

Isolation Frameworks



- 1)** Dependencies, interaction testing
- 2)** Fakes, Stubs & Mocks
- 3)** Moq, NSubstitute, FakeItEasy

TDD (реализовано в лекциях 1,2)

Задача

Программа для нахождения корней
квадратного уравнения $ax^2 + bx + c = 0$

Тесты

$$\begin{array}{ll} x^2 + x - 6 = 0 & \Rightarrow 2, -3 \\ x^2 + 2x + 1 = 0 & \Rightarrow -1, -1 \\ x^2 + 0x - 4 = 0 & \Rightarrow 2, -2 \end{array}$$

(первая итерация)

**Проектирование /
рефакторинг**

Класс `EquationSolver`
с методом `Solve()`

Код

TDD (реализовано в лекциях 1,2)

**Требование
уточняется**

Надо также проверять явно ситуацию отсутствия вещественных корней + порядок корней не важен

Тесты

Корректируем тесты под новые требования + сами что-то вспомнили допроверить

$x^2 + x + 1 = 0$ \Rightarrow Exception
 $4x^2 = 0$ \Rightarrow 0, 0

...

**Проектирование /
рефакторинг**

Корректируем код под новые тесты:

- Вводим exception для ситуации с комплексными корнями
- Меняем сигнатуру метода
- ...

Код

TDD (для больших порядков ~ класс-помощник)

Требование
уточняется

Нужно также решать уравнения больших порядков

Тесты

Корректируем тесты под новые требования

Проектирование /
рефакторинг

Корректируем код под новые тесты

Код

Класс PolynomialRoots
с методом Calculate()

Зависимости (Dependencies)

**Юнит-тестирование -
тестирование отдельных классов и методов**

**Но классы часто находятся в отношении
зависимости с другими классами...**

...или взаимодействуют с внешними ресурсами:

- **Файловой системой**
- **Веб-сервисами**
- **Сетью**
- **Потоками**
- **и т.д.**

Внешние зависимости

Внешняя зависимость (External dependency) -

объект в системе, с которым взаимодействует тестируемый участок кода, и которым программист не может управлять (файловая система, потоки, память, время и т.д.)

Заглушка (Stub) -

управляемая программистом замена внешней зависимости в системе

Seams (ШВЫ)

Шов (Seam) –

участок в коде, где поведение может быть изменено без изменения самого этого участка.

Принцип обращения зависимостей (Inversion of Control) предлагает вместо прямого обращения к классам и объектам (путем доступа по статическим методам или создания экземпляров) предоставлять точки для внедрения зависимостей.

Возвращаясь к нашим баранам...

Слегка изменим класс **EquationSolver**.

Теперь он обрабатывает не строго 3 коэффициента, а произвольный массив коэффициентов

```
public class EquationSolver
{
    public double[] Solve(double[] coeffs)
    {
        // ....
    }
}
```


Возвращаясь к нашим баранам...

```
public double[] Solve(double[] coeffs)
{
    if (coeffs.Length > 3)
    {
        PolynomialRoots r = new PolynomialRoots();
        return r.Calculate(coeffs);
    }

    double a = coeffs[0];
    double b = coeffs[1];
    double c = coeffs[2];

    var d = b * b - 4 * a * c;

    if (d < 0)
    {
        throw new Exception("No real roots!");
    }

    d = Math.Sqrt(d);

    var roots = new double[2];
    roots[0] = (-b + d) / (2 * a);
    roots[1] = (-b - d) / (2 * a);

    return roots;
}
```

Класс PolynomialRoots
занимается расчетом корней
по хитроумным схемам

Како и было

Класс PolynomialRoots, от которого зависит EquationSolver

А вот и он:

```
public class PolynomialRoots
{
    public double[] Calculate(double[] p)
    {
        // здесь будет код какого-нибудь вычислительного метода
        // для нахождения корней полинома
        // (например, метода Лагерра или Дуранда-Крамера)

        return null;
    }
}
```



Но программист Пумба с дипломом признанного государства Бурунди еще пока пишет обильный код начинки этого класса...

Нужно дать возможность подменять конкретные реализации PolynomialRoots

Вводим общий интерфейс

```
public interface IRoots
{
    double[] calculate(double[] p);
}
```

И делаем PolynomialRoots классом, который реализует данный интерфейс

```
public class PolynomialRoots : IRoots
{
    public double[] calculate(double[] p)
    {
        // здесь будет код какого-нибудь вычислительного метода
        // для нахождения корней полинома
        // (например, метода Лагерра или Дуранда-Крамера)

        return null;
    }
}
```

Теперь, в общем случае, будет работать не PolynomialRoots, а любая реализация IRoots

```
public class EquationSolver
{
    private readonly IRoots _rootsEvaluator;

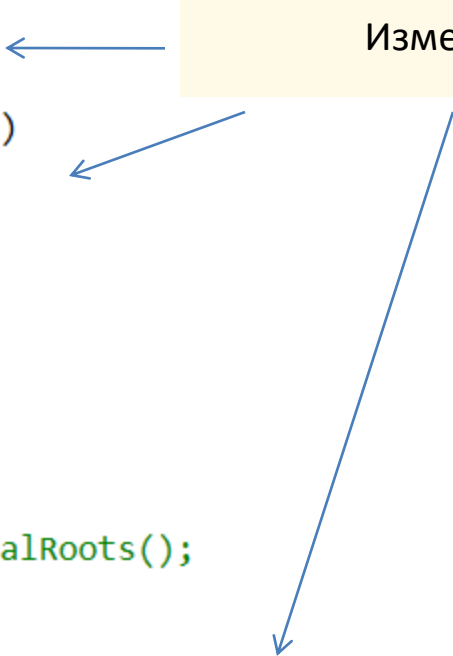
    public EquationSolver(IRoots rootsEvaluator)
    {
        _rootsEvaluator = rootsEvaluator;
    }

    public double[] Solve(double[] coeffs)
    {
        if (coeffs.Length > 3)
        {
            // PolynomialRoots r = new PolynomialRoots();
            // return r.Calculate(coeffs);

            return _rootsEvaluator.Calculate(coeffs);
        }

        // ....
    }
}
```

Изменения



Доберемся, наконец, до тестов...

