Московский государственный технический университет им. Н.Э. Баумана

Факультет ИУ
Кафедра ИУ5

Курс «Основы информатики»
Отчет по лабораторной работе №5(2*)

Выполнил студент группы ИУ5-33Б:
Хасанова К.М.
Подпись и дата:

Проверил преподаватель каф.:
Гапанюк Ю. Е.
Подпись и дата:

Москва, 2024 г

Описание задания

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

figure.py

```
from abc import ABC, abstractmethod

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

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

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
```

rectangle.py

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

class Rectangle(Figure):
    FIGURE_TYPE = "Прямоугольник"

    def __init__(self, width, height, color):
```

```
        self.width = width
        self.height = height
        self.color = Color(color)

    def area(self):
        return self.width * self.height

    @classmethod
    def name(cls):
        return cls.FIGURE_TYPE

    def __repr__(self):
        return "Фигура: {}, цвет: {}, ширина: {}, высота: {}, площадь:
{:.2f}".format(
            self.name(), self.color.color, self.width, self.height, self.area()
        )
```

square.py

```
from lab_python_oop.rectangle import Rectangle

class Square(Rectangle):
    FIGURE_TYPE = "Квадрат"

    def __init__(self, side, color):
        super().__init__(side, side, color)

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

circle.py

```
from lab_python_oop.figure import Figure
from lab_python_oop.color import Color
import math

class Circle(Figure):
    FIGURE_TYPE = "Круг"

    def __init__(self, radius, color):
        self.radius = radius
        self.color = Color(color)

    def area(self):
        return math.pi * (self.radius ** 2)

    @classmethod
    def name(cls):
        return cls.FIGURE_TYPE

    def __repr__(self):
        return "Фигура: {}, цвет: {}, радиус: {}, площадь: {:.2f}".format(
            self.name(), self.color.color, self.radius, self.area()
        )
```

main.py

```python
from lab_python_oop.rectangle import Rectangle
from lab_python_oop.circle import Circle
from lab_python_oop.square import Square
import numpy as np


def create_shapes(n):

   # Создает три объекта: прямоугольник, круг и квадрат.

    rect = Rectangle(n, n, "синий")
    circle = Circle(n, "зеленый")
    square = Square(n, "красный")
    return rect, circle, square


def use_numpy_example():

    # Пример использования numpy: сумма элементов массива.

    arr = np.array([1, 2, 3, 4, 5])
    return np.sum(arr)


def main():
    n = 22
    # Создание объектов фигур
    rect, circle, square = create_shapes(n)

    # Вывод информации о фигурах
    print(rect)
    print(circle)
    print(square)
```

test_main.py

```python
import unittest
from lab_python_oop.rectangle import Rectangle
from lab_python_oop.circle import Circle
from lab_python_oop.square import Square
from main import create_shapes, use_numpy_example

class TestShapes(unittest.TestCase):

    def test_rectangle_area(self):
        """Тест площади прямоугольника"""
        rect = Rectangle(4, 5, "синий")
        self.assertEqual(rect.area(), 20, "Площадь прямоугольника должна быть
20")

    def test_circle_area(self):
        """Тест площади круга"""
        circle = Circle(3, "зеленый")
        self.assertAlmostEqual(circle.area(), 28.27, places=2, msg="Площадь
круга должна быть примерно 28.27")

    def test_square_area(self):
        """Тест площади квадрата"""
        square = Square(4, "красный")
        self.assertEqual(square.area(), 16, "Площадь квадрата должна быть 16")
```

```
class TestNumpyExample(unittest.TestCase):

    def test_numpy_sum(self):
        """Тест функции использования numpy"""
        result = use_numpy_example()
        self.assertEqual(result, 15, "Сумма элементов массива должна быть 15")

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

features/shape_feature

```
Feature: Проверка работы фигур

  Scenario: Проверка площади прямоугольника
    Given I have a rectangle with width 4 and height 5
    Then the area of the rectangle should be 20

  Scenario: Проверка площади круга
    Given I have a circle with radius 3
    Then the area of the circle should be approximately 28.27

  Scenario: Проверка площади квадрата
    Given I have a square with side 4
    Then the area of the square should be 16
```

test_shapes_bdd.py

```
from pytest_bdd import scenarios, given, then
from lab_python_oop.rectangle import Rectangle
from lab_python_oop.circle import Circle
from lab_python_oop.square import Square

# Подключение файла feature
scenarios('features/shapes.feature')

# Шаги

@given("I have a rectangle with width 4 and height 5")
def rectangle():
    return Rectangle(4, 5, "синий")

@then("the area of the rectangle should be 20")
def check_rectangle_area(rectangle):
    assert rectangle.area() == 20

@given("I have a circle with radius 3")
def circle():
    return Circle(3, "зеленый")

@then("the area of the circle should be approximately 28.27")
def check_circle_area(circle):
    assert round(circle.area(), 2) == 28.27

@given("I have a square with side 4")
def square():
    return Square(4, "красный")

@then("the area of the square should be 16")
```

```python
def check_square_area(square):
    assert square.area() == 16
```

Выполнение программы

```
(venv) PS C:\Users\Администратор\PycharmProjects\
....
----------------------------------------------------
Ran 4 tests in 0.002s

OK
```

```
============================= test session starts ==============================
collecting ... collected 4 items

test_main.py::TestShapes::test_circle_area PASSED                        [ 25%]
test_main.py::TestShapes::test_rectangle_area PASSED                     [ 50%]
test_main.py::TestShapes::test_square_area PASSED                        [ 75%]
test_main.py::TestNumpyExample::test_numpy_sum PASSED                    [100%]

============================== 4 passed in 0.15s ===============================
```