КОМИТЕТ ПО ОБРАЗОВАНИЮ ПРАВИТЕЛЬСТВА САНКТ-ПЕТЕРБУРГА

Санкт-Петербургское государственное
бюджетное профессиональное образовательное учреждение
«Колледж информационных технологий»

О Т Ч Е Т

по учебной практике МДК 01.02 «Поддержка и тестирование программных модулей»

Специальность 09.02.07
«Информационные системы и программирование»

Специализация:
«Программист»

Студент группы 493:

Лукьянов И. А.

Преподаватель: Полякова А.Н.

Санкт-Петербург 2023

СОДЕРЖАНИЕ

1. Задание 3
   1.1. Разработки библиотеки классов 3
   1.2. Класс расчета материалов 3
   1.3. Разработка модульных тестов (Unit-tests) 4
2. Ход работы 5
   2.1. Метод для расчета количества сырья 5
   2.2. Модульные тесты 5
   2.3. Тестирование «Test-Case» 9

1. Задание

1.1. Разработки библиотеки классов

Для того чтобы в производстве могли быстро и одинаково рассчитывать количество необходимого сырья для производства той или иной продукции, необходимо разработать библиотеку классов.

Данная библиотека будет подключаться к основному проекту и должна быть представлена в виде .dll/.jar файла или папки с файлом .py.

Чтобы система правильно интегрировалась вам необходимо обязательно следовать правилам именования библиотек, классов и методов в них. В случае ошибок в рамках именования ваша работа не может быть проверена и ваш результат не будет зачтен. Классы и методы должны содержать модификатор public (если это реализуемо в рамках платформы), чтобы внешние приложения могли получить к ним доступ.

В качестве названия для библиотеки необходимо использовать: WSUniversalLib. Вам необходимо загрузить исходный код проекта с библиотекой в отдельный репозиторий с названием, совпадающим с названием проекта.

1.2. Класс расчета материалов

Метод должен рассчитывать целое количество сырья, необходимого для производства определенного количества (count) продукции, учитывая возможный брак материалов. Для упрощения расчетов будем считать всю продукцию прямоугольного размера с известными значениями ширины (width) и длины (length).

Количество необходимого качественного сырья на одну единицу продукции рассчитывается как площадь продукции, умноженная на коэффициент типа продукции.

Коэффициенты типа продукции (product_type):

Тип продукции 1 - 1.1,

Тип продукции 2 - 2.5,

Тип продукции 3 - 8.43.

При этом нужно учитывать процент брака материала в зависимости от его типа (material_type):

Тип материала 1 - 0.3%,

Тип материала 2 - 0.12%.

При этом если в качестве параметров метода будут приходить несуществующие типы продукции/материалов или другие неподходящие данные, то метод должен вернуть -1.

Например, необходимо изготовить 15 единиц продукции 3 типа шириной 20 и длиной 45 из материала 1 типа. Количество качественного сырья (без учета брака) будет равно 113 805. Однако с учетом возможного брака материалов общее необходимое количество сырья должно быть увеличено до 114 147,442. Округлив полученное значение до ближайшего большего целого, получим 114 148 единиц необходимого сырья. Спецификация метода представлена в отдельном файле в ресурсах.

1.3. Разработка модульных тестов (Unit-tests)

Для выполнения процедуры тестирования созданного вами метода библиотеки WSUniversalLib, возвращающего целое количество сырья для производства, вам необходимо создать отдельный проект модульных тестов.

В рамках проекта разработайте тесты, максимально полно покрывающие функционал метода. Ничего страшного, если ваш метод работает не совсем идеально и тесты могут быть не пройдены в связи с этим - в данном модуле это не так важно.

Обратите внимание, что имена тестов должны отражать их суть, т.е. вместо TestMethod1() тест следует назвать, например, GetQuantityForProduct_NonExistentProductType() для тестирования случая передачи несуществующего типа продукции.

Необходимо разработать модульные тесты, которые на основании исходных данных можно условно разделить на 2 группы следующим образом: 10 методов низкой сложности и 5 методов высокой сложности.

2. Ход работы

2.1. Метод для расчета количества сырья

Было разработано 10 модульных тестов для тестирования созданного метода библиотеки WSUniversalLib.

