Московский государственный технический университет имени Н.Э.Баумана

Факультет Радиотехнический

Кафедра ИУ5 Системы обработки информации и управления

**Отчет по лабораторной работе № 2 по курсу**

**Базовые компоненты**

**"Объектно-ориентированные возможности языка Python"**

7

(количество листов)

|  |  |  |
| --- | --- | --- |
| Исполнитель |  |  |
| Студент группы РТ5-31Б | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Валиков И.Е. |
|  |  | «\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. |
|  |  |  |
| Проверил |  |  |
| Доцент кафедры ИУ5 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Гапанюк Ю.Е. |
|  |  | «\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. |

г. Москва – 2022

Оглавление

1 Цель выполнения лабораторной работы .…………………………………… 3

2 Листинг основной программы для проверки классов (main.py) .…...……… 3

3 Листинг программы с абстрактным классом «Геометрическая фигура» (figure.py) ………………………………………………………….…...………… 3

4 Листинг программы с классом «Прямоугольник» (rectangle.py) ...………… 3

5 Листинг программы с классом «Круг» (circle.py) …………....…...………… 4

6 Листинг программы с классом «Квадрат» (square.py) …………....………… 5

7 Листинг программы с классом «Цвет фигуры» (color.py) …..…...………… 6

8 Результаты работы программы …..…………………………………………... 7

9 Выводы по результатам работы …...…………………………………………. 7

Цель выполнения лабораторной работы

Изучение объектно-ориентированных возможностей языка Python путём разработки нескольких наследующихся классов.

Листинг основной программы для проверки классов (main.py)

from lab\_python\_oop.rectangle import Rectangle

from lab\_python\_oop.circle import Circle

from lab\_python\_oop.square import Square

import pymorphy2

def main():

morph = pymorphy2.MorphAnalyzer()

blue = morph.parse("синий")[0].inflect({'gent'})

green = morph.parse("зелёный")[0].inflect({'gent'})

red = morph.parse("красный")[0].inflect({'gent'})

rect = Rectangle(blue.word, 20, 19)

circle = Circle(green.word, 20)

square = Square(red.word, 20)

print(rect)

print(circle)

print(square)

if \_\_name\_\_ == "\_\_main\_\_":

main()

Листинг программы с абстрактным классом «Геометрическая фигура» (figure.py)

'''абстрактный класс "Геометрическая фигура", от которого наследуются другие'''

from abc import ABC, abstractmethod

class Figure(ABC):

@abstractmethod

# вычисление площади

def square(self):

pass

Листинг программы с классом «Прямоугольник» (rectangle.py)

'''класс "Прямоугольник"'''

from lab\_python\_oop.figure import Figure

from lab\_python\_oop.color import FigureColor

class Rectangle(Figure):

FIGURE\_TYPE = "Прямоугольник"

@classmethod

# get-method

def get\_figure\_type(cls):

return cls.FIGURE\_TYPE

def \_\_init\_\_(self, color, width, height):

self.width = width

self.height = height

self.fc = FigureColor()

self.fc.colorproperty = color

# вычисление площади

def square(self):

return self.width \* self.height

# вывод данных фигуры

def \_\_repr\_\_(self):

return '{} {} цвета с шириной {} и высотой {} имеет площадь {}'.format(

Rectangle.get\_figure\_type(),

self.fc.colorproperty,

self.width,

self.height,

self.square()

)

Листинг программы с классом «Круг» (circle.py)

'''класс "Круг"'''

from lab\_python\_oop.figure import Figure

from lab\_python\_oop.color import FigureColor

from math import pi

class Circle(Figure):

FIGURE\_TYPE = "Круг"

@classmethod

# get-method

def get\_figure\_type(cls):

return cls.FIGURE\_TYPE

def \_\_init\_\_(self, color, r):

self.radius = r

self.fc = FigureColor()

self.fc.colorproperty = color

# вычисление площади

def square(self):

return round(pi \* (self.radius \*\* 2), 2)

def \_\_repr\_\_(self):

return '{} {} цвета радиусом {} имеет площадь {}'.format(

Circle.get\_figure\_type(),

self.fc.colorproperty,

self.radius,

self.square()

)

Листинг программы с классом «Квадрат» (square.py)

'''класс "Квадрат"'''

from lab\_python\_oop.rectangle import Rectangle

class Square(Rectangle):

FIGURE\_TYPE = "Квадрат"

@classmethod

# get-method

def get\_figure\_type(cls):

return cls.FIGURE\_TYPE

def \_\_init\_\_(self, color, side):

self.side = side

super().\_\_init\_\_(color, self.side, self.side)

# вывод данных фигуры

def \_\_repr\_\_(self):

return '{} {} цвета со стороной {} имеет площадь {}'.format(

Square.get\_figure\_type(),

self.fc.colorproperty,

self.side,

self.square()

)

Листинг программы с классом «Цвет фигуры» (color.py)

'''класс "Цвет фигуры"'''

class FigureColor:

def \_\_init\_\_(self):

self.\_color = None

@property

# get-метод

def colorproperty(self):

return self.\_color

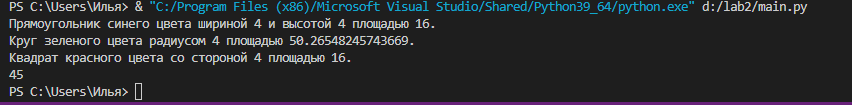
@colorproperty.setter

# set-метод

def colorproperty(self, value):

self.\_color = value

Результаты работы программы



Выводы по результатам работы

При выполнении работы я изучил объектно-ориентированные возможности языка Python и разработал нескольких наследующихся классов.