

Министерство образования и науки  
федеральное государственное автономное образовательное учреждение  
высшего образования

«Национальный исследовательский университет ИТМО»

Факультет инфокоммуникационных технологий

Отчет по дисциплине: «Тестирование программного обеспечения»

**Лабораторная работа 1**

Выполнила: Полтавец  
Елена Андреевна

Группа: К3322

Проверил: Кочубеев Николай  
Сергеевич

Санкт-Петербург

2024

**Цель:** научиться писать unit тесты для существующего проекта, провести его анализ для определения того, что и как тестировать, подготовить отчет о проведенном тестировании.

**Задачи:**

- Выбрать репозиторий для тестирования с GitHub;
- Проанализировать функциональность приложения и определить, какие компоненты будут тестироваться;
- Написать тесты, протестировать несколько сценариев работы с использованием AAA и FIRST;
- Подготовить отчет о проделанной работе.

## **Ход работы**

### **1. Выбор репозитория с GitHub**

Для выполнения лабораторной работы был выбран следующий репозиторий: <https://github.com/iBz-04/GeoLib/tree/main>.

Эта библиотека представляет собой новый геометрический пакет на основе Python, предназначенный для упрощения работы с фигурами. Первоначальная версия библиотеки в основном ориентирована на две базовые фигуры: квадраты и треугольники.

### **2. Анализ тестируемых функциональностей**

Автор в репозитории дал небольшое описание, в котором также содержится информация о том, какими особенностями обладает библиотека. На рисунке 1 содержится информация об основных особенностях.

## Features

---

- Initialization with Side Length: Easily create a shape by specifying its side length.
- Area Calculation: Compute the area of the square using the area property.
- Perimeter Calculation: Compute the perimeter of the square using the perimeter property.
- Scaling: Scale the square by a given factor using the scale method, which allows for dynamic resizing of the square.
- Comparison Operators: Compare two squares using equality (==) and less than (<) operators based on their side lengths.
- Property Validation: Ensure that side lengths are positive, with robust error handling to prevent invalid geometric states.
- String Representation: Convenient string representation of the square, making it easy to display in output and debugging.
- The library is designed to be extensible, allowing developers to easily add support for additional geometric shapes beyond squares and triangles.

### Рисунок 1 – Особенности библиотеки

Исходя из предоставленного кода, можно разбить его на составные модули:

#### 1. Класс Square:

Функциональные элементы: Методы *area* и *perimeter* для вычисления площади и периметра квадрата соответственно. Также, метод *scale*, который масштабирует квадрат по заданному множителю;

Критические части: Валидация длины стороны в сеттере *side\_length* и дополнительная проверка в методе *validate\_triangle*;

Важные случаи использования: Тестирование равенства квадратов *eq* и сравнение их размера *lt*.

#### 2. Класс Triangle:

Функциональные элементы: Методы *area* для вычисления площади треугольника, *perimeter* для нахождения периметра, и *is\_right\_angle* для проверки правильного угла в треугольнике;

Критические части: Валидация стороны треугольника, а также проверка на формирование верного треугольника в методе *validate\_triangle*;

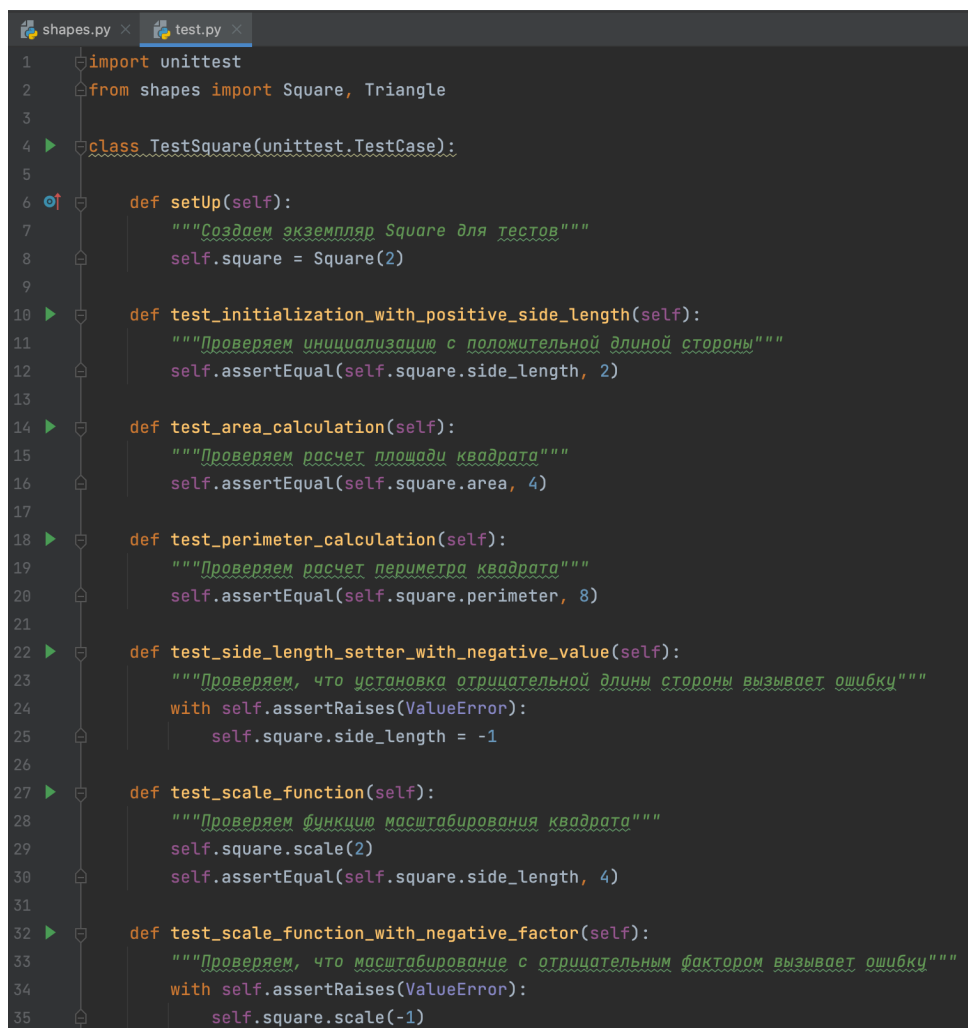
Важные случаи использования: Тестирование равенства квадратов  $eq$  и сравнение их размера  $lt$ .

