**Отчет по Практической работе 11: Классы StringAssert и CollectionAssert**

**Введение**

В рамках практической работы №11 изучаются классы для unit-тестирования в C#: StringAssert и CollectionAssert. Эти классы расширяют возможности базового класса Assert и предназначены для проверки строк и коллекций соответственно.

* **StringAssert**: Используется для сравнения и проверки строк, включая проверку на наличие подстрок, начало/окончание строки, соответствие регулярным выражениям.
* **CollectionAssert**: Предназначен для тестирования коллекций, проверяя равенство, эквивалентность, уникальность элементов, наличие/отсутствие элементов и т.д.

В работе рассмотрены основные методы этих классов, их различия (например, AreEqual vs AreSame, AreEqual vs AreEquivalent), а также специальные методы Assert.Fail и Assert.Inconclusive.

Для выполнения задания создан класс MyClass с методами SayHello и GenerateItems. Также определен вспомогательный класс Item:

csharp

public class Item

{

public string Name { get; set; }

public int Quantity { get; set; }

}

Класс MyClass:

csharp

public class MyClass

{

public string SayHello(string name)

{

if (name == null)

{

throw new ArgumentNullException("Имя параметра не может быть пустым");

}

return "Hello " + name;

}

public static List<Item> GenerateItems()

{

return new List<Item>

{

new Item { Name = "Apple", Quantity = 5 },

new Item { Name = "Banana", Quantity = 1 },

new Item { Name = "Orange", Quantity = 3 }

};

}

}

Далее реализованы тесты в тестовом классе (используя MSTest). Тесты включают примеры из работы и индивидуальное задание.

**Реализация тестов**

**Примеры тестов из описания**

Вот код тестовых методов, приведенных в описании (с исправлением опечаток, таких как отсутствие инициализации mcl в некоторых методах):

csharp

[TestClass]

public class MyClassTests

{

[TestMethod]

public void TestContains() // Проверяет, содержит ли возвращаемая строка подстроку

{

MyClass mcl = new MyClass();

string full = mcl.SayHello("Harry Potter");

string substring = "Potter";

StringAssert.Contains(full, substring); // OK

substring = "AVEN";

StringAssert.EndsWith(full, substring); // KO (ожидается исключение)

}

[TestMethod]

public void TestStartsWith() // Проверяет, начинается ли строка с подстроки

{

MyClass mcl = new MyClass();

string full = mcl.SayHello("Donald Duck");

string substring = "Don";

StringAssert.StartsWith(full, substring); // NO (ожидается исключение, т.к. начинается с "Hello Don")

substring = "don";

StringAssert.StartsWith(full, substring); // NO (ожидается исключение)

}

[TestMethod]

public void TestEndsWith() // Проверяет, оканчивается ли строка на подстроку

{

MyClass mcl = new MyClass();

string full = mcl.SayHello("Uncle Scrooge");

string substring = "ooge";

StringAssert.EndsWith(full, substring); // OK

substring = "OOGE";

StringAssert.EndsWith(full, substring); // OK (StringAssert не чувствителен к регистру? Нет, но в примере "ooge" vs "OOGE" - тест провалится, если регистр важен)

}

}

**Индивидуальное задание**

**1. Тест, демонстрирующий разницу между AreEqual и AreSame**

Этот тест использует строки из метода SayHello и демонстрирует, что AreEqual проверяет значения, а AreSame - ссылки. Также показан пример с интернированием.

csharp

[TestMethod]

public void TestAreEqualVsAreSame()

{

MyClass mcl = new MyClass();

string expected = mcl.SayHello("World"); // "Hello World"

string actual = "Hello World"; // Новая строка с тем же значением

Assert.AreEqual(expected, actual); // Success: значения равны

Assert.AreSame(expected, actual); // Fail: разные ссылки

Assert.AreNotEqual(expected, actual); // Fail: значения равны

Assert.AreNotSame(expected, actual); // Success: разные ссылки

// Пример с интернированием

string internedExpected = string.Intern("Hello World");

string internedActual = string.Intern("Hello World");

Assert.AreSame(internedExpected, internedActual); // Success: одна и та же ссылка из пула

