Методы MS TEST

Утверждения равенства

Assert.AreEqual();

```
//Arrange
Задаем тестовые данные
тип expected = …;
//Act
ИмяКласса obj = new ИмяКласса();
тип  actual = obj.ИмяМетода(тестовые данные);
//Assert
Assert.AreEqual(expected, actual);
```

Метод Assert.AreEqual() проверяет равенство ожидаемого и фактического результата, и выдает исключение, если они не равны. Различные числовые типы рассматриваются как неравные

Assert.AreNotEqual();

```
//Arrange
Задаем тестовые данные
тип expected = …;
//Act
ИмяКласса obj = new ИмяКласса();
тип  actual = obj.ИмяМетода(тестовые данные);
//Assert
Assert.AreNotEqual(expected, actual);
```

Метод Assert.AreNotEqual() проверяет неравенство ожидаемого и фактического результата , и выдает исключение, если они равны.

Проверка условий

Assert.IsTrue();

```
//Arrange
Задаем тестовые данные
//Act
ИмяКласса obj = new ИмяКласса();
тип  actual = obj.ИмяМетода(тестовые данные);
//Assert
Assert.IsTrue(actual);
```

Метод Assert.IsTrue() проверяет, что тестовые данные удовлетворяют проверяемым условиям

Assert.IsFalse();

```
//Arrange
Задаем тестовые данные
//Act
ИмяКласса obj = new ИмяКласса();
тип  actual = obj.ИмяМетода(тестовые данные);
//Assert
Assert.IsFalse(actual);
```

Метод Assert.IsFalse() проверяет, что тестовые данные **НЕ** подходят под проверяемые условия

Методы MS TEST

Assert.IsNull();

```
//Arrange
Задаем тестовые данные
тип expected = …;
//Act
ИмяКласса obj = new ИмяКласса();
тип  actual = obj.ИмяМетода(тестовые данные);
//Assert
Assert.IsNull(actual);
```

Метод Assert.IsNull() означает, что «переданный параметр должен быть null»: если он не равен нулю, то тестовый пример не пройден.

Assert.IsNotNull();

```
//Arrange
Задаем тестовые данные
тип expected = …;
//Act
ИмяКласса obj = new ИмяКласса();
тип  actual = obj.ИмяМетода(тестовые данные);
//Assert
Assert.IsNotNull(actual);
```

Метод Assert.IsNotNull() означает, что «переданный параметр **НЕ** должен быть null»: если он равен нулю, то тестовый пример не пройден.

Assert.IsInstanceOfType();

```
//Arrange
Задаем тестовые данные
тип expected = …;
//Act
ИмяКласса obj = new ИмяКласса();
тип  actual = obj.ИмяМетода(тестовые данные);
//Assert
Assert.IsInstanceOfType(actual, typeof(тип));
```

Метод Assert.IsInstanceOfType() означает, что «переданный параметр» должен иметь указанный тип

Assert.IsNotInstanceOfType();

```
//Arrange
Задаем тестовые данные
тип expected = …;
//Act
ИмяКласса obj = new ИмяКласса();
тип  actual = obj.ИмяМетода(тестовые данные);
//Assert
Assert.IsNotInstanceOfType(actual, typeof(тип));
```

Метод Assert.IsNotInstanceOfType() означает, что «переданный параметр» **НЕ** должен иметь указанный тип

Методы MS TEST

Обработка исключительных ситуаций

Assert.ThrowsException<>();

```
//Arrange
Задаем тестовые данные
//Act
ИмяКласса obj = new ИмяКласса();
Action actual = () => obj.ИмяМетода(тестовые данные);
//Assert
Assert.ThrowsException<ТипИсключения>( actual);
```

Метод Assert.ThrowsException<>(); означает, что введенных тестовых данных будет вызвано исключение

StringAssert.Contains()

```
//Arrange
Задаем тестовые данные
//Act
ИмяКласса obj = new ИмяКласса();
//Assert
try
    {
    тип  actual = obj.ИмяМетода(тестовые данные);
    }
catch (ТипИсключения ex)
    {
     StringAssert.Contains(ex.Message,
"текст_сообщения");
    }
```

Перехват ожидаемого исключения и проверка связанного с ним сообщения

[ExpectedException(typeof(ТипИсключения),"текст_сообщения")]

```
[ExpectedException(typeof(ТипИсключения),"текст_сообщ
ения")]
[TestMethod]

public void ИмяТеста()
      {
         //Arrange
         Задаем тестовые данные
         //Act
         ИмяКласса obj = new ИмяКласса();
         obj.ИмяМетода(тестовые данные);
      }
```

Проверка того, что выполняется вызов исключения и провеяется связанное с ним сообщение

Методы MS TEST

Работа с коллекциями

CollectionAssert.AreEqual();

```
//Arrange
Задаем тестовые данные
List<тип> expected = …;
//Act
ИмяКласса obj = new ИмяКласса();
List<тип> actual = obj.ИмяМетода(тестовые данные);
//Assert
CollectionAssert.AreEqual(expected, actual);
```

Метод CollectionAssert.AreEqual() проверяет равенство коллекций и выдает исключение, если две коллекции не равны. Равенство определяется как наличие одних и тех же элементов в одном и том же порядке и в одном количестве.