Если мы дадим в конструктор EquationSolver, то тест не пройдет, т.к. реализация Пумбы еще не валидна.

Нам нужна своя заглушка для тестов...

```
[TestFixture]
public class EquationSolvingTestClass
{
    readonly EquationSolver _solver = new EquationSolver(new PolynomialRoots());

    [Test]
    public void TestMoreThanThreeCoefficients()
    {
        double[] roots = _solver.Solve(new double[] { 1, 3, -5, -15, 4, 12 });

        Assert.That(roots, Is.EquivalentTo(new double[] { -3, -2, -1, 1, 2 }));
    }
}
```

Доберемся, наконец, до тестов...

Run All | Run... ▾ | Playlist : All Tests ▾

Failed Tests (1)

✖ TestMoreThatThreeCoefficients 9 ms

Passed Tests (7)

✓ TestNoRealRoots 14 ms

✓ TestNoUnknowns 9 ms

✓ TestOneRoot 7 ms

✓ TestTwoDifferentRoots(1,1,-6,2,-3) 1 ms

✓ TestTwoDifferentRoots(1,-3,0,3,0) < 1 ms

✓ TestTwoDifferentRoots(1,5,4,-4,-1) < 1 ms

✓ TestZeroCoefficients < 1 ms

TestMoreThatThreeCoefficients

Source: [EquationSolvingTestClass.cs line 21](#)

✖ Test Failed - TestMoreThatThreeCoefficients

Message: System.ArgumentException : The actual value must be an IEnumerable

Имя параметра: actual

Elapsed time: 9 ms

Stack Trace:

CollectionConstraint.ApplyTo[TActual](TActual
Assert.That[TActual](TActual actual, IResolveC
[EquationSolvingTestClass.TestMoreThatThree](#)

С нашей заглушкой

```
class FakeRoots : IRoots
```

```
{  
    public double[] Calculate(double[] p)  
    {  
        return new double[] { -3, -2, -1, 1, 2 };  
    }  
}
```

Вот наша заглушка для тестов:

Пока мы тестируем для уравнения

$$x^5 + 3x^4 - 5x^3 - 15x^2 + 4x + 12 = 0$$

Корни прошиваем: 1, 2, -3, -1, -2

```
[TestFixture]
```

```
public class EquationSolvingTestClass
```

```
{  
    readonly EquationSolver _solver = new EquationSolver(new FakeRoots());
```

```
[Test]
```

```
public void TestMoreThanThreeCoefficients()
```

```
{  
    double[] roots = _solver.Solve(new double[] { 1, 3, -5, -15, 4, 12 });  
  
    Assert.That(roots, Is.EquivalentTo(new double[] { -3, -2, -1, 1, 2 }));  
}
```

С нашей заглушкой

Run All | Run... ▾ | Playlist : All Tests ▾

▲ Passed Tests (8)

✓ TestMoreThatThreeCoefficients	11 ms
✓ TestNoRealRoots	6 ms
✓ TestNoUnknowns	9 ms
✓ TestOneRoot	3 ms
✓ TestTwoDifferentRoots(1,1,-6,2,-3)	1 ms
✓ TestTwoDifferentRoots(1,-3,0,3,0)	< 1 ms
✓ TestTwoDifferentRoots(1,5,4,-4,-1)	< 1 ms
✓ TestZeroCoefficients	< 1 ms

