



# ОСНОВИ МОДУЛЬНОГО ТЕСТУВАННЯ ОБ'ЄКТНО-ОРІЄНТОВАНОГО КОДУ

Лекція 09  
Об'єктно-орієнтоване програмування

# План лекції

---

- Принципи написання модульних тестів.
- Створення тестових наборів за допомогою MS Test v2.

## Automated Testing With MSTest V2

---



**Jason Roberts**

.NET MVP

@robertsjason dontcodetired.com





# ПРИНЦИПИ НАПИСАННЯ МОДУЛЬНИХ ТЕСТІВ

Питання 9.1.

Курс «Unit Testing for C# Developers»

# Автоматизоване тестування та його переваги

---



Test your code frequently, in less time

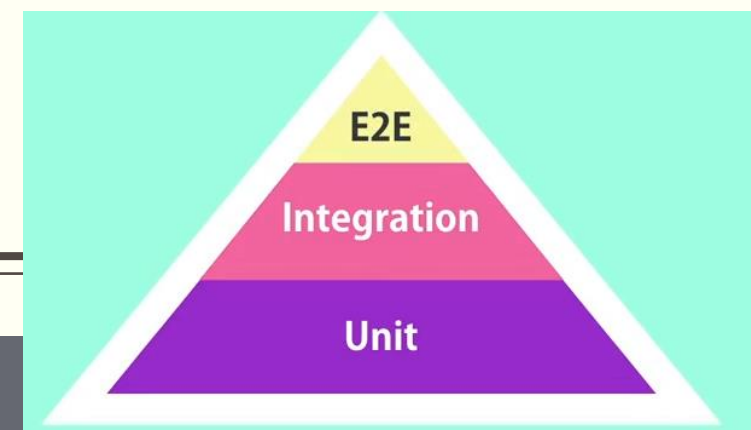
Catch bugs before deploying

Deploy with confidence

Refactor with confidence

Focus more on the quality

# Типи тестів: модульні, інтеграційні, наскрізні (end-to-end)



## UNIT TEST

Tests a unit of an application without its **external dependencies**



Cheap to write

Execute fast

Don't give a lot of confidence

## INTEGRATION TEST

Tests the application with its **external dependencies**

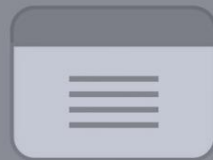


Take longer to execute

Give more confidence

## END-TO-END TEST

Drives an application through its UI.



