ЛАБОРАТОРНАЯ РАБОТА №1

Курс «Тестирование и внедрение программного обеспечения»

Тема: Юнит-тестирование с помощью MS Testing Framework и NUnit.

Цель: Научиться писать и запускать юнит-тесты, применять на практике базовые принципы юнит-тестирования.

Темы для предварительной проработки [УСТНО]**:**

- Виды тестирования ПО.
- Юнит-тестирование.
- Test Driven Development (TDD).
- Работа с MSTest и NUnit.
- Принцип Arrange-Act-Assert.

Индивидуальные задания [КОД] **:**

1. Написать тесты с помощью NUnit для решения задачи, в соответствии с вариантом (*приложение А*), с Assert'ами простых условий и исключений.
2. Написать тесты с помощью MSTest для решения задачи, в соответствии с вариантом, со строковыми Assert'ами.
3. Написать тесты с помощью NUnit для решения задачи, в соответствии с вариантом, с Assert'ами коллекций.
4. Написать юнит-тесты для своего курсового проекта.

Контрольные вопросы [ОТЧЕТ]**:**

1. Перечислите основные виды программных ошибок.
2. Что такое юнит-тестирование? Регрессионное тестирование?
3. Что означает функциональное тестирование, тестирование черного и белого ящика?
4. В чем заключаются особенности Test-Driven Development?
5. Что означает аббревиатура AAA? Где она применяется?
6. Кратко опишите возможности фреймворка MSTest.
7. NUnit. Установка библиотеки, использование NUnit.exe.
8. Опишите разницу между классической моделью и моделью ограничений NUnit (Classic model vs. Constraint model).
9. Кратко опишите назначение основных атрибутов NUnit.

Рекомендуемые источники:

[1] Бек К. Экстремальное программирование. Разработка через тестирование. – СПб.: Питер, 2003. – 512 с.

[2] Месарош Дж. Шаблоны тестирования xUnit: рефакторинг кода тестов. – М.: Вильямс, 2008. – 629 с.

[3] Osherove R. The Art of Unit Testing: with examples in C#. – Manning Publications, 2013. – 296 p.

Приложение А. Варианты индивидуальных заданий.

Вариант 1:	*Вариант 4:*	*Вариант 7:*	*Вариант 10:*
1-1, 1-2, 3-3	*2-1, 1-2, 2-3*	*3-1, 1-2, 3-3*	*1-1, 3-2, 3-3*
Вариант 2:	*Вариант 5:*	*Вариант 8:*	*Вариант 11:*
1-1, 2-2, 2-3	*3-1, 1-2, 2-3*	*1-1, 3-2, 1-3*	*2-1, 3-2, 2-3*
Вариант 3:	*Вариант 6:*	*Вариант 9:*	*Вариант 12:*
2-1, 2-2, 1-3	*2-1, 3-2, 1-3*	*3-1, 2-2, 2-3*	*3-1, 3-2, 1-3*

Набор 1 [Задания 1-1, 1-2, 1-3]

1. В Production Code написать класс Triangle с методом double Area(), который вычисляет площадь треугольника. Стороны могут быть заданы как в методе SetSides(double a, double b, double c), так и в параметризированном конструкторе; при этом если одна из сторон задана отрицательной, бросается исключение FormatException; если стороны не могут образовать треугольник, то бросается исключение ArgumentException. В Test Code написать тесты, которые проверяют соответствие результатов метода спецификации требований.
2. В Production Code написать класс StringFormatter с методом string SafeString(string s), который во входной строке экранирует все одинарные кавычки (дублирует их для защиты от sql-инъекций в Postgre), а также переводит все ключевые слова select, insert, update, delete в верхний регистр; если строка пустая, то метод возвращает также пустую строку; если строка является null, то метод бросает исключение NullReferenceException. В Test Code написать тесты, которые проверяют корректность работы метода.
3. В Production Code написать класс ArrayProcessor с методом int[] SortAndFilter(int[] a), который формирует новый массив на основе входного: сортирует массив (не затрагивая исходный) и удаляет из него все элементы, кроме положительных четырехзначных чисел. В Test Code написать тесты, проверяющие соответствие работы метода спецификации требований.

Набор 2 [Задания 2-1, 2-2, 2-3]

1. В Production Code написать класс Rectangle с методом double Diagonal(), который вычисляет длину диагонали прямоугольника. Стороны вычисляются при установке координат вершин как в параметризированном конструкторе, так и в методе SetVertices(double[] x, double [] y), где x и y – это массивы из 4 координат вершин (x,y). При этом если данные вершины не образуют прямоугольник, бросается исключение ArgumentException. В Test Code написать тесты, которые проверяют соответствие результатов метода спецификации требований.
2. В Production Code написать класс StringFormatter с методом string WebString(string s), который во входной строке добавляет в начале «http://», если сетевой протокол изначально в ней не был указан (если строка оканчивается на «.git», то предварить строку «git://»). Если строка пустая, то метод возвращает также пустую строку; если строка является null, то метод бросает исключение NullReferenceException. В Test Code написать тесты, которые проверяют корректность работы метода.
3. В Production Code написать класс ArrayProcessor с методом int[] SortAndFilter(int[] a), который формирует новый массив на основе входного: сортирует массив (не затрагивая исходный) и удаляет из него все повторяющиеся отрицательные элементы (оставляя по одному). В Test Code написать тесты, которые проверяют соответствие результатов метода спецификации требований.

Набор 3 [Задания 3-1, 3-2, 3-3]

1. В Production Code написать класс LinearEquationsSystem с методом double[] Solve(), который решает систему линейных уравнений 3x3 или 2x2. Коэффициенты матрицы системы задаются как в методе SetCoefficients(double[,] coeffs), так и в параметризированном конструкторе; при этом если размерность массива coeffs не равна 3x3 или 2x2, бросается исключение FormatException; если определитель матрицы равен 0, то бросается исключение ArgumentException. В Test Code написать тесты, которые проверяют соответствие результатов метода спецификации требований.
2. В Production Code написать класс StringFormatter с методом string ShortFileString(string path), который во входной строке с полным путем к файлу оставляет только короткое имя файла, меняет расширение на TMP (в том числе, если исходный файл не имеет расширения) и переводит все символы нижнего регистра в символы верхнего. Если строка пустая, то метод возвращает также пустую строку; если строка является null, то метод бросает исключение NullReferenceException. В Test Code написать тесты, которые проверяют корректность работы метода.
3. В Production написать класс ArrayProcessor с методом double[] SortAndFilter(double[] a), который формирует новый массив на основе входного (не затрагивая его): заменяет в массиве все отрицательные числа на положительные, сортирует массив в порядке убывания и удаляет из него все повторяющиеся элементы (оставляя по одному). В Test Code написать тесты, которые проверяют соответствие результатов метода спецификации требований.