МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
им. Н.Э. Баумана

Факультет “Информатика и системы управления”  
Кафедра “Системы обработки информации и управления”

A logo with a gold star and blue ribbon

Description automatically generated

Дисциплина “Парадигмы и конструкции языков программирования”

Отчет по лабораторной работе №2  
“Объектно-ориентированные возможности языка Python”

**Выполнил:**Студент группы ИУ5-35Б  
Ким А. А.  
**Преподаватель:**Нардид А.Н.

Москва 2025

**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.

**Листинг кода**

**­\_init\_.py:**

from .rectangle import Rectangle  
from .circle import Circle  
from .square import Square  
from .color import Color  
from .figure import Figure  
  
\_\_all\_\_ = ['Rectangle', 'Circle', 'Square', 'Color', 'Figure']

**circle.py:**

from .figure import Figure  
from .color import Color  
import math  
  
class Circle(Figure):  
 def \_\_init\_\_(self, radius, color):  
 super().\_\_init\_\_(color)  
 self.radius = radius  
  
 def area(self):  
 return math.pi \* self.radius \*\* 2  
  
 def perimeter(self):  
 return 2 \* math.pi \* self.radius  
  
 def \_\_repr\_\_(self):  
 return f"Circle(radius={self.radius}, color={self.color.color})"

**color.py:**

class Color:  
 def \_\_init\_\_(self, color):  
 self.\_color = color  
  
 @property  
 def color(self):  
 return self.\_color  
  
 @color.setter  
 def color(self, value):  
 self.\_color = value  
  
 def \_\_repr\_\_(self):  
 return f"Color({self.\_color})"

**figure.py:**

from abc import ABC, abstractmethod  
from .color import Color  
  
  
class Figure(ABC):  
 def \_\_init\_\_(self, color):  
 self.color = color if isinstance(color, Color) else Color(color)  
  
 @abstractmethod  
 def area(self):  
 pass  
  
 @abstractmethod  
 def perimeter(self):  
 pass  
  
 def \_\_repr\_\_(self):  
 return f"{self.\_\_class\_\_.\_\_name\_\_}(color={self.color.color})"

**rectangle.py:**

from .figure import Figure  
from .color import Color  
  
  
class Rectangle(Figure):  
 def \_\_init\_\_(self, width, height, color):  
 super().\_\_init\_\_(color)  
 self.width = width  
 self.height = height  
  
 def area(self):  
 return self.width \* self.height  
  
 def perimeter(self):  
 return 2 \* (self.width + self.height)  
  
 def \_\_repr\_\_(self):  
 return f"Rectangle(width={self.width}, height={self.height}, color={self.color.color})"

**square.py:**

from .rectangle import Rectangle  
  
class Square(Rectangle):  
 def \_\_init\_\_(self, side, color):  
 super().\_\_init\_\_(side, side, color)  
 self.side = side  
  
 def \_\_repr\_\_(self):  
 return f"Square(side={self.side}, color={self.color.color})"

**main.py:**

from lab\_python\_oop.rectangle import Rectangle  
from lab\_python\_oop.circle import Circle  
from lab\_python\_oop.square import Square  
  
  
def main():  
 rectangle = Rectangle(10, 10, "синий")  
 circle = Circle(10, "зеленый")  
 square = Square(10, "красный")  
print("Информация о фигурах:")  
  
 print(f"Прямоугольник: {rectangle}")  
 print(f"Площадь: {rectangle.area()}")  
 print(f"Периметр: {rectangle.perimeter()}")  
 print()  
  
 print(f"Круг: {circle}")  
 print(f"Площадь: {circle.area():.2f}")  
 print(f"Длина окружности: {circle.perimeter():.2f}")  
 print()  
  
 print(f"Квадрат: {square}")  
 print(f"Площадь: {square.area()}")  
 print(f"Периметр: {square.perimeter()}")  
 print()  
figures = [rectangle, circle, square]  
  
 print("Демонстрация полиморфизма:")  
 for figure in figures:  
 area = figure.area()  
 perimeter = figure.perimeter()  
 if isinstance(figure, Circle):  
 print(f"{figure.\_\_class\_\_.\_\_name\_\_}: площадь = {area:.2f}, периметр = {perimeter:.2f}")  
 else:  
 print(f"{figure.\_\_class\_\_.\_\_name\_\_}: площадь = {area}, периметр = {perimeter}")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

**Скриншоты работы программ**