CollectionAssert.AreNotEqual();

```
//Arrange
Задаем тестовые данные
List<тип> expected = …;
//Act
ИмяКласса obj = new ИмяКласса();
List<тип> actual = obj.ИмяМетода(тестовые данные);
//Assert
CollectionAssert.AreNotEqual(expected, actual);
```

Метод CollectionAssert.AreNotEqual() проверяет НЕ равенство коллекций и выдает исключение, если две коллекции равны. Равенство определяется как наличие одних и тех же элементов в одном и том же порядке и в одном количестве.

CollectionAssert.AllItemsAreNotNull();

```
//Arrange
Задаем тестовые данные
List<тип> testList = …;
//Assert
CollectionAssert.AllItemsAreNotNull(testList);
```

Метод CollectionAssert.AllItemsAreNotNull() Проверяет, являются ли все элементы в указанной коллекции ненулевыми, и выдает исключение, если какой-либо элемент имеет значение null.

CollectionAssert.AllItemsAreUnique();

```
//Arrange
Задаем тестовые данные
List<тип> testList = …;
//Assert
CollectionAssert.AllItemsAreUnique(testList);
```

Метод CollectionAssert.AllItemsAreUnique() проверяет, являются ли все элементы в указанной коллекции уникальными или нет, и создает исключение, если какие-либо два элемента в коллекции равны.

Методы MS TEST

CollectionAssert.IsNotSubsetOf();

```
//Arrange
Задаем тестовые данные
List<тип> testList1 = …;
List<тип> testList2 = …;
//Assert
CollectionAssert.IsNotSubsetOf(testList1, testList2);
```

Метод CollectionAssert.IsNotSubsetOf() проверяет, не является ли одна коллекция подмножеством другой коллекции, и выдает исключение, если все элементы в подмножестве также входят в надмножество.

CollectionAssert.IsSubsetOf();

```
//Arrange
Задаем тестовые данные
List<тип> testList1 = …;
List<тип> testList2 = …;
//Assert
CollectionAssert.IsSubsetOf(testList1, testList2);
```

Метод CollectionAssert.IsSubsetOf() проверяет, является ли одна коллекция подмножеством другой коллекции, и выдает исключение, если какой-либо элемент в подмножестве не входит в надмножество.

CollectionAssert.AreEquivalent();

```
//Arrange
Задаем тестовые данные
List<тип> expected = …;
//Act
ИмяКласса obj = new ИмяКласса();
List<тип> actual = obj.ИмяМетода(тестовые данные);
//Assert
CollectionAssert.AreEquivalent(expected, actual);
```

Метод CollectionAssert.AreEquivalent() проверяет содержат ли две коллекции одни и те же элементы, и выдает исключение, если какая-либо коллекция содержит элемент, которого нет в другой коллекции.

CollectionAssert.AreNotEquivalent();

```
//Arrange
Задаем тестовые данные
List<тип> expected = …;
//Act
ИмяКласса obj = new ИмяКласса();
List<тип> actual = obj.ИмяМетода(тестовые данные);
//Assert
CollectionAssert.AreEquivalent(expected, actual);
```

Метод CollectionAssert.AreNotEquivalent() проверяет содержат ли две коллекции одни и те же элементы, и выдает исключение, если какая-либо коллекция содержит элемент, которого нет в другой коллекции.

Методы MS TEST

CollectionAssert.Contains();

```
//Arrange
Задаем тестовые данные
List<тип> testList = …;
//Assert
CollectionAssert.Contains(testList, тестовые данные);
```

Метод CollectionAssert.Contains() проверяет, содержит ли указанная коллекция указанный элемент, и выдает исключение, если элемента нет в коллекции.

CollectionAssert.DoesNotContain();

```
//Arrange
Задаем тестовые данные
List<тип> testList = …;
//Assert
CollectionAssert.DoesNotContain(testList, тестовые данные);
```

Метод CollectionAssert.DoesNotContain() проверяет, не содержит ли указанная коллекция указанный элемент, и выдает исключение, если элемент находится в коллекции.