Give you the greatest confidence

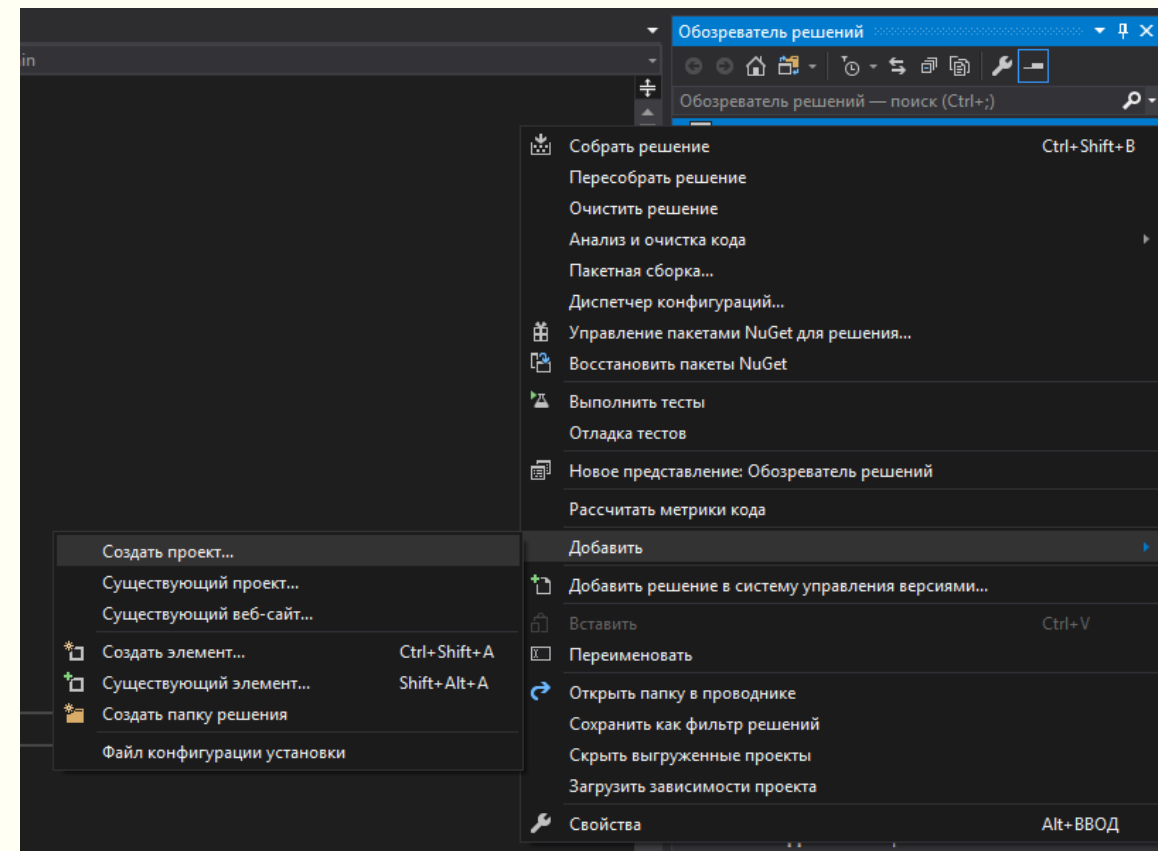
Very slow

Very brittle

# Інструменти тестування для C#-коду

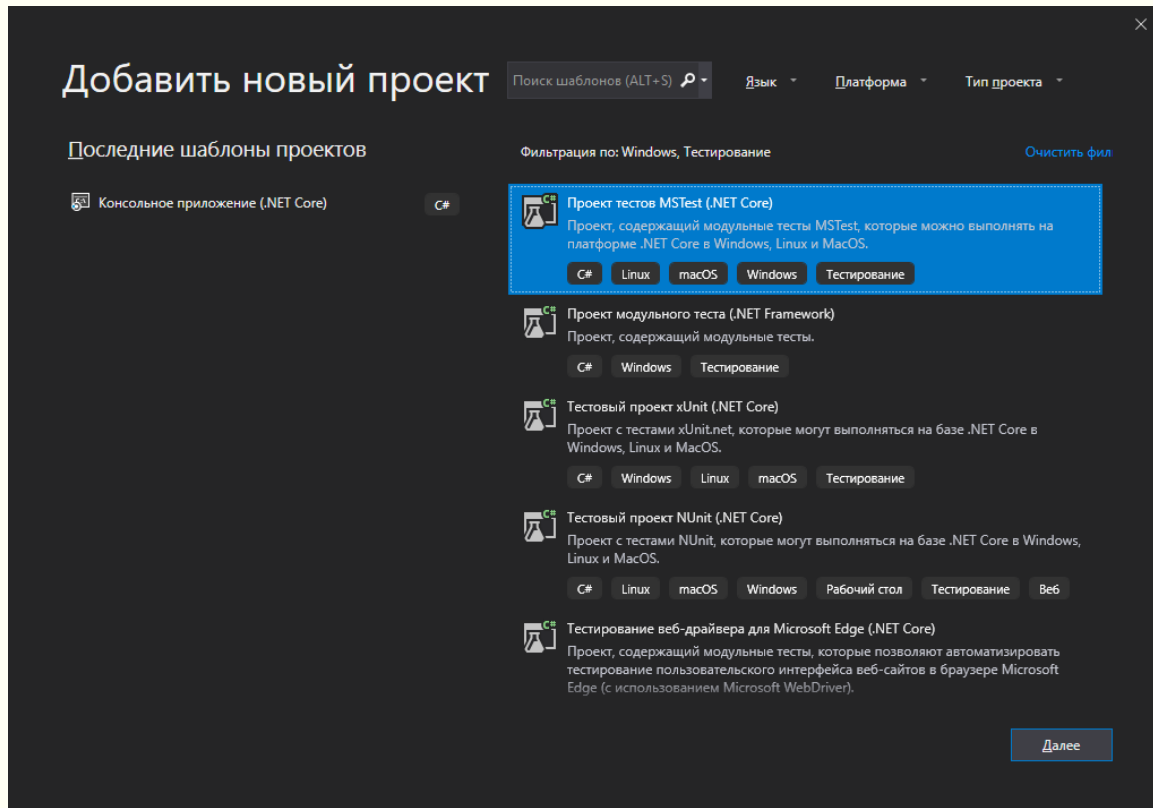
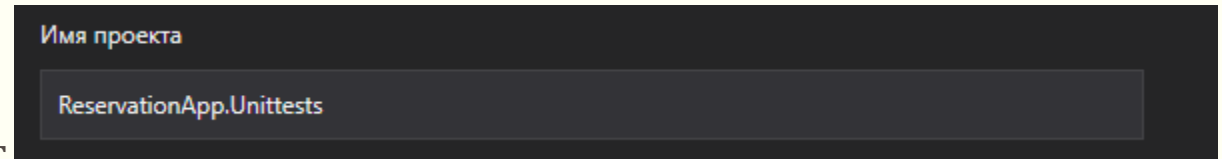
- NUnit, MSTest, xUnit, ReSharper, Rider
- Нехай пишеться додаток для бронювання столика в ресторані

```
1 namespace TestNinja.Fundamentals
2 {
3     public class Reservation
4     {
5         public User MadeBy { get; set; }
6
7         public bool CanBeCancelledBy(User user)
8         {
9             return (user.IsAdmin || MadeBy == user);
10        }
11    }
12
13    public class User
14    {
15        public bool IsAdmin { get; set; }
16    }
17 }
18
```