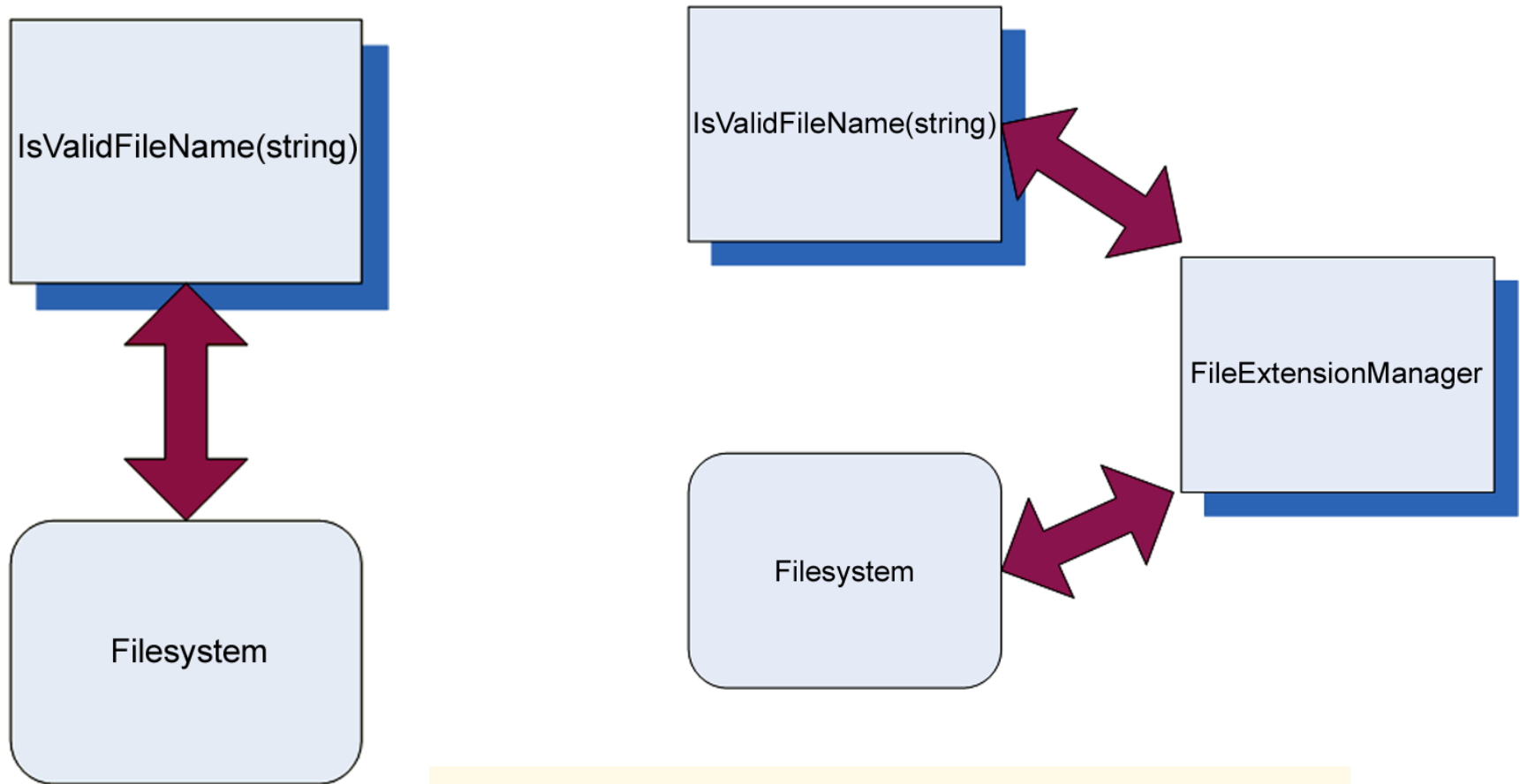
TestMoreThatThreeCoefficients

Source: [EquationSolvingTestClass.cs](#) line 21

✓ Test Passed - TestMoreThatThreeCoefficients

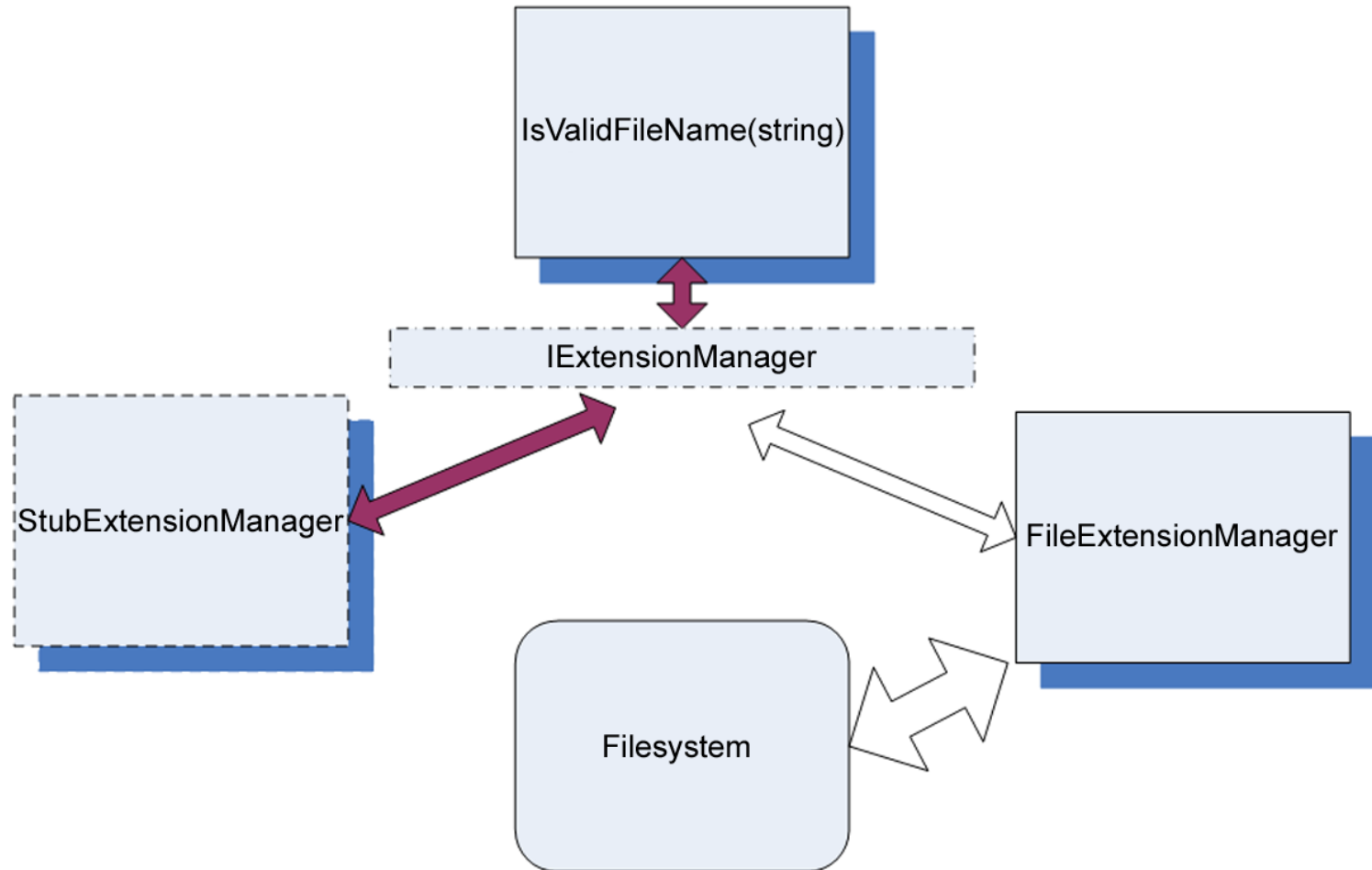
Elapsed time: 11 ms

Теперь пример из книги [R.Osherove]



Здесь обращение к файловой системе: медленно + это уже элемент интеграционного тестирования, поэтому можно сделать заглушку