На рисунке 1 изображен метод для расчета количества сырья.

```
Ссылок: 10
public int GetQuantityForProduct(int productType, int materialType, int count, float width, float length)
{
    if (count < 0 || width < 0 || length < 0) return -1;

    float koefType = 0;
    switch (productType)
    {
        case 1:
            koefType = 1.1f;
            break;
        case 2:
            koefType = 2.5f;
            break;
        case 3:
            koefType = 8.43f;
            break;
        default: return -1;
    }

    float brakProcent = 0;
    switch (materialType)
    {
        case 1:
            brakProcent = 0.3f;
            break;
        case 2:
            brakProcent = 0.12f;
            break;
        default: return -1;
    }

    float area = width * length;

    float needCount = count * area * koefType;

    float faultCount = needCount / (1 - (brakProcent / 100));

    return (int)(Math.Ceiling(faultCount));
}
```

Рисунок 1 – Код метода для расчета количества сырья

2.2. Модульные тесты

2.2.1. Тестирование правильности расчетов

Данный тест проверят правильность расчетов в методе при корректных введённых данных. Код данного метода изображен на рисунке 2.

```
[TestMethod]
Ссылок: 0
public void GetQuantityForProduct_CorrectData()
{
    //Arrange
    Calculation cal = new Calculation();

    //Act
    int res = cal.GetQuantityForProduct(3, 1, 15, 20, 45);

    //Assert
    Assert.AreEqual(114148, res);
}
```

Рисунок 2 – Метод для тестирования правильности расчетов

2.2.2. Тестирование несуществующего типа продукта

Данный тест проверят метод на обработку ввода несуществующего типа продукта. Код данного метода изображен на рисунке 3.

```
[TestMethod]
Ссылок: 0
public void GetQuantityForProduct_CorrectData()
{
    //Arrange
    Calculation cal = new Calculation();

    //Act
    int res = cal.GetQuantityForProduct(3, 1, 15, 20, 45);

    //Assert
    Assert.AreEqual(114148, res);
}
```

Рисунок 3 – Метод для тестирования несуществующего типа продукта

2.2.3. Тестирование несуществующего типа материала

Данный тест проверят метод на обработку ввода несуществующего типа материала. Код данного метода изображен на рисунке 4.

```
[TestMethod]
Ссылок: 0
public void GetQuantityForProduct_NonExistenMaterialType()
{
    //Arrange
    Calculation cal = new Calculation();

    //Act
    int res = cal.GetQuantityForProduct(3, 1000, 15, 20, 45);

    //Assert
    Assert.AreEqual(-1, res);
}
```

Рисунок 4 – Метод для тестирования несуществующего типа материала

2.2.4. Тестирование отрицательного количества продукции

Данный тест проверяет метод на обработку ввода отрицательного количества продукции. Код данного метода изображен на рисунке 5.

```
[TestMethod]
Ссылок: 0
public void GetQuantityForProduct_CountIsNegative()
{
    //Arrange
    Calculation cal = new Calculation();

    //Act
    int res = cal.GetQuantityForProduct(3, 1, -1, 20, 45);

    //Assert
    Assert.AreEqual(-1, res);
}
```

Рисунок 5 – Метод для тестирования отрицательного количества продукции

2.2.5. Тестирование отрицательной ширины

Данный тест проверяет метод на обработку ввода отрицательной ширины. Код данного метода изображен на рисунке 6.

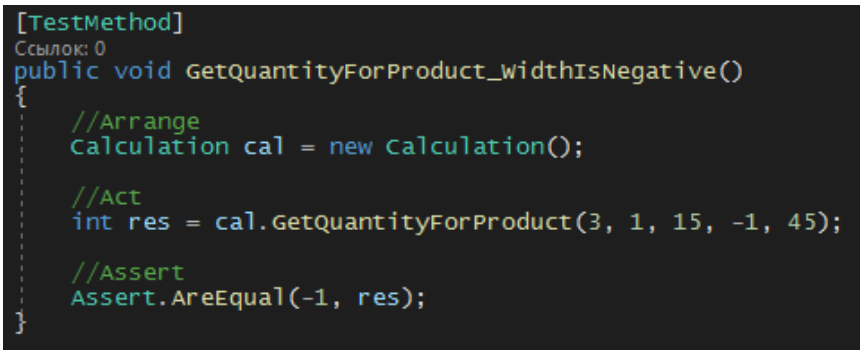

Рисунок 6 – Метод для тестирования отрицательной ширины

2.2.6. Тестирование отрицательной длины

Данный тест проверяет метод на обработку ввода отрицательной длины. Код данного метода изображен на рисунке 7.