# Пишемо тестовий проект

- Додамо новий MSTest-проект
  - Іменування тестового випадку:
  - НазваМодуля\_Сценарій\_Очікуваний результат



```

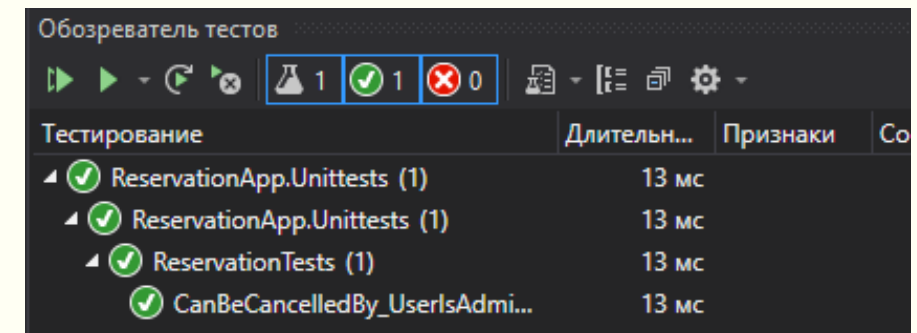
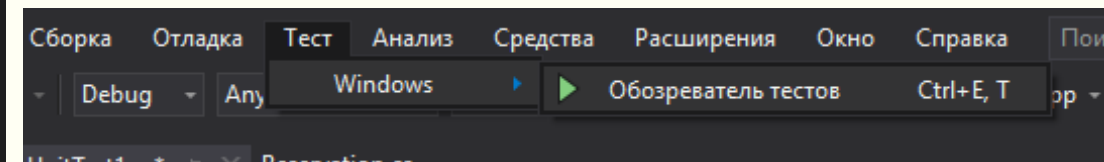
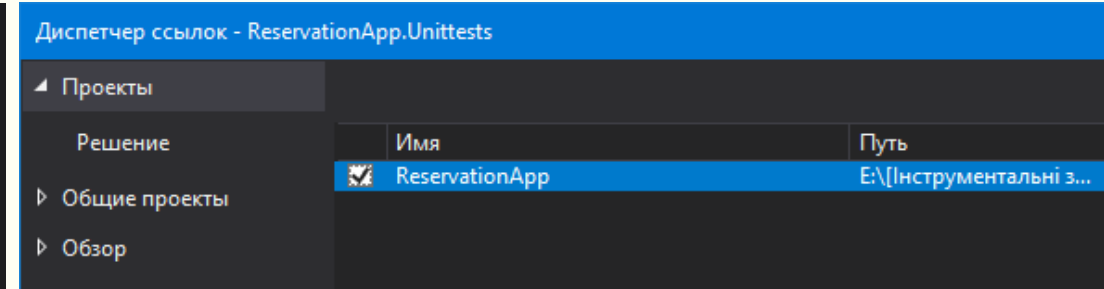
1  using System;
2  using Microsoft.VisualStudio.TestTools.UnitTesting;
3
4  namespace TestNinja.UnitTests
5  {
6      [TestClass]
7      public class ReservationTests
8      {
9          [TestMethod]
10         public void CanBeCancelledBy_UserIsAdmin_ReturnsTrue()
11         {
12         }
13     }
14 }
15

