

ЛЕКЦИЯ №1 ПО ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

**ТЕМА: TEST DRIVEN DEVELOPMENT (РАЗРАБОТКА
ЧЕРЕЗ ТЕСТИРОВАНИЕ)**

Москва, 2020

TDD

САМОДОКУМЕНТИРУЕМЫЙ КОД

**ЭКОНОМИЧЕСКИЙ ЭФФЕКТ ОТ РАЗРАБОТКИ
ПРОГРАММЫ**

ИМЕНОВАНИЕ ПЕРЕМЕННЫХ:

`countOfWord`

**СОВЕТЫ ПО НАПИСАНИЮ ХОРОШО
ОБСЛУЖИВАЕМЫХ ПРОГРАММ:**

- объявление должно быть рядом с кодом, который их использует.
- комментарии должны быть там где необходимо
- проверки входных данных и результатов

ПРИНЦИПЫ РАЗРАБОТКИ СЛОЖНЫХ СИСТЕМ

1. Абстракция и уточнение
 2. Выделение интерфейсов и сокрытие данных
 3. Разделение ответственностей
 4. Разделение интерфейса и реализации
- MVC(Model View Controller) Модель, Представление,
Контроллер
ASP NET MVC

ЖИЗНЕННЫЙ ЦИКЛ ПО

- 1. РАЗРАБОТКА ТЗ**
- 2. РАЗРАБОТКА АРХИТЕКТУРЫ**
- 3. НАПИСАНИЕ КОДА**
- 4. ТЕСТИРОВАНИЕ**
- 5. РАЗРАБОТКА БЕТА ВЕРСИИ ПРОГРАММЫ**
- 6. РАЗРАБОТКА ОКОНЧАТЕЛЬНОЙ ВЕРСИИ**
- 7. СОПРОВОЖДЕНИЕ (SUPPORT)**

Типы тестирования

- Модульное
- Интеграционное
- Системное
- Тестирование производительности
- Юзабилити тестирование

<https://vk.com/club191539433>

Что такое UnitTest

- Это метод, который вызывает тестируемый метод и проверяет его правильность работы. То есть метод возвращает результат, мы его сравниваем с ожидаемым и если они не совпадают, то тест считается проваленным, иначе успешным

Плюсы UnitTest

- Модульное тестирование позволяет быть уверенным, что из-за исправления одного бага у вас не появился другой
- Unit-тесты служат некоторым образом документацией к коду и путеводителем по нему
- Применение unit-тестов даёт более качественное отделение интерфейса от реализации

Минусы UnitTest

- Когда каждая функция тестируется по отдельности, нет гарантии что соединив их вместе программа будет работать правильно
- При недостаточно продуманной архитектуре проекта тесты могут мешать изменению кода, а не способствовать ему
- Тесты, успешно выполненные и перевыполненные после внесения всяческих изменений, не дают никакой гарантии, что в коде нет ошибок

Именованние проектов и классов

<PROJECT_NAME>.Core	➡	<PROJECT_NAME>.Core.Tests
<PROJECT_NAME>.BI	➡	<PROJECT_NAME>.BI.Tests
StringUtil	➡	StringUtilTests
ArrayManager	➡	ArrayManagerTests

Именованние методов

На мой взгляд, лучший способ именованния методов такой:
[Тестируемый метод]_[Сценарий]_[Ожидаемое поведение].

```
class CalculatorTests
{
    public void Sum_2Plus5_7Returned()
    { ... }
}

class ArrayManagerTests
{
    public void Max_NullArray_ExceptionReturned()
    { ... }
}
```

Frameworks

- MSTest
- NUnit
- xUnit.net



Шаблон AAA

Arrange – размещение

Act - действия

Assert - утверждение

```
int expected = 9;  
int[] a = {3,9,-  
22,5};
```

```
int actual =  
ArrayManager.Ma  
x(a);
```

```
Assert.AreEqual(e  
xpected, actual)
```

Классы для тестирования

Assert

- Сравнение двух значений
- Проверка на истину/ложь

CollectionAssert

- Сравнение двух коллекций
- Проверка элемента в коллекции

StringAssert

- Сравнение строк
- Проверка подстроки в строке

Атрибуты

TestClass – Тестирующий класс

TestMethod – Тестирующий метод

TestInitialize – Метод для инициализации. Вызывается перед каждым тестирующим методом.

TestCleanup – Метод для освобождения ресурсов. Вызывается после каждого тестирующего метода

ClassInitiazlie – Вызывается один раз для тестирующего класса, перед запуском тестирующего метода.

ClassCleanup – Вызывается один раз для тестирующего класса, после завершения работы тестирующих методов

AssemblyInitialize – вызывается перед тем как начнут работать тестирующие методы в сборке

AssemblyCleanup – вызывается после завершения работы тестирующих методов в сборке.

TDD

Вспомним, полный цикл TDD состоит из следующих этапов:

1. Добавить небольшой тест.
2. Запустить все тесты, при этом обнаружить, что что-то не срабатывает.
3. Внести небольшое изменение.
4. Снова запустить тесты и убедиться, что все они успешно выполняются.
5. Устранить дублирование с помощью рефакторинга.

Зависимость является ключевой проблемой разработки программного обеспечения. Если фрагменты SQL, зависящие от производителя используемой вами базы данных, разбросаны по всему коду и вы хотите поменять производителя, то непременно окажется, что код зависит от этого производителя. Вы не сможете поменять производителя базы данных и при этом не изменить код.

Зависимость является проблемой, а дублирование — ее симптомом. Чаще всего дублирование проявляется в виде дублирования логики — одно и то же выражение появляется в различных частях кода. Объекты — отличный способ абстрагирования, позволяющий избежать данного вида дублирования.

В отличие от большинства проблем в реальной жизни, где устранение симптомов приводит только к тому, что проблема проявляется в более страшной форме где-то еще, устранение дублирования в программах устраняет и зависимость. Именно поэтому существует второе правило TDD. Устраняя дублирование перед тем, как заняться следующим тестом, мы максимизируем наши шансы сделать его успешным, внося всего одно изменение.