|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**  **«Московский государственный технический университет имени Н.Э. Баумана**  **(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)** |

**Факультет «Информатика и системы управления»**

**Кафедра «Системы обработки информации и управления»**

Расулов Арсен ИУ5-35Б

Парадигмы и конструкции языков программирования

**ОТЧЁТ ПО**

**Лабораторной работе №2**

**«Объектно-ориентированные возможности языка Python. Тестирование проекта.»**

Москва

2023

***Описание задания.***

1. *Написать тесты для следующего задания:*
2. Необходимо создать виртуальное окружение и установить в него хотя бы один внешний пакет с использованием pip.
3. Необходимо разработать программу, реализующую работу с классами. Программа должна быть разработана в виде консольного приложения на языке Python 3.
4. Все файлы проекта (кроме основного файла main.py) должны располагаться в пакете lab\_python\_oop.
5. Каждый из нижеперечисленных классов должен располагаться в отдельном файле пакета lab\_python\_oop.
6. Абстрактный класс «Геометрическая фигура» содержит абстрактный метод для вычисления площади фигуры. Подробнее про абстрактные классы и методы Вы можете прочитать [здесь.](https://docs.python.org/3/library/abc.html)
7. Класс «Цвет фигуры» содержит свойство для описания цвета геометрической фигуры. Подробнее про описание свойств Вы можете прочитать [здесь.](https://docs.python.org/3/library/functions.html#property)
8. 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.
9. Класс «Круг» создается аналогично классу «Прямоугольник», задается параметр «радиус». Для вычисления площади используется константа math.pi из модуля [math.](https://docs.python.org/3/library/math.html)
10. 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 («Прямоугольник», «Квадрат», «Круг») должно задаваться в виде поля данных класса и возвращаться методом класса.
11. 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.
    * Квадрат красного цвета со стороной N.
    * Также вызовите один из методов внешнего пакета, установленного с использованием pip.

***Текст программы.***

**test.py**

from lab\_python\_oop.square import Square  
from lab\_python\_oop.circle import Circle  
from lab\_python\_oop.rectangle import Rectangle  
  
import unittest  
  
blue = "Синего"  
red = "Красного"  
green = "Зеленого"  
  
  
class MyTesting(unittest.TestCase):  
 def setUp(self):  
 self.a = Rectangle(3, 2, blue)  
 self.b = Circle(5, green)  
 self.c = Square(5, red)  
  
 def test\_area(self):  
 from math import pi  
 self.assertEqual(self.a.area(), 6)  
 self.assertEqual(self.b.area(), pi \* 5 \* 5)  
 self.assertEqual(self.c.area(), 25)  
  
 def test\_color(self):  
 self.assertEqual(self.a.color.val, blue)  
 self.assertEqual(self.b.color.val, green)  
 self.assertEqual(self.c.color.val, red)  
  
 def test\_repr(self):  
 self.assertEqual(str(self.a), 'Прямоугольник Синего цвета с длиной стороны 3, шириной стороны 2, площадью 6.')  
 self.assertEqual(str(self.b), 'Круг Зеленого цвета радиусом 5 площадью 78.53981633974483.')  
 self.assertEqual(str(self.c), 'Квадрат Красного цвета с длиной стороны 5, площадью 25.')  
  
  
def main():  
 print(Rectangle(3, 2, "Синего"))  
 print(Circle(5, "Зеленого"))  
 print(Square(5, "Красного"))  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

main.py

from lab\_python\_oop.square import Square

from lab\_python\_oop.circle import Circle

from lab\_python\_oop.rectangle import Rectangle

def main():

print(Rectangle(3, 2, "Синего"))

print(Circle(5, "Зеленого"))

print(Square( 5, "Красного"))

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

main()

figure.py

from abc import ABC, abstractmethod

class Figure(ABC):

@abstractmethod

def area(self):

pass

color.py

class Color:

def \_\_init\_\_(self, color=None):

self.\_color = color

rectangle.py

from .figure import Figure

from .color import Color

class Rectangle(Figure):

type = "Прямоугольник"

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

self.width = width

self.length = length

self.r\_color = Color(color)

def area(self):

return self.width \* self.length

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

return '{} {} цвета с длиной стороны {}, шириной стороны {}, площадью {}.'.format(

self.type,

self.r\_color.\_color,

self.length,

self.width,

self.area()

)

circle.py

**from** .figure **import** Figure

**from** .color **import** Color

**import** math

**class** Circle(Figure):

type **=** "Круг"

**def** \_\_init\_\_(self, radius, color):

self**.**radius **=** radius

self**.**c\_color **=** Color(color)

**def** area(self):

**return** math**.**pi **\*** (self**.**radius **\*\*** 2)

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

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

self**.**type,

self**.**c\_color**.**\_color,

self**.**radius,

self**.**area()

)

square.py

**from** .rectangle **import** Rectangle

**class** Square(Rectangle):

type **=** "Квадрат"

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

self**.**side **=** side

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

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

**return** '{} {} цвета с длиной стороны {}, площадью {}.'**.**format(

self**.**type,

self**.**r\_color**.**\_color,

self**.**side,

self**.**area()

)

***Экранные формы.***