```

# Написання тестового випадку. Принцип Arrange – Act – Assert

```
1 using Microsoft.VisualStudio.TestTools.UnitTesting;
2
3 namespace ReservationApp.Unittests
4 {
5     [TestClass]
6     public class ReservationTests
7     {
8         [TestMethod]
9         public void CanBeCancelledBy_UserIsAdmin_ReturnsTrue()
10        {
11            // Arrange
12            var reservation = new Reservation();
13
14            // Act
15            var result = reservation.CanBeCancelledBy(new User { IsAdmin = true });
16
17            // Assert
18            Assert.IsTrue(result);
19        }
20    }
21 }
```

- подготовка — объявляет и создает экземпляры переменных для ввода и вывода;
- действие — выполняет тестируемый модуль;
- проверка — производит одну или несколько проверок результатов работы модуля.





# Написання тестового випадку

```
public bool CanBeCancelledBy(User user)
{
    return !(user.IsAdmin || MadeBy == user);
}
```

Обозреватель тестов

▶ ▶ ▶ ▶ ▶ 1 0 1

Тестирование	Длительн...	Признаки	Сообщение об ошибке
✖ ReservationApp.Unittests (1)	91 мс		
✖ ReservationApp.Unittests (1)	91 мс		
✖ ReservationTests (1)	91 мс		

- Імітуємо помилку

- Практика: дописати решту тестових випадків

- Характеристики хорошого модульного тесту: простота, відсутність управляючих інструкцій, ізолюваність від інших тестів
- Тестовані функції/методи можна поділити на 2 види:
  - Query-функції: повертають деяке значення; тест перевіряє, чи відповідає це значення очікуваному результату; різні ходи виконання (execution path) – різні тестові сценарії;
  - Command-функції: виконують зміну в системі – дію (action – зміна стану об'єкта, запис у БД, виклик веб-служби, відправка повідомлення в чергу повідомлень тощо); можуть також повертати значення;
    - При зміні стану перевіряють поточний стан об'єкта
    - При підключенні зовнішніх ресурсів перевіряють правильність виклику (звернення) за посиланням на ці ресурси (external dependencies)
- Ніколи не тестуються:
  - Language features
  - Сторонній код (3rd-party code)
- Тести підганяються під вимоги, а не під реалізований код!

# Атрибути фреймворків для тестування

NUnit	MSTest 2.x	xUnit 2.x	Коментарі
[Test]	[TestMethod]	[Fact]	Позначає тестовий метод
[TestFixture]	[TestClass]	--	Позначає тестовий клас
[SetUp]	[TestInitialize]	Конструктор	Спрацьовує перед кожним тестовим випадком
[TearDown]	[TestCleanup]	IDisposable. Dispose	Спрацьовує після кожного тестового випадку
[OneTimeSetUp]	[ClassInitialize]	IClassFixture<T>	Метод спрацьовує 1 раз перед запуском тестового набору
[OneTimeTearDown]	[ClassCleanup]	IClassFixture	Метод спрацьовує 1 раз після запуску тестового набору
[Ignore("reason")]	[Ignore]	[Fact(Skip="reason")]	Пропуск тестового випадку
[Property]	[TestProperty]	[Trait]	Встановлює довільні метадані для тесту
[Theory]	[DataRow]	[Theory]	Конфігурує керовані даними (data-driven) тести
[Category("")]	[TestCategory("")]	[Trait("Category", "")]	Категоризує тестові випадки або класи

