Московский государственный технический

университет им. Н.Э. Баумана

Проверил:

Преподаватель каф. ИУ5

Подпись и дата:

Факультет «Информатика и система управления»

Кафедра ИУ5 «Система обработки информации и управления»

Выполнил:

Студент группы ИУ5-36Б

Яншин Т. И.

Подпись и дата:

Курс «Парадигмы и конструкции языков программирования»

Отчёт по лабораторной работе №2

**Объектно-ориентированные возможности языка Python**

**Цель лабораторной работы:** изучение объектно-ориентированных возможностей языка Python.

**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.](https://docs.python.org/3/library/abc.html)
6. 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)
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.](https://docs.python.org/3/library/math.html)
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 («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 (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.
11. **Дополнительное задание.** Протестируйте корректность работы Вашей программы с помощью модульного теста.

**Код программы:**

**main.py**

from lab\_python\_oop.rectangle import Rectangle  
from lab\_python\_oop.circle import Circle  
from lab\_python\_oop.square import Square  
import requests  
  
def main():  
 N = 22  
  
 # Создание объектов  
 rectangle = Rectangle(width=N, height=N, color="синий")  
 circle = Circle(radius=N, color="зеленый")  
 square = Square(side=N, color="красный")  
  
 # Вывод информации о фигурах  
 print(rectangle)  
 print(circle)  
 print(square)  
  
 # Вызов метода внешнего пакета  
 response = requests.get("https://api.github.com")  
 print(f"Статус запроса к GitHub API: {response.status\_code}")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

**circle.py**

from .geometric\_figure import GeometricFigure  
from .figure\_color import FigureColor  
import math  
  
class Circle(GeometricFigure):  
 def \_\_init\_\_(self, radius: float, color: str):  
 self.radius = radius  
 self.color = FigureColor(color)  
  
 def area(self) -> float:  
 return math.pi \* self.radius \*\* 2  
  
 def \_\_repr\_\_(self) -> str:  
 return "Фигура: {0}, Радиус: {1}, Цвет: {2}, Площадь: {3:.2f}".format(  
 self.name(),  
 self.radius,  
 self.color.color,  
 self.area()  
 )  
  
 @classmethod  
 def name(cls) -> str:  
 return "Круг"

**figure\_color.py**

class FigureColor:  
 def \_\_init\_\_(self, color: str):  
 self.\_color = color  
  
 @property  
 def color(self) -> str:  
 return self.\_color  
  
 @color.setter  
 def color(self, value: str):  
 self.\_color = value

**geometric\_figure.py**

from abc import ABC, abstractmethod  
  
class GeometricFigure(ABC):  
 @abstractmethod  
 def area(self):  
 pass  
  
 @classmethod  
 @abstractmethod  
 def name(cls):  
 pass

**rectangle.py**

from .geometric\_figure import GeometricFigure  
from .figure\_color import FigureColor  
  
class Rectangle(GeometricFigure):  
 def \_\_init\_\_(self, width: float, height: float, color: str):  
 self.width = width  
 self.height = height  
 self.color = FigureColor(color)  
  
 def area(self) -> float:  
 return self.width \* self.height  
  
 def \_\_repr\_\_(self) -> str:  
 return "Фигура: {0}, Ширина: {1}, Высота: {2}, Цвет: {3}, Площадь: {4}".format(  
 self.name(),  
 self.width,  
 self.height,  
 self.color.color,  
 self.area()  
 )  
  
 @classmethod  
 def name(cls) -> str:  
 return "Прямоугольник"

**square.py**

from .rectangle import Rectangle  
  
class Square(Rectangle):  
 def \_\_init\_\_(self, side: float, color: str):  
 super().\_\_init\_\_(width=side, height=side, color=color)  
 self.side = side  
  
 def \_\_repr\_\_(self) -> str:  
 return "Фигура: {0}, Сторона: {1}, Цвет: {2}, Площадь: {3}".format(  
 self.name(),  
 self.side,  
 self.color.color,  
 self.area()  
 )  
  
 @classmethod  
 def name(cls) -> str:  
 return "Квадрат"

**test\_figures.py**

import unittest  
from lab\_python\_oop.rectangle import Rectangle  
from lab\_python\_oop.circle import Circle  
from lab\_python\_oop.square import Square  
  
class TestGeometricFigures(unittest.TestCase):  
 def test\_rectangle\_area(self):  
 rect = Rectangle(width=3, height=4, color="blue")  
 self.assertEqual(rect.area(), 12)  
  
 def test\_circle\_area(self):  
 circle = Circle(radius=1, color="green")  
 self.assertAlmostEqual(circle.area(), 3.141592653589793)  
  
 def test\_square\_area(self):  
 square = Square(side=5, color="red")  
 self.assertEqual(square.area(), 25)  
  
 def test\_square\_is\_rectangle(self):  
 square = Square(side=2, color="yellow")  
 self.assertIsInstance(square, Rectangle)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()  
  
# python -m unittest discover -s tests

Экранные формы с примерами выполнения программы

|  |
| --- |
|  |
|  |
|  |