

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет "Информатика и системы управления"
Кафедра "Системы обработки информации и управления"



Дисциплина "Парадигмы и конструкции языков программирования"

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

Выполнил:
Студент группы ИУ5-35Б
Шаньгин Н.А.
Преподаватель:
Гапанюк Ю.Е.

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](#).
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 "getr", 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

main.py

```
from lab_python_oop.__init__ import RectangularShape2D, SquareShape2D, CircleShape2D
from colorama import init

def main():
    rectangle = RectangularShape2D(21, 21, "Синий")
    print(rectangle)
    circle = CircleShape2D(21, "Зелёный")
    print(circle)
    square = SquareShape2D(21, "красный")
    print(square)

if __name__ == "__main__":
    init(autoreset=True)
    main()
```

__init__.py

```
from lab_python_oop.shape_2d import Shape2D
from lab_python_oop.rectangular_shape_2d import RectangularShape2D
from lab_python_oop.circle_shape_2d import CircleShape2D
from lab_python_oop.square_shape_2d import SquareShape2D
from lab_python_oop.color_unit import ColorUnit

__all__ = ['Shape2D', 'RectangularShape2D', 'CircleShape2D', 'SquareShape2D', 'ColorUnit']
```

shape_2d.py

```
from abc import ABC, abstractmethod

class Shape2D(ABC):
    @property
    @abstractmethod
    def shape_type(self):
        pass

    @property
    @abstractmethod
    def area(self):
        pass

    @abstractmethod
    def __repr__(self):
        pass

    # @abstractmethod
    # def draw():
    #     pass
```

color_unit.py

```
from colorama import Fore

class ColorUnit():
    COLOR_MAPPING = {
        "КРАСНЫЙ": Fore.RED,
        "СИНИЙ": Fore.BLUE,
        "ЗЕЛЁНЫЙ": Fore.GREEN,
        "ЖЁЛТЫЙ": Fore.YELLOW,
        "МАГНЕТА": Fore.MAGENTA,
        "ГОЛУБОЙ": Fore.CYAN,
        "БЕЛЫЙ": Fore.WHITE,
        "ЧЁРНЫЙ": Fore.BLACK,
        "СЕРЫЙ": Fore.LIGHTBLACK_EX,
        "ФИОЛЕТОВЫЙ": Fore.MAGENTA,
        "БИРЮЗОВЫЙ": Fore.CYAN,
    }

    def __init__(self, color):
        self._color_name = color
        self._value = self._validate_color(color)

    @property
    def color_name(self):
        return self._color_name

    def _validate_color(self, color):
```

```

        if color.upper() in self.COLOR_MAPPING:
            return self.COLOR_MAPPING[color.upper()]
        elif hasattr(Fore, color.upper()):
            return getattr(Fore, color.upper())
        else:
            print(f"Предупреждение: цвет '{color}' не найден. Используется белый цвет.")
            return Fore.WHITE

    def get_value(self):
        return self._value

    def set_value(self, color):
        self._value = self._validate_color(color)

```

rectangular_shape_2d.py

```

from lab_python_oop.shape_2d import Shape2D
from lab_python_oop.color_unit import ColorUnit
from colorama import Fore, Style

class RectangularShape2D(Shape2D):
    def __init__(self, width, height, color):
        self._shape_type = "Прямоугольник"
        self._width = width
        self._height = height
        self._color = ColorUnit(color)

    @property
    def shape_type(self):
        return self._shape_type

    @property
    def area(self):
        return self._width * self._height

    def __repr__(self):
        return '{name}: ширина: {width}, высота: {height}, цвет: {color}'.format(
            name = self.shape_type,
            width = self._width,
            height = self._height,
            color =
f"{self._color.get_value()}{self._color.color_name}{Style.RESET_ALL}"
        )

    # def draw(self):
    #     if self._height == self._width:

```

```

#         letter = "S"
#     else:
#         letter = "R"

#     for i in range(self._height):
#         line = letter * self._width
#         print(Back.RED + line)

```

square_shape_2d.py

```

from lab_python_oop.rectangular_shape_2d import RectangularShape2D
from colorama import Fore, Style

```

```

class SquareShape2D(RectangularShape2D):
    def __init__(self, side_length, color):
        super().__init__(side_length, side_length, color)
        self._shape_type = "Квадрат"

```

circle_shape_2d.py

```

from lab_python_oop.shape_2d import Shape2D
from lab_python_oop.color_unit import ColorUnit
from math import pi
from colorama import Back, Style

```

```

class CircleShape2D(Shape2D):
    def __init__(self, radius, color):
        self._shape_type = "Круг"
        self._radius = radius
        self._color = ColorUnit(color)

    @property
    def shape_type(self):
        return self._shape_type

    @property
    def area(self):
        return pi * self._radius ** 2

    def __repr__(self):
        return '{name}: радиус: {radius}, площадь: {area}, цвет: {color}'.format(
            name = self.shape_type,
            radius = self._radius,
            area = self.area,
            color =
f"{self._color.get_value()}{self._color.color_name}{Style.RESET_ALL}"
        )

    # def draw(self):
    #     for h in range(self._radius):

```

```

#         line = ""
#         for w in range(self._radius):
#             if ((w-sqrt(self._radius)) ** 2 + (h-sqrt(self._radius)) ** 2)
<= self._radius:
#                 line += "C"
#             else:
#                 line += "\s"
#         print(Back.RED + line)

```

Результат выполнения программы

```

// ...
(.venv) PS D:\Учёба\3 семестр\ПикЯП\Shangin-PCPL-Labs-2025\lab_2> Set-ExecutionPolicy -ExecutionPolicy Bypass -Scope Process
>> .venv\Scripts\activate
(.venv) PS D:\Учёба\3 семестр\ПикЯП\Shangin-PCPL-Labs-2025\lab_2> py main.py
Прямоугольник: ширина: 21, высота: 21, цвет: Синий
Круг: радиус: 21, площадь: 1385.4423602330987, цвет: Зелёный
Квадрат: ширина: 21, высота: 21, цвет: Красный
(.venv) PS D:\Учёба\3 семестр\ПикЯП\Shangin-PCPL-Labs-2025\lab_2>

```