Пример [R.Osherove]



Внешние зависимости

```
public bool IsValidLogFileName(string fileName)
{
    FileExtensionManager mgr = new FileExtensionManager();
    return mgr.IsValid(fileName);
}
```

```
class FileExtensionManager
{
    public bool IsValid(string fileName)
    {
        //read some file here
    }
}
```

Внешние зависимости

```
public interface IExtensionManager
{
    bool IsValid (string fileName);
}
```

```
public class FileExtensionManager : IExtensionManager
{
    public bool IsValid(string fileName)
    {
        // ...
    }
}
```

```
public class AlwaysValidFakeExtensionManager : IExtensionManager
{
    public bool IsValid(string fileName)
    {
        return true;
    }
}
```

Внешние зависимости

```
//the unit of work under test:
public bool IsValidLogFileName(string fileName)
{
    IExtensionManager mgr = new FileExtensionManager();

    return mgr.IsValid(fileName);
}
```

Внедрение зависимости (DI)

- Receive an interface at the constructor level and save it in a field for later use.
- Receive an interface as a property get or set and save it in a field for later use.
- Receive an interface just before the call in the method under test using one of the following:
 - A parameter to the method (*parameter injection*)
 - A factory class
 - A local factory method
 - Variations on the preceding techniques

Внедрение зависимости в конструкторе

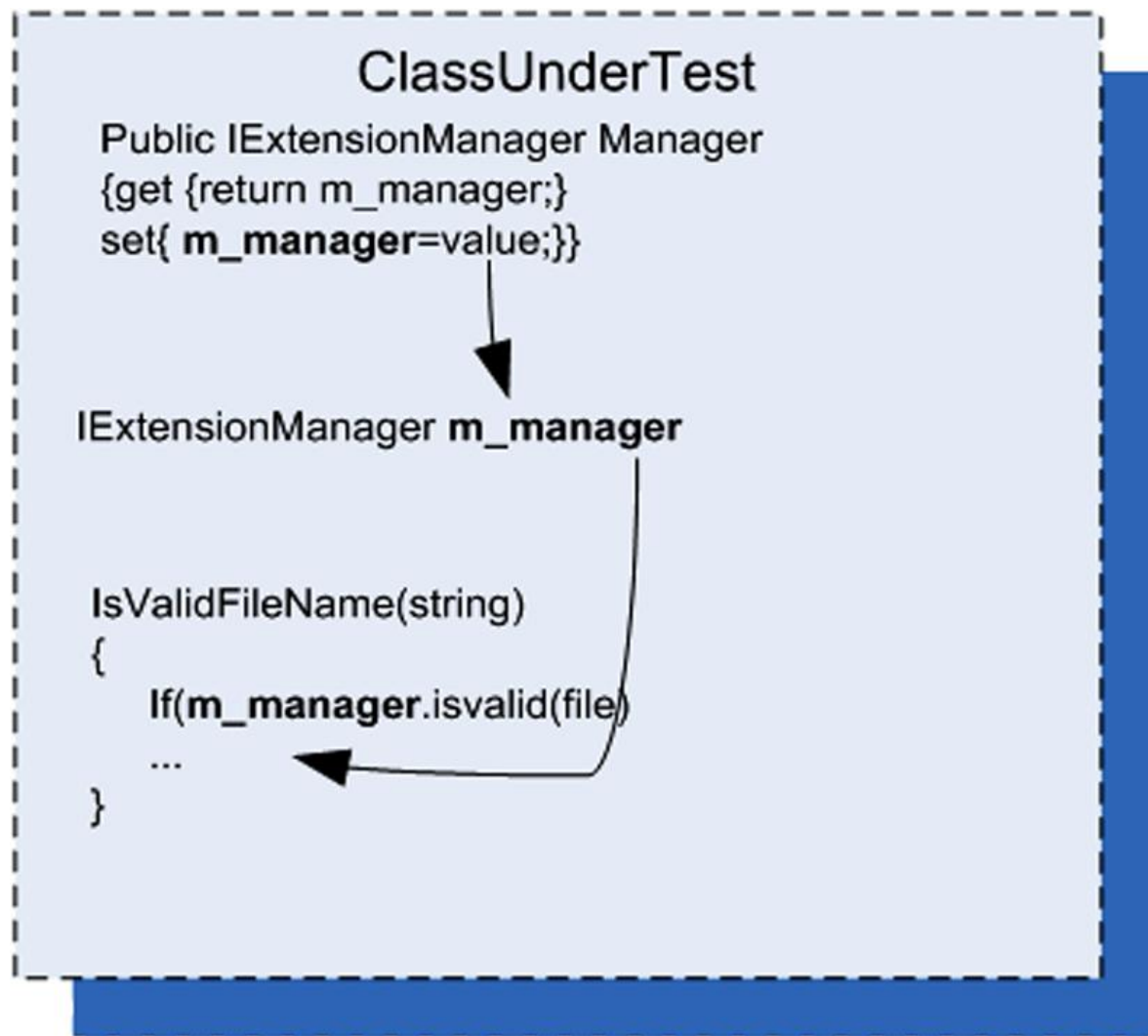
```
[TestFixture]
public class LogAnalyzerTests
{
    [Test]
    public void IsValidFileName_NameSupportedExtension_ReturnsTrue()
    {
        FakeExtensionManager myFakeManager = new FakeExtensionManager();
        myFakeManager.WillBeValid = true;

        LogAnalyzer log = new LogAnalyzer (myFakeManager);

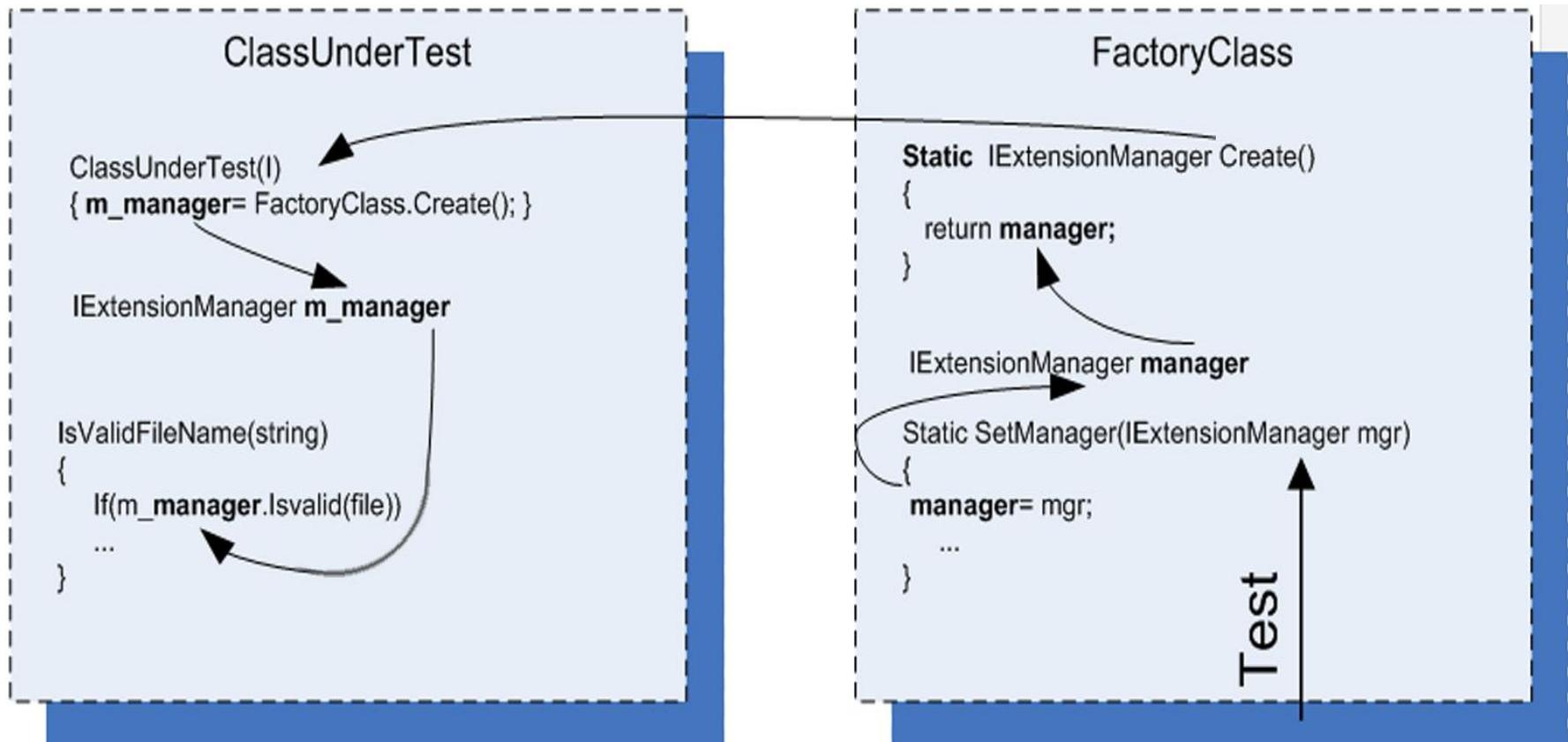
        bool result = log.IsValidLogFileName("short.ext");
        Assert.True(result);
    }
}

internal class FakeExtensionManager : IExtensionManager
{
    public bool WillBeValid = false;
    public bool IsValid(string fileName)
    {
        return WillBeValid;
    }
}
```

