МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«Харківський політехнічний інститут»

ЗВІТ

до лабораторної роботи №3

по курсу: «Тестування і налаштування програмного забезпечення»

на тему: «Модульне тестування»

Виконав: ст.гр. КН-26

Савчук Я.І.

Перевірила:

ас. Луценко С.Ю.

Харків 2018

**Лабораторна робота № 3**

**Тема**: Модульное тестирование в Java, C#

**Мета:** знакомство с методами и техниками модульного тестирования на реальных примерах программного кода, выполненного на разных языках программирования таких как Java, C#.

**Задание:**

Используя созданные в лабораторной работе программы на языках Java и C#, подготовить с использованием библиотеки JUnit тестовые классы и серию тестов каждого класса для Java–программы и C#-программы. Для формирования серии тестов использовать данные лабораторной работы № 2, то есть входные и выходные массивы данных для тестирования.

Шаг 1. Создать проект с тестовыми классами.

Шаг 2. Убедиться в прохождении всех тестовых данных полученных в лабораторной работе № 2.

Шаг 3. Расширить классы тестов так, чтобы в них применялся: а) метод assertTrue и/или assertFalse; б) класс Assert.

Шаг 4. Изменить Java–программу и C#-программы так, чтобы модульное тестирование было реализовано средствами обработки исключений.

**Выполнение работы:**

Модульное тестирование или как его еще называют UNIT-тестирование характеризуется следующими особенностями:

Модульное тестирование - это тестирование программы на уровне отдельно взятых модулей, функций или классов.

Цель модульного тестирования состоит в выявлении локализованных в модуле ошибок в реализации алгоритмов, а также в определении степени готовности системы к переходу на следующий уровень разработки и тестирования.

Модульное тестирование проводится по принципу "белого ящика“.

Юнит-тест – это просто фрагмент кода, который проверяет другой фрагмент кода и выдает ответ на вопрос: ведет ли себя проверяемый фрагмент как задумано или нет.

В разработке через тестирование используется цикл из следующих шагов:

1. Написание теста. Для написания теста разработчик должен полностью разобраться в спецификации и требованиях.
2. Прогон всех тестов. Нужно убедиться, что новый тест не работает. Это подтвердит, что тест правильный.
3. Написание кода. Создается только минимальный код, достаточный для выполнения теста.
4. Прогон автоматизированных тестов. Все тесты должны работать.
5. Рефакторинг.
6. Повторение.

Рассмотрим реализацию методов класса Assert, а именно Equals – проверка на равенство. Assert.AreEqual(expected, actual [, string message])expected – желаемое значение, обычно фиксировано; actual – действительное значение, вычисляется тестируемым фрагментом кода; message – сообщение, выводимое в случае несовпадения желаемого и действительного значений.

Листинг функций тестирования:

[Test]

public void TestMethod1()

{

Form1 sut = new Form1();

double result = sut.Find(2,1,2,3,2);

Assert.AreEqual(8, result);

// TODO: Add your test code here

Assert.Pass("Your first passing test");

}

[Test]

public void TestMethodException1()

{

Form1 sut = new Form1();

double result = sut.Find(2, 1, 2, 3, 2);

if (result == 9)

Assert.Pass("Your second passing test");

else

{

throw new Exception("Exception");

}

}

}

Результат выполнения теста представлен на рисунке1.

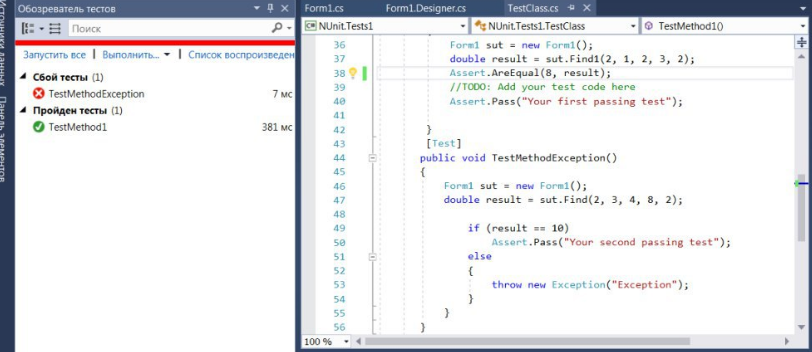


Рисунок 1 – Результат тестирования

**Выводы:** Ознакомились с методами и техниками модульного тестирования и выполнила на реальном примере программ, написанных на Java и C#. По-моему мнению более подходящим вариантом для реализации модульного тестирования является применение библиотеки JUnit, чем использование механизма обработки исключений try-catch.