la technologie orientée objet.




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

class FormeGeometrique(ABC):

    @abstractmethod
    def perimetre(self):
        pass

    @abstractmethod
    def surface(self):
        pass

class Rectangle(FormeGeometrique):
    """Permet de calculer le périmetre et la surface d'un rectangle"""
    def __init__(self, longueur, largeur):
        self._longueur = longueur
        self._larguer = largeur
        self._nom = "rectangle"

    def perimetre(self):
        return f"Le périmètre du {self._nom} est: {2*(self._longueur +
self._larguer)} mètre"
```

```
    def surface(self):
        return f"La surface du {self._nom} est: {self._longueur*self._larguer}
mètre carré"

class Cercle(FormeGeometrique):
    """Permet de calculer le périmètre ou la circonférence et la surface du
cercle"""
    def __init__(self, rayon):
        self._rayon = rayon

    def perimetre(self):
        return f"Le périmètre ou lacirconférence du cercle est:
{2*pi*self._rayon} mètre "

    def surface(self):
        return f"La surface du cercle est: {pi*(self._rayon**2)} mètre carré"

class Triangle(FormeGeometrique):
    """Permet de calculer le périmètre et la surface d'un triangle"""
    def __init__(self, cote_1, cote_2, cote_3):
        self._cote1 = cote_1
        self._cote2 = cote_2
        self._cote3 = cote_3
        self._nom = "triangle"

    def perimetre(self):
        return f"Le périmètre du {self._nom} est: {self._cote1 + self._cote2 +
self._cote3} mètre "

    def surface(self):
        perimetre = self._cote1 + self._cote2 + self._cote3
        return f"""La surface du {self._nom} est:
{sqrt((perimetre/2)*((perimetre/2) - self._cote1)
        *((perimetre/2) - self._cote2)*((perimetre/2) - self._cote3))} metrè
carré"""

class Carre(Rectangle):
    """Permet de calculer le périmètre et la surface d'un Cercle"""
    def __init__(self, cote):
        self._longueur = cote
        self._larguer = cote
        self._nom = "carré"

    def perimetre(self):
        return f"Le périmètre du {self._nom} est: {self._longueur*4} mètre "

    def surface(self):
        return f"""La surface du {self._nom} est:
{self._longueur*self._longueur} metrè carré"""
```

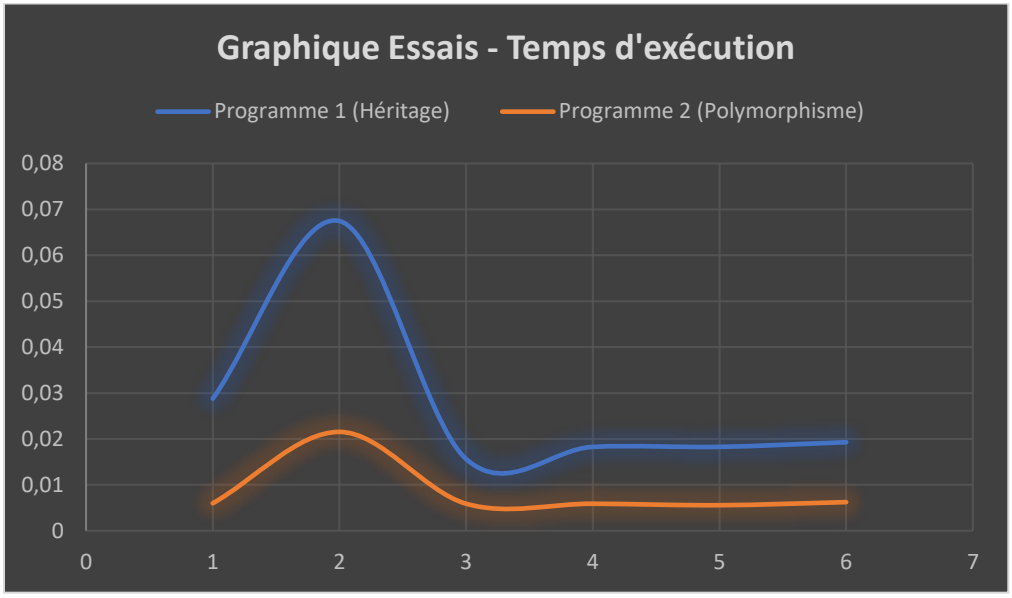
```
class TriangleRectangle(Triangle):
    """Permet de calculer le périmètre et la surface d'un triangle
rectangle"""
    def __init__(self, cote_1, cote_2):
        self._cote1 = cote_1
        self._cote2 = cote_2
        self._cote3 = sqrt((self._cote1**2) + (self._cote2**2))
        self._nom = "triangle réctangle"

    def perimetre(self):
        return f"Le périmètre du {self._nom} est: {self._cote1 + self._cote2 +
self._cote3} mètre "

    def surface(self):
        perimetre = self._cote1 + self._cote2 + self._cote3
        return f"""La surface du {self._nom} est:
{sqrt((perimetre/2)*((perimetre/2) - self._cote1)
        *((perimetre/2) - self._cote2)*((perimetre/2) - self._cote3))} metrè
carré"""


def formeGemetrique(*figures):
    return [(figure.perimetre(), figure.surface()) for figure in figures]
```

9)

Les données des essais:

Données						**Temps d'éxecution**	
Essais	Rectangle	Cerce	Triangle	Carre	TriangleRectangle	Programme 1 (Héritage)	Programme 2 (Polymorphisme)
1	(7 ; 5)	5	(5 ; 6 ; 5)	4	(4 ; 3)	0,02877	0,00599
2	(8 ; 6)	7	(6 ; 7 ; 6)	5	(5 ; 4)	0,06747	0,02154
3	(9 ; 7)	8	(7 ; 8 ; 7)	6	(6 ; 4)	0,01563	0,0059
4	(10 ; 8)	9	(8 ; 9 ; 8)	7	(7 ; 4)	0,01828	0,0059
5	(11 ; 9)	10	(9 ; 10 ; 9)	8	(8 ; 4)	0,01828	0,00557
6	(12 ; 10)	11	(10 ; 11 ; 10)	9	(9 ; 4)	0,01929	0,00624

Graphique



Nous constatons que le temps d’exécution du programme 2 (Polymorphisme) est petit par rapport au temps d’exécution du programme 1 (Héritage).

10) Nous avons choisi le programme 2 (Polymorphisme) car en termes de temps d’exécution, il est avantageux. Vu que le temps d’exécution d’un programme fait parti de critère de choix d’un algorithme.