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»

Лабораторная работа 2

по дисциплине «Базовые компоненты интернет-технологий»

«Объектно-ориентированные возможности языка Python»

Выполнил:

Студент группы ИУ5-34Б

Хатин М.С.

2021 г.

Постановка задачи

Необходимо создать виртуальное окружение и установить в него хотя бы один внешний пакет с использованием pip.

1.Необходимо разработать программу, реализующую работу с классами. Программа должна быть разработана в виде консольного приложения на языке Python 3.

2.Все файлы проекта (кроме основного файла main.py) должны располагаться в пакете lab_python_oop.

3.Каждый из нижеперечисленных классов должен располагаться в отдельном файле пакета lab_python_oop.

4.Абстрактный класс «Геометрическая фигура» содержит абстрактный метод для вычисления площади фигуры. Подробнее про абстрактные классы и методы Вы можете прочитать здесь.

5.Класс «Цвет фигуры» содержит свойство для описания цвета геометрической фигуры. Подробнее про описание свойств Вы можете прочитать здесь.

6.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.

7.Класс «Круг» создается аналогично классу «Прямоугольник», задается параметр «радиус». Для вычисления площади используется константа math.pi из модуля math.

8.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.

9.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

Файл main.py

```
from lab_python_oop.rectangle import Rectangle
from lab_python_oop.circle import Circle
from lab_python_oop.square import Square
import plotly.graph_objects as go  # рисует продвинутые графики, иногда используется в VDA

if __name__ == '__main__':
    print("Введите число N: ", end=' ')
    n = int(input())
    rec = Rectangle(n, n, 'blue')
    circle = Circle(n, 'green')
    sqr = Square(n, 'red')

    fig = go.Figure()

    # Выставим одинаковый масштаб по всем осям, иначе окружность будет эллипсом, а квадрат - не квадратом
    fig.update_yaxes(
        scaleanchor="x",
        scaleratio=1,
    )

    # квадрат
    fig.add_shape(type="rect",
                  xref='x', yref='y',
                  x0=3 * n + 2, y0=0, x1=4 * n + 2, y1=n,  # координаты вершин прямоугольника,расположенных по диагонали
                  line=dict(color='red', width=8),
                  fillcolor='coral'
                  )
    # прямоугольник
    fig.add_shape(type="rect",
                  xref='x', yref='y',
                  x0=2*n+1, y0=0, x1=3*n+1, y1=n,  # координаты вершин прямоугольника, расположенных по диагонали
                  line=dict(color='blue', width=5),
                  fillcolor='Royalblue'
                  )
    # окружность
    fig.add_shape(type="circle",
                  xref="x", yref="y",
                  x0=0, y0=0, x1=2*n, y1=2*n,  # координаты вершин квадрата, описанного около окружности
                  line_color="green",
                  )
    fig.add_trace(go.Scatter(
        x=[n, 2.5*n + 1, 3.5*n + 2],
        y=[n, n+.25, n+.25],
        text=["Окружность",
              "Прямоугольник",
              "Квадрат"],
        mode="text",
    ))


    # А ещё можно так
    fig1 = go.Figure(
        go.Scatter(x=[2*n+1, 3*n+1, 3*n+1, 2*n+1, 2*n+1, None, 3*n+2, 4*n+2, 4*n+2, 3*n+2, 3*n+2],
                   y=[0, 0, n, n, 0,  None, 0, 0, n, n, 0],
```

```
                fill="toself"))

    fig1.update_yaxes(
        scaleanchor="x",
        scaleratio=1,
    )

    fig1.show()
    fig.show()
```

Файл GeometricFigure.py

```
from abc import ABC, abstractmethod


class Figure(ABC):
    FIGURE_TYPE = 'Фигура'  # можно же наследовать статические переменные?
    '''Пример абстрактного класса'''

    # Вроде для user-friendly вывода переопределяют str, а не repr, как требуется в задании?
    def __str__(self):
        return 'geometric figure'

    @abstractmethod
    def square(self):
        pass

    @classmethod
    def get_type(cls):
        return cls.FIGURE_TYPE
```

Файл circle.py

```
from math import pi
from lab_python_oop.GeometricFigure import Figure
from lab_python_oop.color import Color


class Circle(Figure):
    """r - радиус"""

    FIGURE_TYPE = 'Круг'

    def __init__(self, r, color):
        self._r, self._color = r, Color(color)

    def square(self):
        return pi * self._r ** 2

    def __str__(self):
        return f'{self.FIGURE_TYPE} радиуса {self._r} цвета {self._color}'
```

Файл square.py

```
from lab_python_oop.rectangle import Rectangle

class Square(Rectangle):

    FIGURE_TYPE = 'Квадрат'

    def __init__(self, side_length, color):
        """ a - длина стороны квадрата"""
        self._side = side_length
        super().__init__(side_length, side_length, color)

    def __str__(self):
        return f'{self.FIGURE_TYPE} стороной {self._side} цвета {self._color}'
```

Файл color.py

```
class Color:
    def __init__(self, color_str=None):
        self._color = color_str

    def set_color(self, color):
        self._color = color

    def get_color(self):
        return self._color

    def del_color(self):
        del self._color

    color = property(get_color, set_color, del_color)
 # В чём разница, использовать конструкцию с @property или с property() ?
```

Файл rectangle.py

```
from lab_python_oop.GeometricFigure import Figure
from lab_python_oop.color import Color


class Rectangle(Figure):
    '''h - высота, w - ширина'''

    FIGURE_TYPE = 'Прямоугольник'

    def __init__(self, h, w, color):
        self._h, self._w, self._color = h, w, Color(color)

    def square(self):
        return self._h * self._w

    def __str__(self):
        return f'{self.FIGURE_TYPE} высотой {self._h} и шириной {self._w} цвета {self._color}'
```

Результат выполнения программы

```
/home/maxim/PycharmProjects/default/bin/python /home/maxim/PycharmProjects/lab_02[BKIT]/main.py
Введите число N:  21

Process finished with exit code 0
```