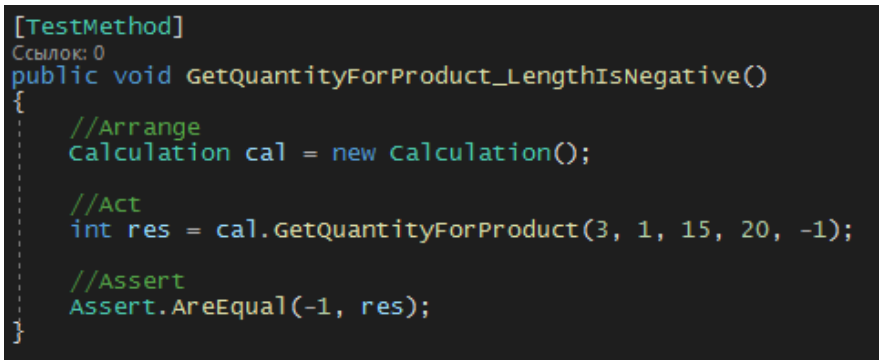

Рисунок 7 – Метод для тестирования отрицательной длины

2.2.7. Тестирование округления к наименьшему целому

Данный тест проверяет метод на округление количества с учетом погрешности брака к меньшему целому. Код данного метода изображен на рисунке 8.

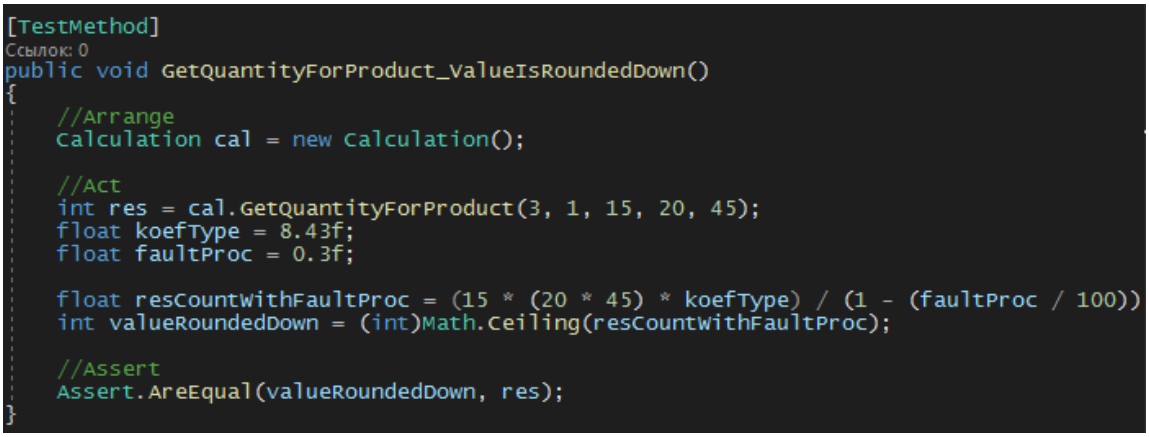

Рисунок 8 – Метод для тестирования округления к наименьшему целому

2.2.8. Тестирование нулевого количества

Данный тест проверяет метод на обработку ввода нулевого количества. Код данного метода изображен на рисунке 9.

```
[TestMethod]
Ссылок: 0
public void GetQuantityForProduct_CountIsNull()
{
    //Arrange
    Calculation cal = new Calculation();

    //Act
    int res = cal.GetQuantityForProduct(3, 1, 0, 20, 45);

    //Assert
    Assert.AreEqual(-1, res);
}
```

