# Введение. Основы TDD и Unit Тестирования.

| **№ урока:** | 1 | **Курс:** | Test Driven Development |
|---|---|---|---|

| **Средства обучения:** | Компьютер с установленной Visual Studio<br>NUnit Framework<br>NUnit Adapter (optional) |
|---|---|

## Обзор, цель и назначение урока

- Понимать работу юнит-тестов
- Научится создавать юнит-тесты с использованием атрибутов NUnit.
- Понимать технику TDD.

## Изучив материал данного занятия, учащийся сможет:

- Разрабатывать приложения с использованием TDD.
- Интегрировать и использовать NUnit в Visual Studio 2012.
- Создавать Unit тесты.

## Содержание урока

1. Что такое Unit тесты?
2. Свойства хорошего Unit теста.
3. Интеграционное тестирование.
4. Разработка через тестирование – определение.
5. Фреймворки для Unit тестирования.
6. Основные атрибуты NUnit.

## Резюме

- Unit тест – блок кода (обычно метод), который вызывает тестируемый блок кода и проверяет его правильность работы. Если результат юнит-теста не совпадает с ожидаемым результатом, тест считается не пройденным.
- Юнит-тест представляет собой автоматизированный кусок кода, который вызывает тестируемый метод или класс, а затем проверяет некоторые предположения о логическом поведении этого метода или класса. Это полностью автоматизировано, надежно, легко читаемо и сопровождаемо.
- **Unit тест должен быть:**
  1. Автоматизированным и повторяемым.
  2. Простым в реализации.
  3. После написания он должен остаться для последующего использования.
  4. Кто угодно в команде должен иметь возможность запустить Unit тест.
  5. Должен запускаться одним нажатием кнопки.
  6. Должен выполняться быстро.
- Интеграционное тестирование – тестирование нескольких связанных модулей программного обеспечения как единого целого.
- При интеграционном тестировании существует много критических точек, в которых приложение может дать сбой, что делает поиск ошибок сложнее.



CyberBionic systematics

CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com

- Test-Driven Development (TDD) – разработка через тестирование. Подход разработки ПО, который заключается в написании юнит-теста перед написанием самого кода.
- **Техника TDD достаточно проста:**
  1. Написать тест, который доказывает неработоспособность конечного продукта. Мы должны писать тест, как будто у нас уже есть рабочий код, так что ошибка теста означает недоработку производственного кода.
  2. Пройти тест, с помощью написания нового кода, который будет отвечать всем ожиданиям нашего теста. Он должен быть написан так просто, как это возможно.
  3. Рефакторинг кода. Если тест пройден, вы можете переходить к следующему юнит-тесту или почистись код, чтобы сделать его более удобным для чтения, убрать дублирование кода и так далее. Рефакторинг может быть сделан после написания нескольких тестов или после написания каждого теста. Это важная практика, поскольку она обеспечивает чистоту вашего кода, читабельность и сопровождение, в то же время прохождение всех ранее написанных тестов.
- Правильно реализованное TDD может повысить качество вашего кода, уменьшить количество ошибок, повысить уверенность в коде, сократить время для поиска ошибок, улучшить дизайн кода, и сделать вашего менеджера счастливей.
- Юнит-тест фреймворки состоят из библиотек и модулей, которые помогают разработчикам тестировать свой код с помощью юнит-тестов.
- Есть более чем 150 юнит-тест фреймворков, практически по одному на каждый язык программирования. .Net имеет 9 различных юнит-тест фреймворков. Среди них, NUnit является стандартом.
- В совокупности, эти юнит-тест фреймворки называют xUnit фреймворками, потому что их имена обычно начинаются с первых букв языка, для которых они были построены. Возможно, вы будете встречать CppUnit для C++, JUnit для Java, NUnit для .Net и HUnit для языка программирования Haskell. Не все из них следуют этим правилам именования, но большинство их используют.
- Мы будем использовать NUnit для юнит-тестирования, котрый позволяет легко писать тесты, выполнять их, получать результаты.
- **Атрибуты NUnit:**
- [SetUp] – этот атрибут мы можем применять к методу так же, как и атрибут [Test], он вызывает метод каждый раз перед запуском тестов в вашем классе.
- [TearDown] – этот атрибут будет вызывать метод один раз после выполнения каждого теста.
- [TextFixtureSetUp], [TextFixtureTearDown] – позволяют установить состояние один раз перед запуском всех тестов или после запуска тестов.
- [ExpectedException] – сообщение об исключении предоставляется в качестве параметра атрибута.
- Нет смысла получать от метода логическое значение, потому что вызов метода должен вызвать исключение.
- [Ignore] – применяется к методам, которые не будут выполняться при запуске всех юнит-тестов.
- Вы можете настроить тесты для работы под конкретные категории испытаний, таких как медленных и быстрых тестов. Сделать это вы можете с помощью атрибута [Category].
- Вы можете создать несколько категорий тестов в коде, а затем выбрать конкретную категорию для запуска в среде NUnit.

## Закрепление материала

1. Что такое Unit тест?



CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com

2. Какие свойства хорошего юнит-теста?
3. Что такое TDD?
4. Расскажите технику использования TDD.
5. Назовите основные атрибуты NUnit.



CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com