# Test Execution Workflow. Атрибути для налаштування тестів (MSTest)

---

```
[TestClass]
public class YourUnitTests
{
    [AssemblyInitialize]
    public static void AssemblyInit(TestContext context)
    {
        // Executes once before the test run. (Optional)
    }

    [ClassInitialize]
    public static void TestFixtureSetup(TestContext context)
    {
        // Executes once for the test class. (Optional)
    }

    [TestInitialize]
    public void Setup()
    {
        // Runs before each test. (Optional)
    }

    [AssemblyCleanup]
    public static void AssemblyCleanup()
    {
        // Executes once after the test run. (Optional)
    }
}
```

```
[ClassCleanup]
public static void TestFixtureTearDown()
{
    // Runs once after all tests in this class are executed.
    (Optional)
    // Not guaranteed that it executes instantly after all tests
    from the class.
}

[TestCleanup]
public void TearDown()
{
    // Runs after each test. (Optional)
}
// Mark that this is a unit test method. (Required)

[TestMethod]
public void YouTestMethod()
{
    // Your test code goes here.
}
}
```

# Твердження (assertions)

---

```
Assert.AreEqual(28, _actualFuel); // Tests whether the specified values are equal.
Assert.AreNotEqual(28, _actualFuel); // Tests whether the specified values are unequal. Same as AreEqual for numeric values.
Assert.AreSame(_expectedRocket, _actualRocket); // Tests whether the specified objects both refer to the same object
Assert.AreNotSame(_expectedRocket, _actualRocket); // Tests whether the specified objects refer to different objects
Assert.IsTrue(_isThereEnoughFuel); // Tests whether the specified condition is true
Assert.IsFalse(_isThereEnoughFuel); // Tests whether the specified condition is false
Assert.IsNull(_actualRocket); // Tests whether the specified object is null
Assert.IsNotNull(_actualRocket); // Tests whether the specified object is non-null
Assert.IsInstanceOfType(_actualRocket, typeof(Falcon9Rocket)); // Tests whether the specified object is an instance of the expected type
Assert.IsNotInstanceOfType(_actualRocket, typeof(Falcon9Rocket)); // Tests whether the specified object is not an instance of type
StringAssert.Contains(_expectedBellatrixTitle, "Bellatrix"); // Tests whether the specified string contains the specified substring
StringAssert.StartsWith(_expectedBellatrixTitle, "Bellatrix"); // Tests whether the specified string begins with the specified substring
StringAssert.Matches("(281)388-0388", @"(?d{3})?-? *d{3}-? *-?d{4}"); // Tests whether the specified string matches a regular expression
StringAssert.DoesNotMatch("(281)388-0388", @"(?d{3})?-? *d{3}-? *-?d{4}"); // Tests whether the specified string does not match a regular expression
CollectionAssert.AreEqual(_expectedRockets, _actualRockets); // Tests whether the specified collections have the same elements in the same order and quantity.
CollectionAssert.AreNotEqual(_expectedRockets, _actualRockets); // Tests whether the specified collections does not have the same elements or the elements are in a different order and quantity.
CollectionAssert.AreEqual(_expectedRockets, _actualRockets); // Tests whether two collections contain the same elements.
CollectionAssert.AreNotEquivalent(_expectedRockets, _actualRockets); // Tests whether two collections contain different elements.
CollectionAssert.AllItemsAreInstancesOfType(_expectedRockets, _actualRockets); // Tests whether all elements in the specified collection are instances of the expected type
CollectionAssert.AllItemsAreNotNull(_expectedRockets); // Tests whether all items in the specified collection are non-null
CollectionAssert.AllItemsAreUnique(_expectedRockets); // Tests whether all items in the specified collection are unique
CollectionAssert.Contains(_actualRockets, falcon9); // Tests whether the specified collection contains the specified element
CollectionAssert.DoesNotContain(_actualRockets, falcon9); // Tests whether the specified collection does not contain the specified element
CollectionAssert.IsSubsetOf(_expectedRockets, _actualRockets); // Tests whether one collection is a subset of another collection
CollectionAssert.IsNotSubsetOf(_expectedRockets, _actualRockets); // Tests whether one collection is not a subset of another collection
Assert.ThrowsException<ArgumentNullException>(() => new Regex(null)); // Tests whether the code specified by delegate throws exact given exception of type T
```