Рисунок 9 – Метод для тестирования нулевого количества

2.2.9. Тестирование нулевой ширины

Данный тест проверяет метод на обработку ввода нулевой ширины. Код данного метода изображен на рисунке 10.

```
[TestMethod]

public void GetQuantityForProduct_WidthIsNull()
{
    //Arrange
    Calculation cal = new Calculation();

    //Act
    int res = cal.GetQuantityForProduct(3, 1, 15, 0, 45);

    //Assert
    Assert.AreEqual(-1, res);
}
```

Рисунок 10 – Метод для тестирования нулевой ширины

2.2.10. Тестирование нулевой длины

Данный тест проверяет метод на обработку ввода нулевой длины. Код данного метода изображен на рисунке 11.

```
[TestMethod]
Ссылок: 0
public void GetQuantityForProduct_LengthIsNull()
{
    //Arrange
    Calculation cal = new Calculation();

    //Act
    int res = cal.GetQuantityForProduct(3, 1, 15, 20, 0);

    //Assert
    Assert.AreEqual(-1, res);
}
```

Рисунок 11 – Метод для тестирования нулевой количества

2.2.11. Успешность методов тестирования

На рисунке 12 показан результат выполнения всех методов тестирования.

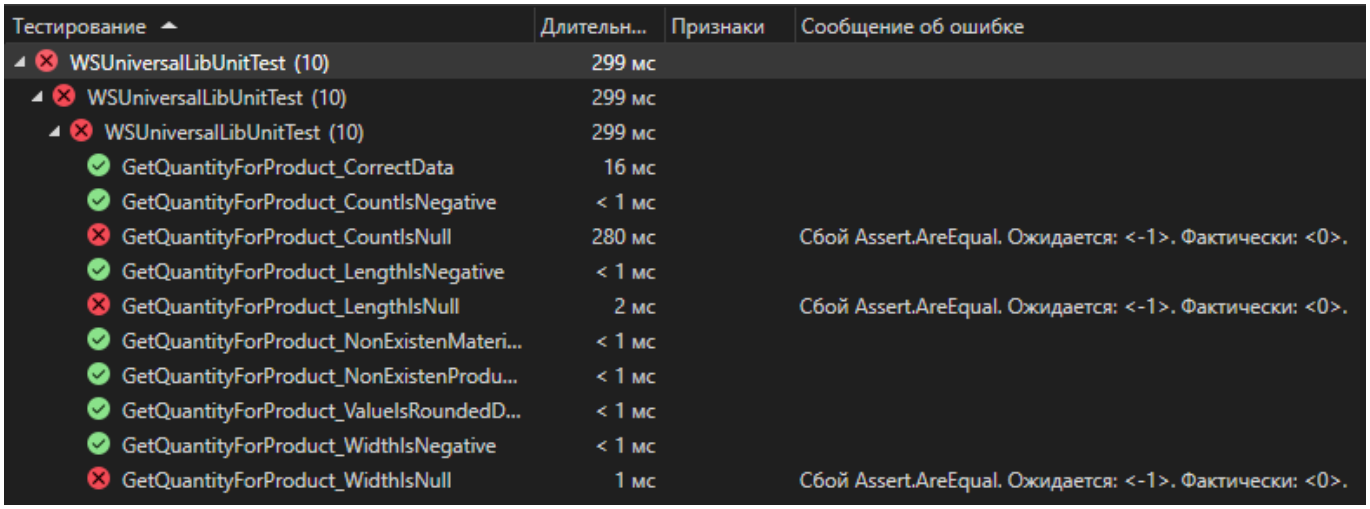

Рисунок 12 – Результат выполнения методов тестирования

2.3. Тестирование «Test-Case»

2.3.1. Тестовый пример 1

На рисунке 13 показан первый тестовый пример.

