Отчет по лабораторной работе №3

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.**
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 («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.

Текст программы

main.py:

```
from lab_python_oop.rectangle import Rectangle
from lab_python_oop.circle import Circle
from lab_python_oop.square import Square
from prettytable import PrettyTable

def main():
    N = 5
    rect = Rectangle(N, N, "blue")
    circle = Circle(N, "green")
    square = Square(N, "red")

    figures = [rect, circle, square]

    table = PrettyTable(["Figure", "Color", "Parameters", "Area"])
    for fig in figures:
        table.add_row([
            fig.figure_name(),
            fig.color.color,
            repr(fig),
            f"{fig.area():.2f}"
        ])
```

```
    print(table)

if __name__ == "__main__":
    main()
```

geometric_figure.py:

```
from abc import ABC, abstractmethod

class GeometricFigure(ABC):
    @abstractmethod
    def area(self):
        pass

    @classmethod
    @abstractmethod
    def figure_name(cls):
        pass
```

color.py:

```
class FigureColor:
    def __init__(self, color: str):
        self._color = color

    @property
    def color(self):
        return self._color

    @color.setter
    def color(self, value: str):
        self._color = value
```

rectangle.py:

```
from geometric_figure import GeometricFigure
from 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):
```

```
        return self.width * self.height

    @classmethod
    def figure_name(cls):
        return "Rectangle"

    def __repr__(self):
        return "{}: width={}, height={}, color={}, area={:.2f}".format(
            self.figure_name(), self.width, self.height, self.color.color, self.area()
        )
```

circle.py:

```
from geometric_figure import GeometricFigure
from 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):
        return math.pi * self.radius**2

    @classmethod
    def figure_name(cls):
        return "Circle"

    def __repr__(self):
        return "{}: radius={}, color={}, area={:.2f}".format(
            self.figure_name(), self.radius, self.color.color, self.area()
        )
```

square.py:

```
from rectangle import Rectangle

class Square(Rectangle):
    def __init__(self, side: float, color: str):
        super().__init__(side, side, color)

    @classmethod
    def figure_name(cls):
```

return "Square"

Примеры выполнения программы

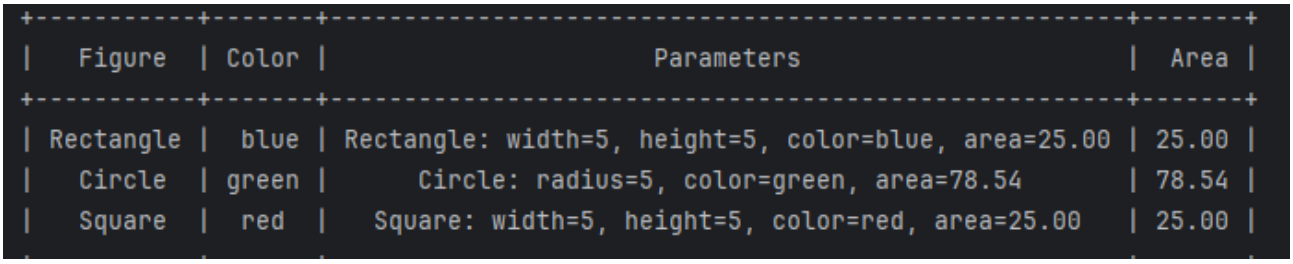

Figure	Color	Parameters	Area
Rectangle	blue	Rectangle: width=5, height=5, color=blue, area=25.00	25.00
Circle	green	Circle: radius=5, color=green, area=78.54	78.54
Square	red	Square: width=5, height=5, color=red, area=25.00	25.00

Вывод

Я изучил объектно-ориентированные возможности языка Python