**Московский государственный технический университет им. Н.Э. Баумана**

Факультет «Радиотехника»

Кафедра «Информатика и вычислительная техника»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №2

«Объектно-ориентированные возможности языка Python»

Выполнил: Проверил:

студент группы РТ5-31Б: преподаватель каф. ИУ5

Кузнецов М.Д. Гапанюк Ю.Е.

Подпись и дата: Подпись и дата:

Москва, 2023 г.

**Постановка задачи:**

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»:
   * Определите метод "repr", который возвращает в виде строки основные параметры фигуры, ее цвет и площадь. Используйте метод format -<https://pyformat.info/>
   * Название фигуры должно задаваться в виде поля данных класса и возвращаться методом класса

1. 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 с использованием pip.

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

**Текст программы:**

# Main.py

import numpy as np

from lab\_python\_oop.rectangle import Rectangle from lab\_python\_oop.circle import Circle from lab\_python\_oop.square import Square

def main():

r = Rectangle(8, 8, "синего") c = Circle(8, "зеленого") s = Square(8, "красного") print(r) print(c) print(s)

print(np.zeros([2, 3]))

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

main()

# Сircle.py

from lab\_python\_oop import GeometryFigure from lab\_python\_oop import Color import math class Circle(GeometryFigure.GeometryFigure):

name = "Круг" def \_\_init\_\_(self, color, r):

self.color=Color.Color(color).color self.r=r self.area = Circle.Area(self) def Area(self):

self.area = (self.r \*\* 2) \* math.pi return self.area

# GeometryFigure.py

from abc import ABC, abstractmethod

class GeometryFigure(ABC): @abstractmethod def Area(self):

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

return f'{self.name}, площадь = {self.area}, цвет = {self.color}.'

# Rectangle.py

from lab\_python\_oop import GeometryFigure from lab\_python\_oop import Color class Rectangle(GeometryFigure.GeometryFigure):

name="Прямоугольник"

def \_\_init\_\_(self, color, a, b=1):

self.color=Color.Color(color).color self.a=a self.b=b self.area = self.Area() def Area(self):

return self.a \* self.b

'

# Square.py

from lab\_python\_oop import Rectangle class Square(Rectangle.Rectangle):

name = "Квадрат" def Area(self): return self.a\*\*2

# Сolor.py

class Color(): def \_\_init\_\_(self, color):

self.color = color

@property def colorFigure(self): return self.color

# test\_Circle.py

from lab\_python\_oop import Circle import math import pytest

@pytest.mark.parametrize("color, area", [("Красный", 17),

("Синий", 10), ("Зеленый", 100)]) def test\_Circle(color, area):

assert Circle.Circle(color, area).color == color assert Circle.Circle(color, area).area == (area\*\* 2) \* math.pi assert Circle.Circle(color, area).name == "Круг”

# test\_Rectangle.py

from lab\_python\_oop import Rectangle import pytest

@pytest.mark.parametrize("color, a, b", [("Красный", 17, 17),

("Синий", 10, 17),

("Зеленый", 100,

2)]) def test\_Rectangle(color, a, b):

assert Rectangle.Rectangle(color, a, b).color == color assert Rectangle.Rectangle(color, a, b).area == a\*b assert Rectangle.Rectangle(color, a, b).name ==

"Прямоугольник" **test\_Square.py**

from lab\_python\_oop import Rectangle import pytest

@pytest.mark.parametrize("color, a, b", [("Красный", 17, 17),

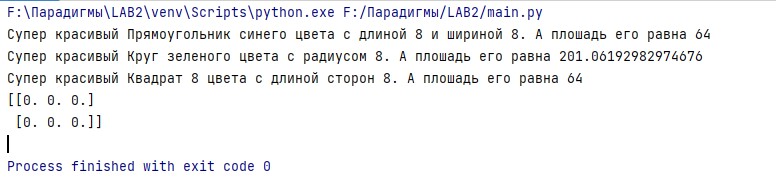
("Синий", 10, 17),

("Зеленый", 100,

2)]) def test\_Rectangle(color, a, b):

assert Rectangle.Rectangle(color, a, b).color == color assert Rectangle.Rectangle(color, a, b).area == a\*b assert Rectangle.Rectangle(color, a, b).name == "Прямоугольник"

**Выполнение программы:**



# Выполнение test\_Circle.py, test\_Rectangle.py, test\_Square.py