# Конкретні та загальні тести. Тестування рядків

## ■ Тестований код

```
1 namespace TestNinja.Fundamentals
2 {
3     public class HtmlFormatter
4     {
5         public string FormatAsBold(string content)
6         {
7             return $"<strong>{content}</strong>";
8         }
9     }
10 }
```

## ■ Конкретний тест:

- `Assert.That(result, Is.EqualTo("<strong>abc</strong>"));`

## ■ Загальний тест:

- `Assert.That(result, Does.StartWith("<strong>"));`
- `Assert.That(result, Does.EndsWith("</strong>"));`
- `Assert.That(result, Does.Contain("abc"));`

## Повний тест в NUnit

```
[TestFixture]
public class HtmlFormatterTests
{
    [Test]
    public void FormatAsBold_WhenCalled_ShouldEncloseStringWithStrongElement()
    {
        var formatter = new HtmlFormatter();

        var result = formatter.FormatAsBold("abc");

        // this is specific: in this case, specific is good
        // in some cases, it may be more appropriate to be more general

        Assert.That(result, Is.EqualTo("<strong>abc</strong>"));

        // you can also ignore case
        Assert.That(result, Is.EqualTo("<strong>ABC</strong>").IgnoreCase);

        // examples of more general assertions that aren't good enough in this case
        // but they give you some ideas of how to be more general
        Assert.That(result, Does.StartWith("<strong>"));
        Assert.That(result, Does.EndsWith("</strong>"));
        Assert.That(result, Does.Contain("abc"));
    }
}
```

# Конкретні та загальні тести. Тестування масивів та колекцій

---

## ■ Тестований код

```
public class Math
{
    public int Add(int a, int b)
    {
        return a + b;
    }

    public int Max(int a, int b)
    {
        return (a > b) ? a : b;
    }

    public IEnumerable<int> GetOddNumbers(int limit)
    {
        for (var i = 0; i <= limit; i++)
            if (i % 2 != 0)
                yield return i;
    }
}
```

## Налаштування тестів

```
private Math math;

[SetUp]
protected void SetUp()
{
    math = new Math();
}
```

- `var result = _math.GetOddNumbers(5);`
- Загальні тести:
  - `Assert.That(result, Is.Not.Empty);`
  - `Assert.That(result.Count(), Is.EqualTo(3));`
- Конкретний тест:
  - `Assert.That(result, Is.EquivalentTo( new [] {1, 3, 5} ));`



# Конкретні та загальні тести. Тестування масивів та колекцій

---

- Загальний вигляд можливого тесту (NUnit)

```
[Test]
public void GetOddNumbers_LimitIsGreaterThanZero_ReturnOddNumbersUpToLimit()
{
    var result = _math.GetOddNumbers(5);

    Assert.That(result, Is.Not.Empty);

    Assert.That(result.Count(), Is.EqualTo(3));

    Assert.That(result, Does.Contain(1));
    Assert.That(result, Does.Contain(3));
    Assert.That(result, Does.Contain(5));

    Assert.That(result, Is.EquivalentTo(new [] {1, 3, 5}));

    Assert.That(result, Is.Ordered);
    Assert.That(result, Is.Unique);
}
```

# Конкретні та загальні тести.

## Перевірка типу значення, яке повертає метод

---

- Тестований код
  - Має 2 шляхи виконання

```
public class CustomerController
{
    public ActionResult GetCustomer(int id)
    {
        if (id == 0)
            return new NotFound();

        return new Ok();
    }
}

public class ActionResult { }

public class NotFound : ActionResult { }

public class Ok : ActionResult { }
```

```
private CustomerController _controller;

[SetUp]
public void SetUp()
{
    _controller = new CustomerController();
}

[Test]
public void GetCustomer_IdIsZero_ReturnNotFound()
{
    var result = _controller.GetCustomer(0);

    // Not Found
    Assert.That(result, Is.TypeOf<NotFound>());

    // NotFound or one of its derivatives
    //Assert.That(result, Is.InstanceOf<NotFound>());
}

[Test]
public void GetCustomer_IdIsNotZero_ReturnOk()
{
    var result = _controller.GetCustomer(1);

    Assert.That(result, Is.TypeOf<Ok>());
}
```



