Московский государственный технический
университет им. Н.Э. Баумана

Факультет информатика и системы управления
Кафедра системы обработки информации и управления

Курс «Парадигмы и конструкции языков программирования»
Отчет по лабораторной работы №2

Выполнил:
студент группы ИУ5-32Б:
Шпакова Д. Л.
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Нардид А.Н.
Подпись и дата:

Москва, 2025 г.

Описание задания

Цель лабораторной работы: изучение объектно-ориентированных возможностей языка

Задание:

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

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

се файлы проекта (кроме основного файла main.py) должны располагаться в пакете

аждый из нижеперечисленных классов должен располагаться в отдельном файле пакета lab_python_oop.

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

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

ласс «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.

ласс «Круг» создается аналогично классу «Прямоугольник», задается параметр «

ласс «Квадрат» наследуется от класса «Прямоугольник». Класс должен содержать конструктор по длине стороны. Для классов «Прямоугольник», «Квадрат», «Круг»:

- пределите метод "repr", который возвращает в виде строки основные
- азвание фигуры («Прямоугольник», «Квадрат», «Круг») должно задаваться в виде поля данных класса и возвращаться методом класса.

корневом каталоге проекта создайте файл main.py для тестирования Ваших классов (используйте следующую конструкцию

ыведите о них информацию в консоль (N - номер Вашего варианта по списку группы):

- рямоугольник синего цвета шириной N и высотой N.

руг зеленого цвета радиусом N.

вадрат красного цвета со стороной N.

акже вызовите один из методов внешнего пакета, установленного с использованием pip.

ополнительное задание. Протестируйте корректность работы Вашей программы с помощью модульного теста.

Текст программы

m
a

```
from lab2.rectangle import Rectangle
from lab2.circle import Circle
from lab2.square import Square
import numpy as np

def main():
    # Установим N (например, номер по списку)
    N = 10

    # Создаем объекты
    rect = Rectangle(N, N, "синего")
    circ = Circle(N, "зеленого")
    sq = Square(N, "красного")

    # Выводим информацию об объектах
    print(rect)
    print(circ)
    print(sq)

    # Демонстрация использования numpy для вычислений с фигурами
    print("\n" + "="*50)
    print("Использование NumPy для анализа фигур:")
    print("="*50)

    # Создаем массив площадей фигур с помощью numpy
    areas = np.array([
        rect.calculate_area(),
        circ.calculate_area(),
        sq.calculate_area()
    ])

    # Создаем массив названий фигур
    names = np.array([rect.get_name(), circ.get_name(), sq.get_name()])

    # print(f"Площади фигур: {areas}")
    print(f"Средняя площадь: {np.mean(areas):.2f}")
    print(f"Максимальная площадь: {np.max(areas):.2f}")
    print(f"Минимальная площадь: {np.min(areas):.2f}")

    # Находим фигуру с максимальной площадью
    max_area_index = np.argmax(areas)
    print(f"Фигура с наибольшей площадью: {names[max_area_index]}
({areas[max_area_index]:.2f})")

    # Сумма всех площадей
    print(f"Сумма всех площадей: {np.sum(areas):.2f}")
```

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

`test_shapes.py`

```
import unittest
import math

from lab2.rectangle import Rectangle
from lab2.circle import Circle
from lab2.square import Square

class TestShapeAreas(unittest.TestCase):

    def test_rectangle_area(self):
        # Тестирование вычисления площади прямоугольника.
        rect = Rectangle(5, 10, "синий")
        self.assertEqual(rect.calculate_area(), 50)

    def test_circle_area(self):
        # Тестирование вычисления площади круга.
        circ = Circle(7, "зеленый")
        # Используем assertAlmostEqual для сравнения чисел с плавающей запятой.
        self.assertAlmostEqual(circ.calculate_area(), math.pi * (7 ** 2))

    def test_square_area(self):
        # Тестирование вычисления площади квадрата.
        sq = Square(6, "красный")
        self.assertEqual(sq.calculate_area(), 36)

    def test_square_inheritance(self):
        # Тестирование того, что у квадрата ширина и высота равны.
        sq = Square(8, "желтый")
        self.assertEqual(sq.width, 8)
        self.assertEqual(sq.height, 8)

# Это позволяет запускать тесты напрямую из этого файла.
if __name__ == "__main__":
    unittest.main()
```

