**Постановка задачи**

Необходимо выполнить:

I. Базовый класс для всех вариантов:  
classFigure  
сместить фигуру на (dx,dy) –только видимую  
установить цвет фигуры –только видимой  
получить цветпоказать спрятать фигуру  
признак видимости  
вычислить периметр и площадь фигуры

II. Выполнить:  
1.Реализацию методов класса Figure.  
2.Скорректировать интерфейс Figure, чтобы методы calcParams и draw стали чисто виртуальными.  
3.Определить производный класс (в соответствии с вариантом задания).  
4.Определить дополнительный метод в производном классе для изменения размеров:  
setSizes(длина, высота);  
или setSizes(длина, высота, радиус);  
или setSizes(радиус, угол1, угол2);  
и т.д.,(т.е. задавать значения, указываемые в аргументах конструктора, начиная с 4-го )  
5.От написанного класса произвести новый дочерний класс -закрашенная фигура.  
Например, закрашенный ромб (FilledRomb ← Romb ← Figure).  
6.Добавить к параметрам конструктора нового дочернего класса цвет заполнения.  
7.Определить дополнительный метод у нового дочернего класса для изменения цвета заполнения:  
setFillColor(c);  
8.Переопределить у нового дочернего класса метод изменения цвета границы:  
setBorderColor(c);   
(если новый цвет границы совпадает с текущим цветом заполнения, то цвет границы не изменять)  
9.При тестировании динамически создать две фигуры 2 разных классов, вызвать все методы.

6. Сектор круга (незакрашенный)  
Sector(цвет линий, x и y цетра круга, радиус, угол1, угол2)

**Код программы**

import turtle

class Figure():

def \_\_init\_\_(self, borderColor, x, y, visible = True):

self.x = x

self.y = y

self.borderColor = borderColor

self.visible = visible

def moveTo(self, dx, dy):

self.x += dx

self.y += dy

def setBorderColor(self, borderColor):

self.borderColor = borderColor

def set\_visible(self, visible):

self.visible = visible

def calcParams(self):

return 0

def draw(self):

return 0

class Sector(Figure):

def \_\_init\_\_(self, borderColor, x, y, radius, angle1, angle2):

Figure.\_\_init\_\_(self, borderColor, x, y)

self.radius = radius

self.angle1 = angle1

self.angle2 = angle2

def draw(self):

if self.visible == True:

turtle.penup()

turtle.setpos(self.x, self.y - self.radius)

turtle.pendown()

turtle.pencolor(self.borderColor)

turtle.setheading(self.angle1)

turtle.circle(self.radius, self.angle2)

turtle.penup()

turtle.setpos(self.x, self.y)

turtle.pendown()

turtle.setheading(self.angle1 - 90)

turtle.forward(self.radius)

turtle.penup()

turtle.setpos(self.x, self.y)

turtle.pendown()

turtle.setheading(self.angle2 - 90)

turtle.forward(self.radius)

def setSizes(self, radius, angle1, angle2):

self.rad = radius

self.ang1 = angle1

self.ang2 = angle2

def calcParams(self):

print("Периметр: ", 3.14 \* self.radius \* (self.angle2 - self.angle1) / 180)

print("Площадь: ", 3.14 \* self.radius \* self.radius \*(self.angle2 - self.angle1) )

class FilledSector(Sector):

def \_\_init\_\_(self, borderColor, x, y, fillColor, radius, angle1, angle2):

Sector.\_\_init\_\_(self, borderColor, x, y, radius, angle1, angle2)

self.fillColor = fillColor

def setFillColor(self, fillColor):

if self.borderColor != fillColor:

self.fillColor = fillColor

def setBorderColor(self, borderColor):

if self.fillColor != borderColor:

self.borderColor = borderColor

def draw(self):

turtle.penup()

turtle.setpos(self.x, self.y - self.radius)

turtle.pendown()

turtle.begin\_fill()

turtle.fillcolor(self.fillColor)

turtle.pencolor(self.borderColor)

turtle.setheading(self.angle1)

turtle.circle(self.radius, self.angle2)

turtle.end\_fill()

turtle.penup()

turtle.pendown()

turtle.begin\_fill()

turtle.fillcolor(self.fillColor)

turtle.setheading(self.angle2 + 90)

turtle.forward(self.radius)

turtle.penup()

turtle.pendown()

turtle.setheading(self.angle1 - 90)

turtle.forward(self.radius)

turtle.end\_fill()

turtle.pensize(3)

print("Сектор 1")

sector1 = Sector("black", -200, 40, 70, 0, 45)

sector1.draw()

sector1.moveTo(75, -150)

sector1.calcParams()

print("Сектор 2")

sector1.setBorderColor("red")

sector1.setSizes(90, 45, 90)

sector1.draw()

sector1.calcParams()

print("Сектор 3")

sector2 = FilledSector("red", 200, 40, "green", 70, 0, 45)

sector2.draw()

sector2.moveTo(-75, -150)

sector2.calcParams()

print("Сектор 4")

sector2.setFillColor("yellow")

sector2.setSizes(90, 45, 90)

sector2.draw()

sector2.calcParams()

input()

**Результат работы:**

 