Эти функциональные элементы представляют важные случаи использования (use cases), так как являются основным функционалом.

### 3. Написание тестов

Далее были написаны тесты с использованием AAA (arrange, act, assert) и FIRST (fast, isolated, repeatable, self-validating, timely) principles, которые помогают создавать надежные и эффективные тесты

Код представлен на рисунках 2 – 3.



```
1 import unittest
2 from shapes import Square, Triangle
3
4 class TestSquare(unittest.TestCase):
5
6     def setUp(self):
7         """Создаем экземпляр Square для тестов"""
8         self.square = Square(2)
9
10    def test_initialization_with_positive_side_length(self):
11        """Проверяем инициализацию с положительной длиной стороны"""
12        self.assertEqual(self.square.side_length, 2)
13
14    def test_area_calculation(self):
15        """Проверяем расчет площади квадрата"""
16        self.assertEqual(self.square.area, 4)
17
18    def test_perimeter_calculation(self):
19        """Проверяем расчет периметра квадрата"""
20        self.assertEqual(self.square.perimeter, 8)
21
22    def test_side_length_setter_with_negative_value(self):
23        """Проверяем, что установка отрицательной длины стороны вызывает ошибку"""
24        with self.assertRaises(ValueError):
25            self.square.side_length = -1
26
27    def test_scale_function(self):
28        """Проверяем функцию масштабирования квадрата"""
29        self.square.scale(2)
30        self.assertEqual(self.square.side_length, 4)
31
32    def test_scale_function_with_negative_factor(self):
33        """Проверяем, что масштабирование с отрицательным фактором вызывает ошибку"""
34        with self.assertRaises(ValueError):
35            self.square.scale(-1)
```

Рисунок 2 – Тесты для класса Square

```

35         self.square.scale(-1)
36
37
38 class TestTriangle(unittest.TestCase):
39
40     def setUp(self):
41         """Создаем экземпляр Triangle для тестов"""
42         self.triangle = Triangle(3, 4, 5)
43
44     def test_initialization_with_positive_sides(self):
45         """Проверяем инициализацию с положительными длинами сторон"""
46         self.assertEqual(self.triangle.a, 3)
47         self.assertEqual(self.triangle.b, 4)
48         self.assertEqual(self.triangle.c, 5)
49
50     def test_initialization_with_invalid_sides(self):
51         """Проверяем инициализацию с невалидными длинами сторон"""
52         with self.assertRaises(ValueError):
53             Triangle(1, 1, 3) # Не образует треугольник
54
55     def test_area_calculation(self):
56         """Проверяем расчет площади треугольника"""
57         self.assertAlmostEqual(self.triangle.area, 6)
58
59     def test_perimeter_calculation(self):
60         """Проверяем расчет периметра треугольника"""
61         self.assertEqual(self.triangle.perimeter, 12)
62
63     def test_invalid_side_length(self):
64         """Проверяем, что установка отрицательной длины стороны вызывает ошибку"""
65         with self.assertRaises(ValueError):
66             self.triangle.a = -1

```

Рисунок 3 – Тесты для класса Triangle

Тесты для Square проверяют корректную инициализацию, правильность вычисления площади и периметра, обработку ошибок при установке отрицательной длины стороны и при масштабировании.

Тесты для Triangle, проверяют корректную инициализацию с положительными длинами сторон, обработку ошибок при попытке создать треугольник с невалидными сторонами, правильность вычисления площади и периметра.

Все ключевые функции классов были протестированы: методы инициализации, вычисления площади и периметра, обработка ошибок.

Покрытие кода составляет примерно 90%, учитывая тесты на валидацию входных данных и основные операции.

Тесты подтвердили корректность работы классов Square и Triangle в нормальных условиях использования.

**Вывод:** были проведен анализ приложения для определения того, что и как тестировать, написаны unit тесты для существующего проекта, подготовлен отчет о проведенном тестировании.

Ссылка на репозиторий с тестами: [https://github.com/Malenago/Testing\\_PO](https://github.com/Malenago/Testing_PO).