Внедрение зависимости в сеттере

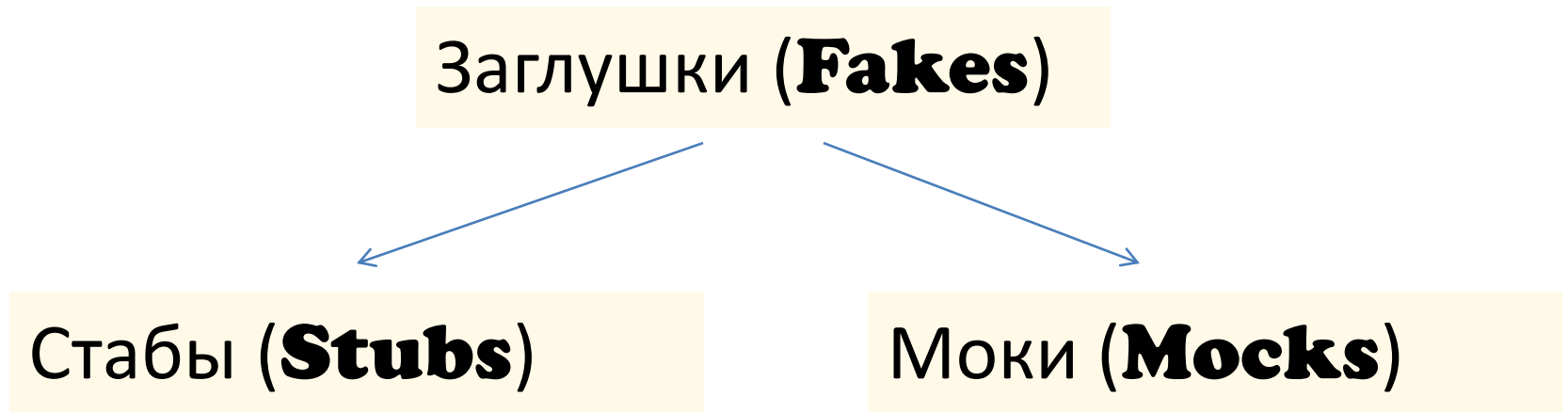


Фабричный метод

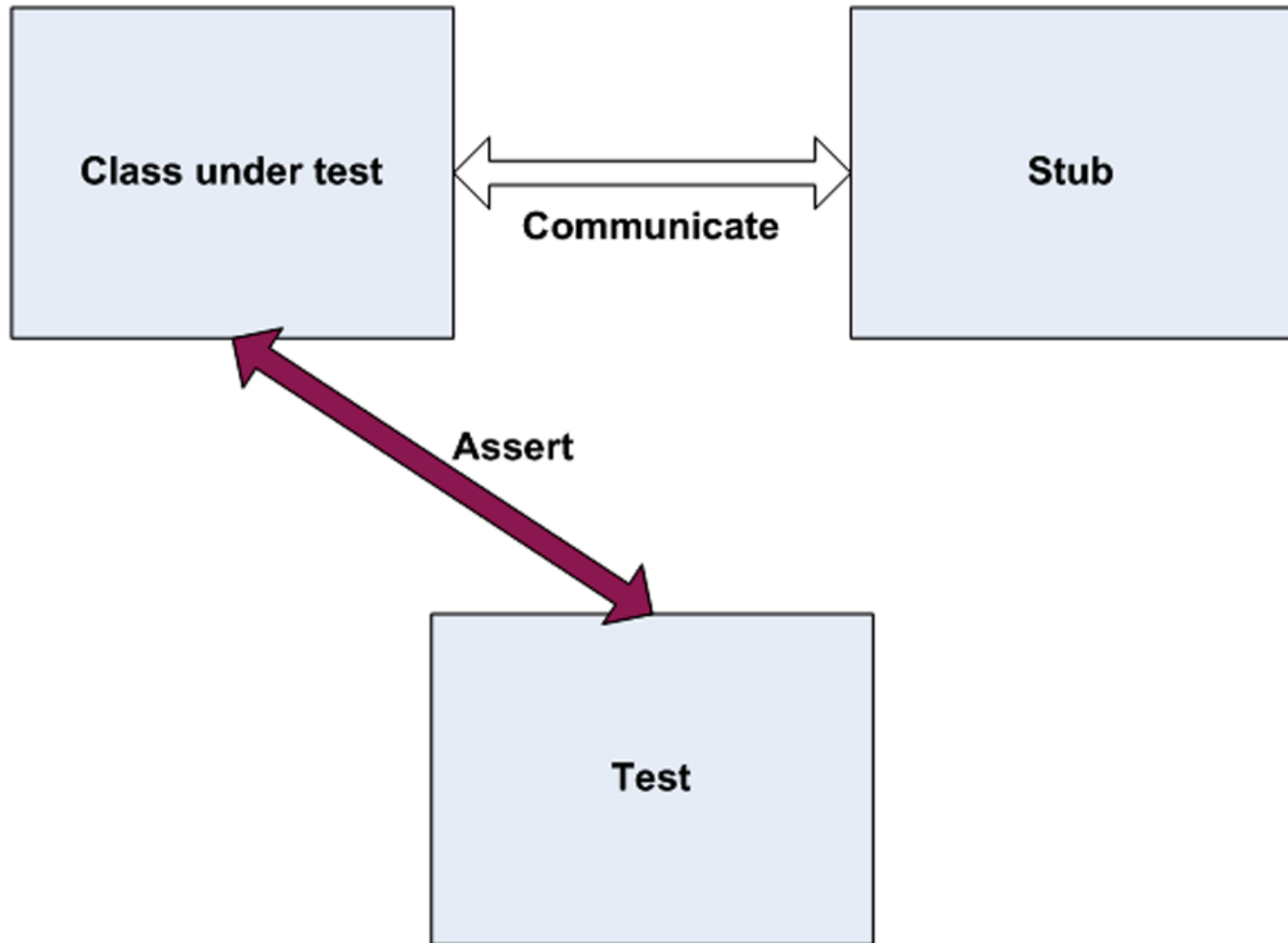


Interaction Testing

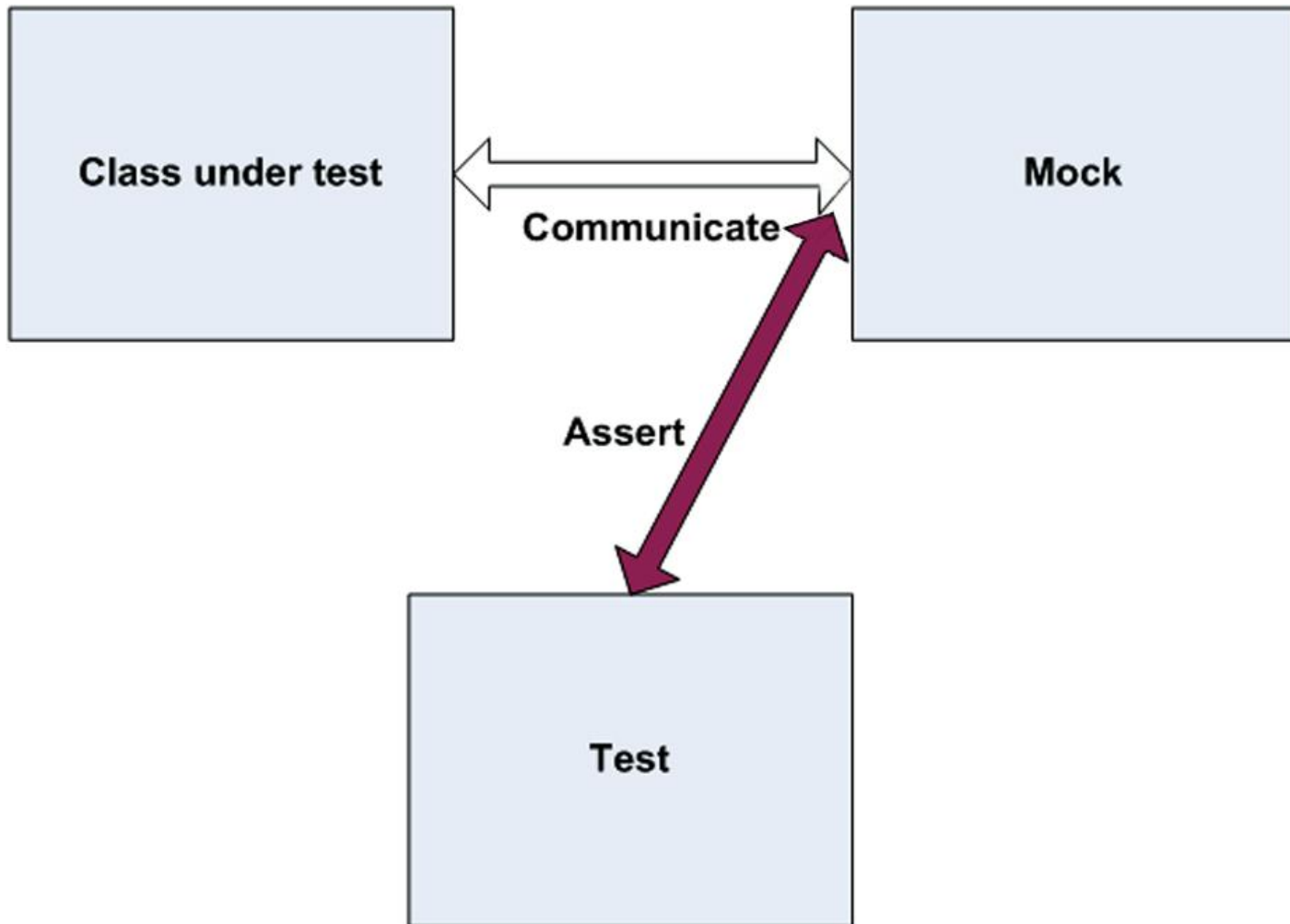
DEFINITION *Interaction testing* is testing how an object sends messages (calls methods) to other objects. You use interaction testing when calling another object is the end result of a specific unit of work.