# Конкретні та загальні тести. Тестування void-методів

---

- Зазвичай такими є command-функції
- Тестований код:

```
public string LastError { get; set; }

public event EventHandler<Guid> ErrorLogged;

public void Log(string error)
{
    if (String.IsNullOrEmpty(error))
        throw new ArgumentNullException();

    LastError = error;

    // Write the log to a storage
    // ...

    ErrorLogged?.Invoke(this, Guid.NewGuid());
}
```

- Тестується зміна стану об'єкта (властивості LastError)

```
private ErrorLogger _logger;

[SetUp]
public void SetUp()
{
    _logger = new ErrorLogger();
}

[Test]
public void Log_WhenCalled_SetTheLastErrorProperty()
{
    _logger.Log("a");

    Assert.That(_logger.LastError, Is.EqualTo("a"));
}
```

# Конкретні та загальні тести. Тестування методів, які викидають винятки

---

- Тестуються варіанти: `null`, `""`, `" "`. Маємо параметризований тест
  - Для тестування методів, які викидають винятки, використовують делегати або лямбда-вирази

```
[Test]
[TestCase(null)]
[TestCase("")]
[TestCase(" ")]
public void Log_InvalidError_ThrowArgumentNullException(string error)
{
    Assert.That(() => _logger.Log(error), Throws.ArgumentNullException);
}
```

# Конкретні та загальні тести. Тестування методів, які викликають (raise) подію

---

- Перевіримо спрацювання події ErrorLogged
  - Спочатку підписуємось на подію
  - Перевіряємо зміну значення id (Not Empty) після обробки події

```
[Test]
public void Log_ValidError_RaiseErrorLoggedEvent()
{
    var id = Guid.Empty;
    _logger.ErrorLogged += (sender, args) => { id = args; };

    _logger.Log("a");

    Assert.That(id, Is.Not.EqualTo(Guid.Empty));
}
```

# Конкретні та загальні тести. Тестування приватних та захищених методів

---

- Приватні та захищені атрибути класу описують деталі реалізації.
  - Якщо написати тест до них, вони з ним стануть зв'язаними (coupling)
  - Зміна деталей реалізації зведе тест нанівець
- Для демонстрації виведемо виклик події в окремий захищений віртуальний метод та викличемо його.
  - Тестувати потрібно метод Log(), а не OnErrorLogged()

```
// Write the log to a storage
// ...

OnErrorLogged(Guid.NewGuid());
}

protected virtual void OnErrorLogged(Guid errorId)
{
    ErrorLogged?.Invoke(this, errorId);
}
```

# Покриття коду тестами (Code Coverage)

- Частини коду можна виключати з перевірки покриття тестами:

```
public class ExampleClass1
{
    [ExcludeFromCodeCoverage]
    void ExampleMethod() {...}

    [ExcludeFromCodeCoverage] // exclude property
    int ExampleProperty1
    { get {...} set {...} }

    int ExampleProperty2
    {
        get
        {
            ...
        }
        [ExcludeFromCodeCoverage] // exclude setter
        set
        {
            ...
        }
    }
}

[ExcludeFromCodeCoverage]
class ExampleClass2 { ... }
```

The screenshot illustrates the process of analyzing code coverage in Visual Studio. The 'Test' menu is open, showing options like 'Run', 'Debug', 'Test Settings', 'Analyze Code Coverage', and 'Windows'. The 'Test Explorer' window shows a list of passed tests: 'QuickNonZero' (15 ms), 'RootTestNeg...' (13 ms), and 'SignatureTest' (1 ms). The 'Code Coverage Results' window displays a table with coverage data for the test run 'ctsoasm\_MAIN50531 2012-06-07 02...'. The table shows that 44 lines of code were not covered (80.00% not covered, 11 lines covered). The 'Fabrikam.Math' assembly also shows 7 lines not covered (50.00% not covered, 7 lines covered). A code snippet for 'SquareRoot' is shown with 'Not covered' and 'Covered' annotations. A 'Turn on coloring' button is visible in the bottom right corner of the code editor.

Hierarchy	Not Cov...	Not Covered (%...	Cov...
ctsoasm_MAIN50531 201...	44	80.00%	11
fabrikam.math.dll	7	50.00%	7
Fabrikam.Math	7	50.00%	7



# **ДЯКУЮ ЗА УВАГУ!**

**Наступне питання: Створення тестових наборів за допомогою MS Test v2**