`lab_2/circle.py`

```
import math
import numpy as np
from .geometric_figure import GeometricFigure
from .figure_color import FigureColor


class Circle(GeometricFigure):
    """Класс для представления круга."""
    name = "Круг"
```

```
    def __init__(self, radius, color):
        self.radius = radius
        self.figure_color = FigureColor(color)

    def calculate_area(self):
        """Вычисляет площадь круга с использованием numpy для точности."""
        return np.pi * (self.radius ** 2)

    def calculate_circumference(self):
        """Вычисляет длину окружности с использованием numpy."""
        return 2 * np.pi * self.radius

    def get_numpy_array(self):
        """Возвращает параметры круга как numpy array."""
        return np.array([self.radius, self.calculate_area(),
self.calculate_circumference()])

    def __repr__(self):
        return "Фигура: {}. Радиус: {}. Цвет: {}. Площадь: {:.2f}. Длина
окружности: {:.2f}.".format(
            self.get_name(),
            self.radius,
            self.figure_color.color,
            self.calculate_area(),
            self.calculate_circumference()
        )

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

`lab_2/figure_color.py`

```
class FigureColor:
    """Класс для хранения цвета фигуры."""

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

    @property
    def color(self):
        """Свойство для получения цвета."""
        return self._color

    @color.setter
    def color(self, value):
        """Сеттер для установки цвета."""
        self._color = value
```

`lab_2/geometric_figure.py`

```
from abc import ABC, abstractmethod


class GeometricFigure(ABC):
    """Абстрактный класс для геометрических фигур."""

    @abstractmethod
    def calculate_area(self):
        """Абстрактный метод для вычисления площади фигуры."""
        pass
```

`lab_2/rectangle.py`

```
import numpy as np
from .geometric_figure import GeometricFigure
from .figure_color import FigureColor


class Rectangle(GeometricFigure):
    """Класс для представления прямоугольника."""

    name = "Прямоугольник"

    def __init__(self, width, height, color):
        self.width = width
        self.height = height
        self.figure_color = FigureColor(color)

    def calculate_area(self) -> float:
        """Вычисляет площадь прямоугольника."""
        return self.width * self.height

    def calculate_perimeter(self):
        """Вычисляет периметр прямоугольника."""
        return 2 * (self.width + self.height)

    def get_numpy_array(self):
        """Возвращает параметры прямоугольника как numpy array."""
        return np.array([self.width, self.height, self.calculate_area(),
self.calculate_perimeter()])

    def get_diagonal(self):
        """Вычисляет диагональ прямоугольника с использованием numpy."""
        return np.sqrt(self.width**2 + self.height**2)

    def __repr__(self):
        return "Фигура: {}. Ширина: {}. Высота: {}. Цвет: {}. Площадь: {}.
Периметр: {}. Диагональ: {:.2f}.".format(
            self.get_name(),
```

```
            self.width,
            self.height,
            self.figure_color.color,
            self.calculate_area(),
            self.calculate_perimeter(),
            self.get_diagonal()
        )

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

`lab_2/square.py`

```
import numpy as np
from .rectangle import Rectangle


class Square(Rectangle):
    """Класс для представления квадрата, наследуется от Прямоугольника."""

    name = "Квадрат"

    def __init__(self, side, color):
        """Вызываем конструктор родительского класса (Rectangle)."""
        super().__init__(side, side, color)
        self.side = side

    def get_numpy_array(self):
        """Возвращает параметры квадрата как numpy array."""
        return np.array([self.side, self.calculate_area(),
self.calculate_perimeter(), self.get_diagonal()])

    def __repr__(self):
        return "Фигура: {}. Сторона: {}. Цвет: {}. Площадь: {}. Периметр: {}.
Диагональ: {:.2f}.".format(
            self.get_name(),
            self.side,
            self.figure_color.color,
            self.calculate_area(),
            self.calculate_perimeter(),
            self.get_diagonal()
        )

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

Скриншот работы приложения

```
(.venv) dasha@Magic14:~/LABS_2025_3TERM/LABS_2025_3TERM-1/lab2$ python main.py
Фигура: Прямоугольник. Ширина: 10. Высота: 10. Цвет: синего. Площадь: 100. Периметр: 40. Диагональ: 14.14.
Фигура: Круг. Радиус: 10. Цвет: зеленого. Площадь: 314.16. Длина окружности: 62.83.
Фигура: Квадрат. Сторона: 10. Цвет: красного. Площадь: 100. Периметр: 40. Диагональ: 14.14.

==================================================
Использование NumPy для анализа фигур:
==================================================
Средняя площадь: 171.39
Максимальная площадь: 314.16
Минимальная площадь: 100.00
Фигура с наибольшей площадью: Круг (314.16)
Сумма всех площадей: 514.16
```

Рисунок 1 Результат работы main.py

```
(.venv) dasha@Magic14:~/LABS_2025_3TERM/LABS_2025_3TERM-1/lab2$ python3 -m unittest tests/test_shapes.py
....
----------------------------------------------------------------------
Ran 4 tests in 0.001s

OK
```

Рисунок 2 Тесты геометрических фигур