**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»

Вариант №6

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы РТ5-31Б: |  | преподаватель каф. ИУ5 |
| Ергалиев А.Г. |  | 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()

**circle.py**

from lab\_python\_oop.figure import Figure  
from lab\_python\_oop.Сolor import Color  
import math  
  
  
class Circle(Figure):  
 FIGURE\_TYPE = "Круг"  
  
 @classmethod  
 def get\_figure\_type(cls):  
 return cls.FIGURE\_TYPE  
  
 def \_\_init\_\_(self, radius, color):  
 self.radius = radius  
 self.fc = Color()  
 self.fc.colorproperty = color  
  
 def square(self):  
 return self.radius \* self.radius \* math.pi  
  
 def \_\_repr\_\_(self):  
 return 'Супер красивый {} {} цвета с радиусом {}. А плошадь его равна {}'.format(  
 Circle.get\_figure\_type(),  
 self.fc.colorproperty,  
 self.radius,  
 self.square()  
 )

**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.Сolor import Color  
  
class Rectangle(Figure):  
 FIGURE\_TYPE = "Прямоугольник"  
  
 @classmethod  
 def get\_figure\_type(cls):  
 return cls.FIGURE\_TYPE  
  
 def \_\_init\_\_(self, width, length, color):  
 self.width = width  
 self.length = length  
 self.fc = Color()  
 self.fc.colorproperty = color  
  
  
 def square(self):  
 return self.width\*self.length  
  
  
 def \_\_repr\_\_(self):  
 return 'Супер красивый {} {} цвета с длиной {} и шириной {}. А плошадь его равна {}'.format(  
 Rectangle.get\_figure\_type(),  
 self.fc.colorproperty,  
 self.length,  
 self.width,  
 self.square()  
 )

**square.py**

from lab\_python\_oop.rectangle import Rectangle  
  
class Square(Rectangle):  
 FIGURE\_TYPE = "Квадрат"  
  
 @classmethod  
 def get\_figure\_type(cls):  
 return cls.FIGURE\_TYPE  
  
 def \_\_init\_\_(self, length, color):  
 self.length = length  
 super().\_\_init\_\_(color, self.length, self.length)  
  
 def square(self):  
 return (self.length\*\*2)  
  
 def \_\_repr\_\_(self):  
 return 'Супер красивый {} {} цвета с длиной сторон {}. А плошадь его равна {}'.format(  
 Square.get\_figure\_type(),  
 self.fc.colorproperty,  
 self.length,  
 self.square()  
 )

**color.py**

class Color:  
 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

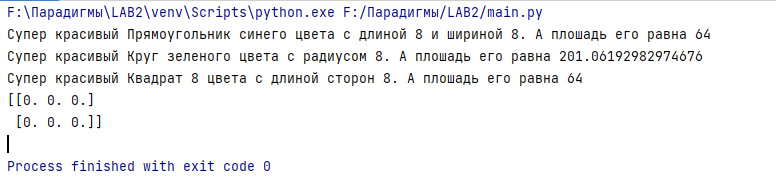
**test\_circle.py**

import unittest  
from lab\_python\_oop.circle import Circle  
  
  
class TestCircle(unittest.TestCase):  
 def test\_square(self):  
 c = Circle(8, "зеленого")  
 self.assertAlmostEqual(c.square(), 201.06192982974676, places=5)  
  
 def test\_repr(self):  
 c = Circle(8, "зеленого")  
 expected\_repr = 'Супер красивый Круг зеленого цвета с радиусом 8. А плошадь его равна 201.06192982974676'  
 self.assertEqual(repr(c), expected\_repr)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()

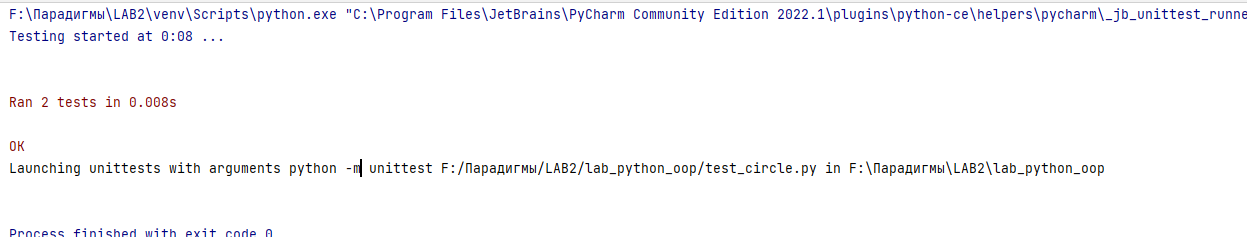
**test\_main.py**

import unittest  
from main import main  
from io import StringIO  
  
  
  
class TestMain(unittest.TestCase):  
 def test\_main(self):  
 captured\_output = StringIO()  
 expected\_output = [  
 'Супер красивый Прямоугольник синего цвета с длиной 8 и шириной 8. А плошадь его равна 64\n',  
 'Супер красивый Круг зеленого цвета с радиусом 8. А плошадь его равна 201.06192982974676\n',  
 'Супер красивый Квадрат красного цвета с длиной сторон 8. А плошадь его равна 64\n',  
 '[[0. 0. 0.]\n [0. 0. 0.]]\n'  
 ]  
  
 with captured\_output as (out, err):  
 main()  
  
 output = out.getvalue()  
  
 for i in range(len(expected\_output)):  
 self.assertEqual(output, expected\_output[i])  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()

**Выполнение программы:**



**Выполнение test\_circle**



**Выполнение test\_main**

