Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №2
«Объектно-ориентированные возможности языка Python»

Выполнил:
Студент группы ИУ5-32Б:
Секретов Кирилл
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.
Подпись и дата:

Москва, 2022 г.

Описание задания

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.

6. Класс «Цвет фигуры» содержит свойство для описания цвета геометрической фигуры.

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»:

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

10. В корневом каталоге проекта создайте файл main.py для тестирования Ваших классов. 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.
    - Квадрат красного цвета со стороной N.

- Также вызовите один из методов внешнего пакета, установленного сиспользованием pip.

11. **Дополнительное задание.** Протестируйте корректность работы Вашейпрограммы с помощью модульного теста.

Текст программы

Circle.py

```
import math
from lab_python_oop.figure import Figure
from lab_python_oop.color import FigureColor


class Circle(Figure):
    FIGURE_TYPE = "Круг"

    def __init__(self, radius, color):
        self.radius = radius
        self.color = FigureColor()
        self.color.colorproperty = color

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

    @classmethod
    def get_figure_type(cls):
        return cls.FIGURE_TYPE
```

```
    def __repr__(self):
        return "{} {} цвета, радиусом {} и площадью {}.".format(
            self.get_figure_type(),
            self.color.colorproperty,
            self.radius,
            self.square()
        )
```

color.py

```
class FigureColor:
    def __init__(self):
        self._color = None

    @property
    def colorproperty(self):
        return self._color

    @colorproperty.setter
    def colorproperty(self, val):
        self._color = val

    @colorproperty.deleter
    def colorproperty(self):
        del self._color
```

figure.py

```
from abc import ABC, abstractmethod



class Figure(ABC):
    @abstractmethod
    def square(self):
        pass
```

rectangle.py

```
from lab_python_oop.figure import Figure
from lab_python_oop.color import FigureColor



class Rectangle(Figure):
    FIGURE_TYPE = "Прямоугольник"

    def __init__(self, width, height, color):
        self.width = width
        self.height = height
        self.color = FigureColor()
        self.color.colorproperty = color

    def square(self):
```

```
        return self.width * self.height

    @classmethod
    def get_figure_type(cls):
        return cls.FIGURE_TYPE

    def __repr__(self):
        return "{} {} цвета, шириной {}, высотой {} и площадью {}.".format(
            self.get_figure_type(),
            self.color.colorproperty,
            self.width,
            self.height,
            self.square()
        )
```

square.py

```
from lab_python_oop.rectangle import Rectangle
from lab_python_oop.color import FigureColor


class Square(Rectangle):
    FIGURE_TYPE = "Квадрат"

    def __init__(self, side, color):
        self.side = self.width = self.height = side
```

```
        # self.side = side
        self.color = FigureColor()
        self.color.colorproperty = color

    @classmethod
    def get_figure_type(cls):
        return cls.FIGURE_TYPE

    def __repr__(self):
        return "{} {} цвета со стороной {} и площадью {}.".format(
            self.get_figure_type(),
            self.color.colorproperty,
            self.side,
            self.square()
        )
```

main.py

```
from lab_python_oop.rectangle import Rectangle
from lab_python_oop.circle import Circle
from lab_python_oop.square import Square
import matplotlib.pyplot as plt
import numpy as np

# Data for plotting
t = np.arange(0.0, 2.0, 0.01)
s = 1 + np.sin(2 * np.pi * t)
```

```
fig, ax = plt.subplots()
ax.plot(t, s)

ax.set(xlabel='time (s)', ylabel='voltage (mV)',
       title='About as simple as it gets, folks')
ax.grid()

fig.savefig("test.png")
plt.show()



def main():
    rect = Rectangle(20, 20, "Синего")
    circle = Circle(20, "Зеленого")
    square = Square(20, "Красного")
    print(rect, circle, square, sep="\n")

if __name__ == "__main__":
    main()
```

test.py

```
import unittest
from lab_python_oop.color import FigureColor
from lab_python_oop.rectangle import Rectangle
```

```
from lab_python_oop.circle import Circle
from lab_python_oop.square import Square


class TestLabPythonOOP(unittest.TestCase):
    def test_color(self):
        color = FigureColor()
        color.colorproperty = "red"
        self.assertDictEqual(color.__dict__, {"_color": "red"})

        color.colorproperty = "green"
        self.assertDictEqual(color.__dict__, {"_color": "green"})

        self.assertFalse(hasattr(FigureColor, "_color"))

    def test_rectangle(self):
        self.assertEqual(Rectangle.get_figure_type(), "Прямоугольник")

        rect = Rectangle(3, 5, "black")
        self.assertEqual(rect.width, 3)
        self.assertEqual(rect.height, 5)
        self.assertEqual(rect.color.colorproperty, "black")
        self.assertEqual(rect.square(), 15)

    def test_circle(self):
```

```
        self.assertEqual(Circle.get_figure_type(), "Круг")

        circle = Circle(5, "white")
        self.assertEqual(circle.color.colorproperty, "white")
        self.assertEqual(circle.radius, 5)
        self.assertTrue(abs(78.539816 - circle.square()) < 0.000001)

    def test_square(self):
        self.assertEqual(Square.get_figure_type(), "Квадрат")

        square = Square(8, "yellow")
        self.assertEqual(square.side, 8)
        self.assertEqual(square.color.colorproperty, "yellow")
        self.assertEqual(square.square(), 64)


if __name__ == "__main__":
    unittest.main()
```

Примеры выполнения программы:

```
Прямоугольник Синего цвета, шириной 20, высотой 20 и площадью 400.
Круг Зеленого цвета, радиусом 20 и площадью 1256.6370614359173.
Квадрат Красного цвета со стороной 20 и площадью 400.
>>>
```