Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Казовые компоненты интернет технологий»

Отчет по лабораторной работе №2

«Объектно-ориентированные возможности языка Python»

Выполнил: студент группы ИУ5-33Б Попов С. Д.

Руководитель: преподаватель каф. ИУ5 Гапанюк Ю.Е.

Москва, 2022 г.

Задание:

1. Необходимо создать виртуальное окружение и установить в него хотя бы один внешний пакет с использованием pip.
2. Необходимо разработать программу, реализующую работу с классами. Программа должна быть разработана в виде консольного приложения на языке Python 3.
3. Все файлы проекта (кроме основного файла main.py) должны располагаться в пакете lab\_python\_oop.
4. Каждый из нижеперечисленных классов должен располагаться в отдельном файле пакета lab\_python\_oop.
5. Абстрактный класс «Геометрическая фигура» содержит абстрактный метод для вычисления площади фигуры. Подробнее про абстрактные классы и методы Вы можете прочитать здесь.
6. Класс «Цвет фигуры» содержит свойство для описания цвета геометрической фигуры. Подробнее про описание свойств Вы можете прочитать здесь.
7. Класс «Прямоугольник» наследуется от класса «Геометрическая фигура». Класс должен содержать конструктор по параметрам «ширина», «высота» и «цвет». В конструкторе создается объект класса «Цвет фигуры» для хранения цвета. Класс должен переопределять метод, вычисляющий площадь фигуры.
8. Класс «Круг» создается аналогично классу «Прямоугольник», задается параметр «радиус». Для вычисления площади используется константа math.pi из модуля math.
9. Класс «Квадрат» наследуется от класса «Прямоугольник». Класс должен содержать конструктор по длине стороны. Для классов «Прямоугольник», «Квадрат», «Круг»:

o Определите метод "repr", который возвращает в виде строки основные параметры фигуры, ее цвет и площадь. Используйте метод format - https://pyformat.info/ o Название фигуры («Прямоугольник», «Квадрат», «Круг») должно задаваться в виде поля данных класса и возвращаться методом класса.

1. В корневом каталоге проекта создайте файл main.py для тестирования Ваших классов (используйте следующую конструкцию

- https://docs.python.org/3/library/\_\_main\_\_.html). Создайте следующие объекты и выведите о них информацию в консоль (N - номер Вашего варианта по списку группы):

* + Прямоугольник синего цвета шириной N и высотой N.
  + Круг зеленого цвета радиусом N. o Квадрат красного цвета со стороной N.
  + Также вызовите один из методов внешнего пакета, установленного с использованием pip.

1. Дополнительное задание. Протестируйте корректность работы Вашей программы с помощью модульного теста.

# Текст программы:

# main.py

from lab\_python\_oop.rectangle import Rectangle

from lab\_python\_oop.circle import Circle

from lab\_python\_oop.square import Square

from cowsay import cow, trex, dragon

def main():

    r = Rectangle("синего", 9, 9)

    c = Circle("зеленого", 9)

    s = Square("красного", 9)

    trex(r.\_\_repr\_\_())

    dragon(c.\_\_repr\_\_())

    cow(s.\_\_repr\_\_())

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

    main()

# test.py

import unittest

from lab\_python\_oop.rectangle import Rectangle

from lab\_python\_oop.circle import Circle

from lab\_python\_oop.square import Square

import math

class Test\_Rectangle(unittest.TestCase):

    def test\_rect\_init(self):

        r = Rectangle("малинового", 19, 11)

        self.assertEqual(r.get\_figure\_type(), 'Прямоугольник')

        self.assertEqual(r.fc.colorproperty, 'малинового')

        self.assertEqual(r.width, 19)

        self.assertEqual(r.height, 11)

        self.assertEqual(r.square(), r.height \* r.width)

    def test\_rect\_repr(self):

        r = Rectangle("малинового", 19, 11)

        self.assertEqual(r.\_\_repr\_\_(), 'Прямоугольник малинового цвета шириной 19 и высотой 11 площадью 209.')

class Test\_Circle(unittest.TestCase):

    def test\_circ\_init(self):

        c = Circle("зеленого", 12)

        self.assertEqual(c.get\_figure\_type(), 'Круг')

        self.assertEqual(c.fc.colorproperty, 'зеленого')

        self.assertEqual(c.r, 12)

        self.assertEqual(c.square(), math.pi \* (c.r \*\* 2))

    def test\_circ\_repr(self):

        c = Circle("зеленого", 12)

        self.assertEqual(c.\_\_repr\_\_(), 'Круг зеленого цвета радиусом 12 площадью 452.3893421169302.')

class Test\_Square(unittest.TestCase):

    def test\_sq\_init(self):

        s = Square("красного", 12)

        self.assertEqual(s.get\_figure\_type(), 'Квадрат')

        self.assertEqual(s.fc.colorproperty, 'красного')

        self.assertEqual(s.width, 12)

        self.assertEqual(s.height, 12)

        self.assertEqual(s.square(), s.height \* s.width)

    def test\_sq\_repr(self):

        s = Square("красного", 12)

        self.assertEqual(s.\_\_repr\_\_(), 'Квадрат красного цвета со стороной 12 площадью 144.')

if \_\_name\_\_ == '\_\_main\_\_':

    unittest.main()

lab\_python\_opp/figure.py

from abc import ABC, abstractmethod

class Figure(ABC):

    """

    Абстрактный класс «Геометрическая фигура»

    """

    @abstractmethod

    def square(self):

        """

        содержит виртуальный метод для вычисления площади фигуры.