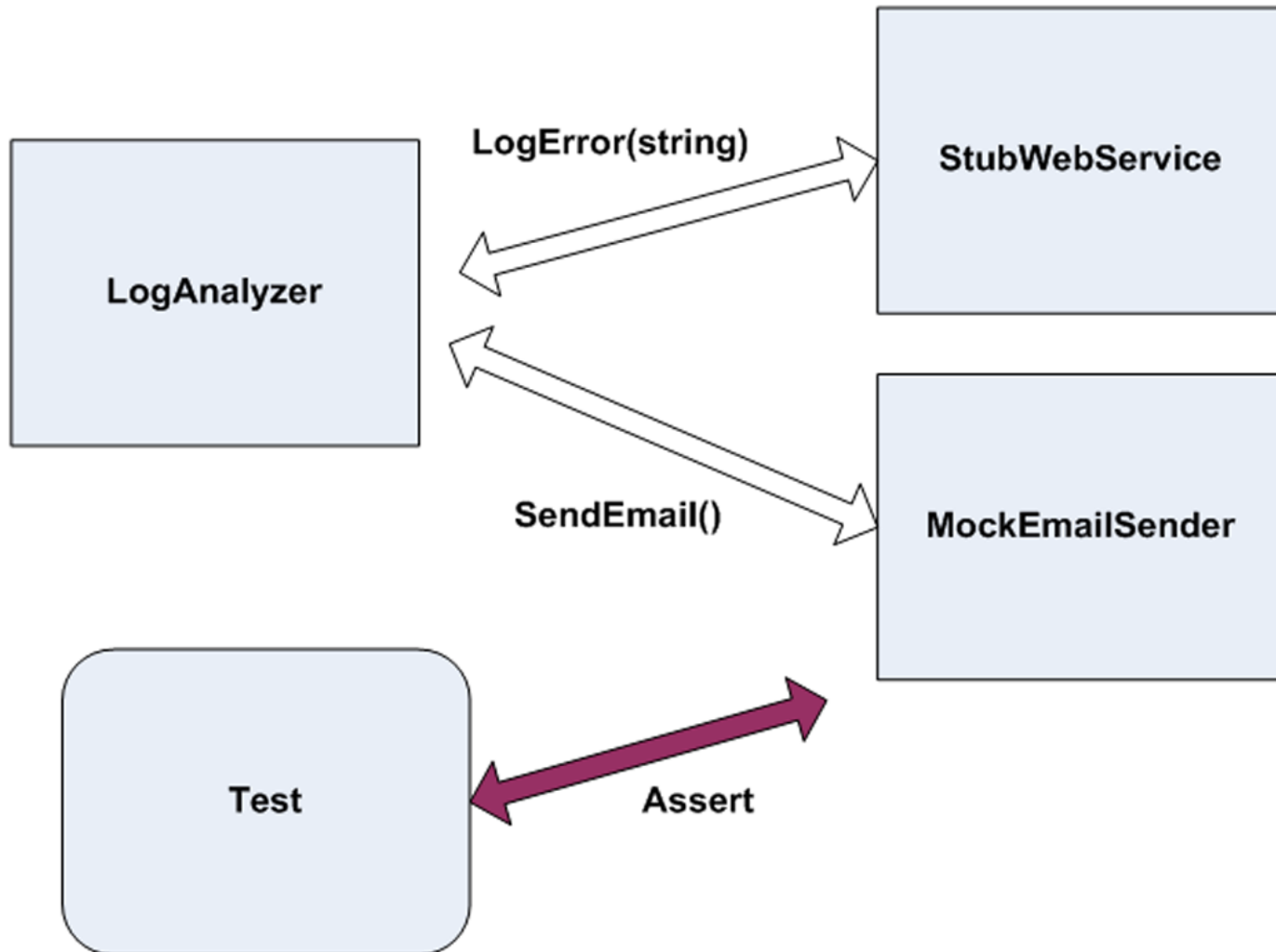
Разница между моками и стабами



Разница между моками и стабами



Одновременно и мок, и стаб



Пример из [R.Osherove]

```
// LogAnalyzer logic

if (fileName.Length < 8)
{
    try
    {
        service.LogError("Filename too short:" + fileName);
    }
    catch (Exception e)
    {
        email.SendEmail("a", "subject", e.Message);
    }
}
```

