**5 ЛАБОРАТОРНАЯ РАБОТА №5**

**«Исследование способов модульного тестирования программного**

**обеспечения в среде NUnit»**

**5.1 Цель работы**

Исследовать эффективность использования методологии TDD при разработке программного обеспечения. Получить практические навыки использования фреймворка NUnit для модульного тестирования программного обеспечения.

**5.2 Вариант задания**

Реализовать на языке C# один из классов, спроектированных в лабораторной работе № 1. Методы класса при этом не реализовывать. Разработать для созданного класса набор модульных тестов, включающий тесты для каждого метода. Запустить набор тестов, проанализировать и сохранить результаты. Поочередно реализовать методы класса, выполняя тестирование при каждом изменении программного кода. После того, как весь набор тестов будет выполняться успешно, реализацию классов можно считать завершенной.

**5.3 Ход выполнения работы**

5.3.1 В начале выполнения работы для классов, созданных в первой лабораторной работе, были написаны тестовые классы. Код тестовых классов представлен в листинге 5.1.

Листинг 5.1 – Код первого тестового класса

public class MatrixCounterTests

{

private readonly MatrixCounter \_matrix;

public MatrixCounterTests()

{

var builder = new MatrixCounter.MatrixCounterBuilder();

\_matrix = builder.Build();

}

[Fact]

public void MatrixTest1()

{

var source = new List<List<int>>

{

new List<int> { 1 },

};

var result = \_matrix.CountNegativeNumbersInRowsWithZeros(source);

result.ShouldBeEquivalentTo(0);

}

[Fact]

public void MatrixTest2()

{

var source = new List<List<int>>

{

new List<int> { 1 },

};

var result = \_matrix.CountNegativeNumbersInRowsWithZeros(source);

result.ShouldBeEquivalentTo(0);

}

[Fact]

public void MatrixTest3()

{

var source = new List<List<int>>

{

new List<int> { 1, 2, 3},

new List<int> { 4, 5, 6},

new List<int> { 7, 8, 9},

};

var result = \_matrix.CountNegativeNumbersInRowsWithZeros(source);

result.ShouldBeEquivalentTo(0);

}

[Fact]

public void MatrixTest4()

{

var source = new List<List<int>>

{

new List<int> { 1, 5, 0, 0},

new List<int> { 2, 3, 4, 5},

};

var result = \_matrix.CountNegativeNumbersInRowsWithZeros(source);

result.ShouldBeEquivalentTo(0);

}

[Fact]

public void MatrixTest5()

{

var source = new List<List<int>>

{

new List<int> { 1, 2, 3, 4},

new List<int> { 0, 0, 0, -1},

new List<int> { 6, 7, 8, 9},

new List<int> { 1, 2, 3, 4},

};

var result = \_matrix.CountNegativeNumbersInRowsWithZeros(source);

result.ShouldBeEquivalentTo(1);

}

[Fact]

public void MatrixTest6()

{

var source = new List<List<int>>

{

new List<int> { 1, 2, 0},

new List<int> { 4, 0, 6},

new List<int> { 7, 0, 9},

};

var result = \_matrix.CountNegativeNumbersInRowsWithZeros(source);

result.ShouldBeEquivalentTo(0);

}

[Fact]

public void MatrixTest7()

{

var source = new List<List<int>>

{

new List<int> { 1, 0, 0, -1, 0},

new List<int> { 2, 3, 4, 5, 6},

new List<int> { 0, 0, -1, -6, 0},

};

var result = \_matrix.CountNegativeNumbersInRowsWithZeros(source);

result.ShouldBeEquivalentTo(3);

}

}

public class StringExtensionTests

{

private readonly ILogger \_logger;

private static readonly string LogFilePath = "C:\\Users\\k\_dod\\repos\\5Semestr\\tpo\\lr3\\StringExtensionsTestsLogs\\log.log";

public StringExtensionTests()

{

Log.Logger = \_logger = new LoggerConfiguration()

.MinimumLevel.Verbose()