Тестовый пример #1:

Тестовый пример #	1
Приоритет тестирования	Высокий
Заголовок/название теста	Возможность запуска
Краткое изложение теста	Возможность запуска проекта с правильными данными
Этапы теста	1. Ввести тип продукта: «3» 2. Ввести тип материала: «1» 3. Ввести количество: «15» 4. Ввести ширину: «20» 5. Ввести длину: «45» 6. Запустить метод
Тестовые данные	Тип продукта: «3», Тип материала: «1», Количество: «15», Ширина: «20», Длина: «45»
Ожидаемый результат	Проект запуститься и вернёт целое число.
Фактический результат	Проект запустился и вернул целое число.
Статус	Зачет
Предварительное условие	
Постусловие	
Примечания/комментарии	

Рисунок 13 – Первый тестовый пример

2.3.2. Тестовый пример 2

На рисунке 14 показан второй тестовый пример.

Тестовый пример #2:

Тестовый пример #	2
Приоритет тестирования	Высокий
Заголовок/название теста	Проверка правильности расчетов в проекте
Краткое изложение теста	Правильность расчетов метода с корректными данными
Этапы теста	1. Ввести тип продукта: «3» 2. Ввести тип материала: «1» 3. Ввести количество: «15» 4. Ввести ширину: «20» 5. Ввести длину: «45» 6. Запустить метод
Тестовые данные	Тип продукта: «3», Тип материала: «1», Количество: «15», Ширина: «20», Длина: «45»
Ожидаемый результат	Проект запуститься и вернёт целое число: «114148»
Фактический результат	Проект запустился и вернул целое число: «114148»
Статус	Зачет
Предварительное условие	
Постусловие	
Примечания/комментарии	

Рисунок 14 – Второй тестовый пример

2.3.1. Тестовый пример 3

На рисунке 15 показан третий тестовый пример.

Тестовый пример #3:

Тестовый пример #	3
Приоритет тестирования	Высокий
Заголовок/название теста	Проверка на несуществующий тип продукта
Краткое изложение теста	Проверка на контроль за существованием типа продукта
Этапы теста	1. Ввести тип продукта: «1000» 2. Ввести тип материала: «1» 3. Ввести количество: «15» 4. Ввести ширину: «20» 5. Ввести длину: «45» 6. Запустить метод
Тестовые данные	Тип продукта: «1000», Тип материала: «1», Количество: «15», Ширина: «20», Длина: «45»
Ожидаемый результат	Проект запуститься и вернёт целое число: «-1»
Фактический результат	Проект запустился и вернул целое число: «-1»
Статус	Зачет
Предварительное условие	
Постусловие	
Примечания/комментарии	

Рисунок 15 – Третий тестовый пример

2.3.1. Тестовый пример 4

На рисунке 16 показан четвертый тестовый пример.

Тестовый пример #4:

Тестовый пример #	4
Приоритет тестирования	Высокий
Заголовок/название теста	Проверка на отрицательные значения
Краткое изложение теста	Контроль на отрицательность введённых значений
Этапы теста	1. Ввести тип продукта: «3» 2. Ввести тип материала: «1» 3. Ввести количество: «-1» 4. Ввести ширину: «20» 5. Ввести длину: «45» 6. Запустить метод
Тестовые данные	Тип продукта: «3», Тип материала: «1», Количество: «-1», Ширина: «20», Длина: «45»
Ожидаемый результат	Проект запуститься и вернёт целое число: «-1»
Фактический результат	Проект запустился и вернул целое число: «-1»
Статус	Зачет
Предварительное условие	
Постусловие	
Примечания/комментарии	

Рисунок 16 – Четвертый тестовый пример

2.3.1. Тестовый пример 5

На рисунке 17 показан пятый тестовый пример.

Test case #5:

Test Case #	5
Приоритет тестирования	Высокий
Заголовок/название теста	Проверка на отрицательные значения
Краткое изложение теста	Контроль над нулевыми значениями
Этапы теста	1. Ввести тип продукта: «3» 2. Ввести тип материала: «1» 3. Ввести количество: «-0» 4. Ввести ширину: «20» 5. Ввести длину: «45» 6. Запустить метод
Тестовые данные	Тип продукта: «3», Тип материала: «1», Количество: «0», Ширина: «20», Длина: «45»
Ожидаемый результат	Проект запуститься и вернёт целое число: «-1»
Фактический результат	Проект запустился и вернул целое число: «0»
Статус	Незачет
Предварительное условие	
Постусловие	
Примечания/комментарии	

Рисунок 17 – Пятый тестовый пример