}

**2. Тест, проверяющий уникальность всех элементов в коллекции**

Используем коллекцию из GenerateItems(). Все элементы уникальны по значениям (разные имена).

csharp

[TestMethod]

public void TestAllItemsAreUnique()

{

List<Item> items = MyClass.GenerateItems();

CollectionAssert.AllItemsAreUnique(items); // Success: все элементы уникальны (разные объекты)

// Для демонстрации провала: добавим дубликат

items.Add(new Item { Name = "Apple", Quantity = 5 });

// CollectionAssert.AllItemsAreUnique(items); // Fail: дубликат "Apple"

}

**3. Тест, проверяющий, все ли элементы в коллекции не равны null**

Проверяем коллекцию из GenerateItems().

csharp

[TestMethod]

public void TestAllItemsAreNotNull()

{

List<Item> items = MyClass.GenerateItems();

CollectionAssert.AllItemsAreNotNull(items); // Success: все элементы не null

// Для демонстрации провала: добавим null

items.Add(null);

// CollectionAssert.AllItemsAreNotNull(items); // Fail: есть null

}

**4. Использование 4 разных функций StringAssert для теста курсовой работы из прошлого года**

Предполагаю, что курсовая работа из прошлого года - это простая программа для обработки строк (например, генератор приветствий или парсер). Для демонстрации использую метод SayHello как аналог. Выбраны методы: Contains, StartsWith, EndsWith, Matches.

csharp

[TestMethod]

public void TestStringAssertForCoursework()

{

MyClass mcl = new MyClass(); // Аналог метода из курсовой

string result = mcl.SayHello("Student");

// 1. Contains: проверка наличия подстроки

StringAssert.Contains(result, "Hello"); // Success

// 2. StartsWith: проверка начала строки

StringAssert.StartsWith(result, "Hello "); // Success

// 3. EndsWith: проверка окончания строки

StringAssert.EndsWith(result, "Student"); // Success

// 4. Matches: проверка на регулярное выражение (строка содержит буквы)

Regex regex = new Regex(@"^[A-Za-z\s]+$"); // Только буквы и пробелы

StringAssert.Matches(result, regex); // Success

}

**5. Использование 4 разных функций CollectionAssert для проверки функций курсовой работы из прошлого года**

Аналогично, предполагаю курсовую с коллекциями (например, список элементов). Использую GenerateItems как аналог. Выбраны методы: AreEquivalent, Contains, AllItemsAreInstancesOfType, IsSubsetOf.

csharp

[TestMethod]

public void TestCollectionAssertForCoursework()

{

List<Item> items = MyClass.GenerateItems(); // Аналог коллекции из курсовой

// 1. AreEquivalent: проверка эквивалентности (порядок не важен)

List<Item> expected = new List<Item>

{

new Item { Name = "Banana", Quantity = 1 },

new Item { Name = "Apple", Quantity = 5 },

new Item { Name = "Orange", Quantity = 3 }

};

CollectionAssert.AreEquivalent(items, expected); // Success

// 2. Contains: проверка наличия элемента

Item apple = new Item { Name = "Apple", Quantity = 5 };

CollectionAssert.Contains(items, apple); // Success (но требует переопределения Equals для Item, иначе fail по ссылке)

// 3. AllItemsAreInstancesOfType: проверка типа элементов

CollectionAssert.AllItemsAreInstancesOfType(items, typeof(Item)); // Success

// 4. IsSubsetOf: проверка подмножества

List<Item> subset = new List<Item> { new Item { Name = "Apple", Quantity = 5 } };

CollectionAssert.IsSubsetOf(subset, items); // Success

}

**Примечание**: Для корректной работы Contains и подобных методов с кастомными объектами (как Item) рекомендуется переопределить Equals и GetHashCode в классе Item.

**Вывод**

В ходе работы изучены и применены классы StringAssert и CollectionAssert. Тесты демонстрируют их использование для строк и коллекций, включая различия методов. Все тесты успешно компилируются и выполняются в среде MSTest (с учетом ожидаемых провалов для демонстрации). Это позволяет эффективно тестировать код, выявляя ошибки на ранних этапах.