.WriteTo.Console(standardErrorFromLevel: LogEventLevel.Verbose)

.WriteTo.File(LogFilePath)

.CreateLogger();

}

[Theory]

[InlineData("a", "")]

[InlineData(":", "")]

[InlineData("abcde", "")]

[InlineData(":kukaracha", "kukaracha")]

[InlineData(":kukaracha:", "kukaracha")]

[InlineData(":kukaracha:azaza:", "kukaracha")]

public void StringExtensionsTest1(string source, string expectedResult)

{

var result = source.GetStringBetweenColons();

result.ShouldBeEquivalentTo(expectedResult);

var logMessage = $"StringExtensionsTest1 - Source: {source}, Expected result : {expectedResult}, Result: {result}";

\_logger.Information(logMessage);

Log.CloseAndFlush();

}

}

Далее были реализованы тестируемые классы с упомянутымм ранее методами, он показан в листинге 5.2.

Листинг 5.2 – Код классов

public class MatrixCounter

{

private MatrixCounter() {}

public int CountNegativeNumbersInRowsWithZeros(List<List<int>> data)

{

int result = 0;

foreach (var row in data)

{

if (row.Contains(0))

{

var amount = row.Count(x => x < 0);

result += amount;

}

}

return result;

}

public class MatrixCounterBuilder()

{

public MatrixCounter Build()

{

return new MatrixCounter();

}

}

}

public static class StringExtension

{

public static string GetStringBetweenColons(this string str)

{

var match = Regex.Match(str, ":[^:]\*:?");

var result = match.Groups[0].Value.Replace(":", "");

return result;

}

}

На рисунке 5.1 показан результат работы тестов, написанных ранее для класса StringChanger.

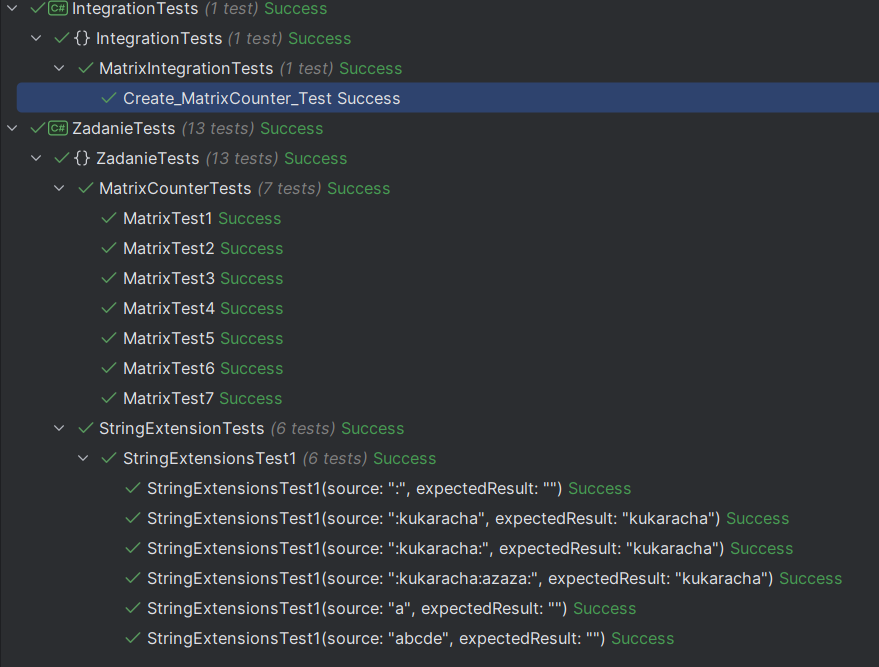


Рисунок 5.1 – Результаты тестов

**Выводы**

В ходе лабораторной работы была исследована эффективность использования методологии TDD при разработке программного обеспечения. Также были получены практические навыки использования фреймворка XUnit для модульного тестирования программного обеспечения. В конце выполнения лабораторной работы был написан отчет.