        """

        pass

# lab\_python\_opp/circle.py

from lab\_python\_oop.figure import Figure

from lab\_python\_oop.color import FigureColor

import math

class Circle(Figure):

    """

    Класс «Круг» наследуется от класса «Геометрическая фигура».

    """

    FIGURE\_TYPE = "Круг"

    @classmethod

    def get\_figure\_type(cls):

        return cls.FIGURE\_TYPE

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

        """

        Класс должен содержать конструктор по параметрам «радиус» и «цвет». В конструкторе создается объект класса «Цвет фигуры» для хранения цвета.

        """

        self.r = r\_param

        self.fc = FigureColor()

        self.fc.colorproperty = color\_param

    def square(self):

        """

        Класс должен переопределять метод, вычисляющий площадь фигуры.

        """

        return math.pi\*(self.r\*\*2)

    def \_\_repr\_\_(self):

        return '{} {} цвета радиусом {} площадью {}.'.format(

            Circle.get\_figure\_type(),

            self.fc.colorproperty,

            self.r,

            self.square()

        )

# lab\_python\_opp/rectangle.py

from lab\_python\_oop.figure import Figure

from lab\_python\_oop.color import FigureColor

class Rectangle(Figure):

    """

    Класс «Прямоугольник» наследуется от класса «Геометрическая фигура».

    """

    FIGURE\_TYPE = "Прямоугольник"

    @classmethod

    def get\_figure\_type(cls):

        return cls.FIGURE\_TYPE

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

        """

        Класс должен содержать конструктор по параметрам «ширина», «высота» и «цвет». В конструкторе создается объект класса «Цвет фигуры» для хранения цвета.

        """

        self.width = width\_param

        self.height = height\_param

        self.fc = FigureColor()

        self.fc.colorproperty = color\_param

    def square(self):

        """

        Класс должен переопределять метод, вычисляющий площадь фигуры.

        """

        return self.width\*self.height

    def \_\_repr\_\_(self):

        return '{} {} цвета шириной {} и высотой {} площадью {}.'.format(

            Rectangle.get\_figure\_type(),

            self.fc.colorproperty,

            self.width,

            self.height,

            self.square()

        )

# lab\_python\_opp/square.py

from lab\_python\_oop.rectangle import Rectangle

class Square(Rectangle):

    """

    Класс «Квадрат» наследуется от класса «Прямоугольник».

    """

    FIGURE\_TYPE = "Квадрат"

    @classmethod

    def get\_figure\_type(cls):

        return cls.FIGURE\_TYPE

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

        """

        Класс должен содержать конструктор по параметрам «сторона» и «цвет».

        """

        self.side = side\_param

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

    def \_\_repr\_\_(self):

        return '{} {} цвета со стороной {} площадью {}.'.format(

            Square.get\_figure\_type(),

            self.fc.colorproperty,

            self.side,

            self.square()

        )

# lab\_python\_opp/color.py

class FigureColor:

    """

    Класс «Цвет фигуры»

    """

    def \_\_init\_\_(self):

        self.\_color = None

    @property

    def colorproperty(self):

        """

        Get-аксессор

        """

        return self.\_color

    @colorproperty.setter

    def colorproperty(self, value):

        """

        Set-аксессор

        """

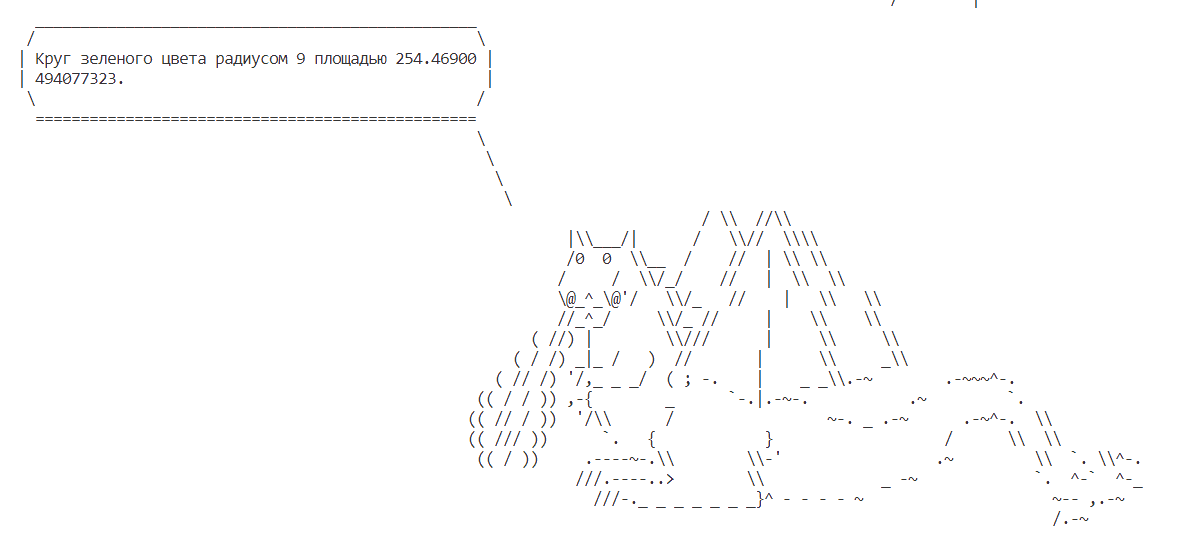
        self.\_color = value

requrements.txt



Примеры выполнения:

main.py



test.py

