ЛАБОРАТОРНАЯ РАБОТА №2

Курс «Тестирование и внедрение программного обеспечения»

Тема: Заглушки в юнит-тестировании (стабы и моки). Изоляционные фреймворки.

Цель: Научиться писать заглушки для юнит-тестов вручную, а также применять на практике изоляционные фреймворки.

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

- Внешние зависимости, тестирование взаимодействия.
- Виды заглушек: фейки, стабы, моки.
- Изоляционные фреймворки (Moq, NSubstitute, FakeItEasy, Microsoft Fakes).

Индивидуальные задания [КОД] **:**

1. Написать тесты и собственную заглушку для решения задачи, в соответствии с вариантом (*приложение А*). Внедрение зависимости произвести в конструкторе. Протестировать как состояние (написать стаб), так и поведение (написать мок).
2. Написать тесты и собственную заглушку для решения задачи, в соответствии с вариантом. Внедрение зависимости произвести в сеттере. Протестировать как состояние (написать стаб), так и поведение (написать мок).
3. Написать тесты с помощью Moq или любого другого изоляционного фреймворка для задач 1-2. Протестировать как состояние (написать стаб), так и поведение (написать мок).

Контрольные вопросы [ОТЧЕТ]**:**

1. Приведите определения понятий «Заглушка (Stub)», «Внешняя зависимость (External Dependency)», «Шов (Seam)».
2. Опишите общую схему написания собственной заглушки для юнит-тестов.
3. Какие есть ООП-механизмы для внедрения зависимости (Dependency Injection, DI)?
4. В чем заключается разница между стабами и моками?
5. Для чего и как используются изоляционные фреймворки (на примере Moq или NSubstitute)?

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

Набор 1 [Задания 1-1, 1-2, 1-3, 1-4]

1. В Production Code написать класс FileService с методом int MergeTemporaryFiles(string dir). В данном методе класс обращается к директории dir и компонует из всех файлов с расширением .tmp в данной директории один файл backup.tmp (простой конкатенацией), после чего удаляет все учтенные файлы. Метод возвращает количество учтенных файлов; если передан путь несуществующего каталога, должно быть брошено исключение. Данный метод должен вызываться, в свою очередь, из метода PrepareData() класса ReportViewer. Этот метод должен сразу прекратить выполнение, если количество учтенных файлов было равно нулю; иначе – присвоить свойству BlockCount класса ReportViewer это количество. Протестировать все ситуации.
2. В Production Code написать класс SqlQueryPreparator с методом string[] PrepareQueries(string[] queries). Данный метод преобразует строки из массива queries в «безопасный формат», для чего производится обращение к методу string SafeString(string s) класса StringFormatter (см. все спецификации касательно этого метода в л/р №1).
3. В Production Code написать класс Signal со свойством double[] Samples и методом FullRectify(), который вызывает метод SortAndFilter() класса ArrayProcessor (см. задание 3-3 из л/р №1) и записывает результат в массив Samples. Также метод записывает сумму, разность и среднее арифметическое в три строки в файл results.log. Если такой файл уже существует, то создать файл results(1).log; если файл results(1).log существует, то создать results(2).log и т.д. Самостоятельно продумать структуру теста и реализовать его.
4. В Production Code написать класс LinksRepository, в котором хранится коллекция ссылок, с методом Add(string link). Данный метод перед тем, как добавить строку в коллекцию, вызывает метод string WebString(string s) класса StringFormatter (см. все спецификации касательно этого метода в л/р №1).

Набор 2 [Задания 2-1, 2-2, 2-3]

1. В Production Code написать класс FileService с методом int RemoveTemporaryFiles(string dir). В данном методе класс находит в директории dir файл ToRemove.txt (в нем хранятся построчно имена файлов, которые необходимо удалить), считывает строки из него, последовательно удаляет файлы с соответствующими именами и возвращает суммарный размер удаленных файлов в байтах. Если передан путь несуществующего каталога или файл ToRemove.txt отсутствует в директории, бросается исключение. Если какой-то файл из списка отсутствует, то запись об этом добавляется в файл error.log: дописывается строка «Файл *<имя_не_найденного_файла>* не был найден». Данный метод должен вызываться, в свою очередь, из метода Clean() класса ReportViewer. Этот метод должен сразу прекратить выполнение, если было брошено исключение; иначе – присвоить свойству UsedSize класса ReportViewer значение, которое вернул метод RemoveTemporaryFiles. Протестировать все ситуации.
2. В Production Code написать класс LinearEquationsSystem с методом double[] Solve(), который решает систему 2 линейных уравнений по методу Крамера. Коэффициенты матрицы системы задаются в методе SetCoefficients(double[,] coeffs). Для решения системы уравнений в методе Solve() вызывается метод Determinant() класса Matrix. Если этот метод возвращает 0, то в методе Solve() бросается исключение.
3. В Production Code написать класс TemporaryFileRepository, в котором хранится коллекция имен файлов, с методом Add(string filename). Данный метод перед тем, как добавить имя файла в коллекцию, вызывает метод string ShortFileString(string path) класса StringFormatter (см. все спецификации касательно этого метода в л/р №1).