Пример из [R.Osherove]

```
[Test]
public void Analyze_WebServiceThrows_SendsEmail()
{
    FakeWebService stubService = new FakeWebService();
    stubService.ToThrow = new Exception("fake exception");

    FakeEmailService mockEmail = new FakeEmailService();

    LogAnalyzer2 log = new LogAnalyzer2(stubService, mockEmail);
    string tooShortFileName = "abc.ext";

    log.Analyze(tooShortFileName);

    StringAssert.Contains("someone@somewhere.com", mockEmail.To);
    StringAssert.Contains("fake exception", mockEmail.Body);
    StringAssert.Contains("can't log", mockEmail.Subject);
}
```

И опять к нашим баранам...

```
class FakeRoots : IRoots
{
    public bool SolvedState { get; private set; }

    public double[] Calculate(double[] p)
    {
        SolvedState = true;

        return new double[] { -3, -2, -1, 1, 2 };
    }
}
```

Данный баран несколько надуман, но и сам Рой говорит, что ассерт на заглушке нужно делать в последнюю очередь, когда это действительно нужно потестить...

```
[Test]
public void TestPolynomialRootsGetCalled()
{
    FakeRoots rootsEvaluator = new FakeRoots();
    EquationSolver solver = new EquationSolver(rootsEvaluator);

    double[] roots = solver.Solve(new double[] { 1, 3, -5, -15, 4, 12 });

    Assert.That(rootsEvaluator.SolvedState, Is.EqualTo(true));
}
```


Isolation Frameworks

Есть фреймворки, которые берут на себя задачу создания конкретных объектов заглушек. Т.е. нам достаточно объявить только сам интерфейс и указать специальным образом фреймворку, чтобы он создал конкретную заглушку по этому интерфейсу



Moq



NSubstitute

FakeItEasy

Microsoft Fakes





NMock

Install Moq via NuGet

NuGet: EquationSolvingMockTestProject ➦ ✕ EquationSolvingTestClass.cs

[Browse](#) Installed Updates **1**

Moq ✕ ↺ ☐ Include prerelease

	Moq by Daniel Cazzulino, kzu, 20M downloads Moq is the most popular and friendly mocking framework for .NET	✓ v4.8.0
	Moq.Contrib by kzu.net, 84,2K downloads Contributed features and third-party integration for Moq.	v0.3.0
	Grace.Moq by Ian Johnson, 10,1K downloads Grace.Moq is a companion library for Grace and Moq	v2.4.2
	this.log-moq by RealDimensions Software, LLC, 1,56K downloads this.Log-Moq - this.Log logging extension using Moq	v0.0.3

Пример использования Moq для мока

```
private readonly Mock<IRoots> _mockRoots = new Mock<IRoots>();

[Test]
public void TestPolynomialRootsGetCalled()
{
    var solver = new EquationSolver(_mockRoots.Object);

    double[] coeffs = { 1, 3, -5, -15, 4, 12 };
    double[] roots = solver.Solve(coeffs);

    _mockRoots.Verify(r => r.Calculate(coeffs));

    // more checks:

    // _mockRoots.Verify(r => r.Calculate(coeffs), Times.Once());
    // _mockRoots.VerifyNoOtherCalls();
}
```

Read more: <https://github.com/Moq/moq4/wiki/Quickstart>

Пример использования Moq для стаба

```
private readonly Mock<IRoots> _mockRoots = new Mock<IRoots>();

[Test]
public void TestMoreThanThreeCoefficients()
{
    double[] coeffs = { 1, 3, -5, -15, 4, 12 };
    double[] validRoots = { -3, -2, -1, 1, 2 };

    _mockRoots.Setup(r => r.Calculate(coeffs)).Returns(validRoots);

    var solver = new EquationSolver(_mockRoots.Object);

    double[] roots = solver.Solve(coeffs);





    Assert.That(roots, Is.EquivalentTo(validRoots));
}
```

Install NSubstitute via NuGet

NuGet: EquationSolvingMockTestProject ▢ ✕

[Browse](#) Installed Updates **1**

NSubstitute ✕ ↻ ☐ Include prerelease

	NSubstitute by Anthony Egerton,David Tchepak,Alexandr Nikitin, 3,77M downloads ✓ v3.1.0
NSubstitute is a friendly substitute for .NET mocking frameworks. It has a simple, succinct syntax to help developers write clearer tests. NSubstitute is designed for Arrange-Act-Assert (AAA) testing an...	
	Grace.NSubstitute by Ian Johnson, 9,76K downloads v2.4.2
Grace.NSubstitute is a companion library for Grace and NSubstitute	
	NSubstitute.Construct by Elia Franke, 2,79K downloads v1.0.0.13
Extends NSubstitute with capability to automatically construct instances of an object, with optional automatic mapping of injected parameters.	
	NSubstitute.Verify.That by Ben Arnold, 72 downloads v1.0.0
Glue to allow NSubstitute to use Fluent Assertions when verifying received calls.	

Пример использования NSubstitute для мока

```
private readonly IRoots _mockRoots = Substitute.For<IRoots>();

[Test]
public void TestPolynomialRootsGetCalled()
{
    var solver = new EquationSolver(_mockRoots);

    double[] coeffs = { 1, 3, -5, -15, 4, 12 };
    double[] roots = solver.Solve(coeffs);

    _mockRoots.Received().Calculate(coeffs);

    // equivalent to:
    // _mockRoots.Received(1).Calculate(coeffs);
}
```

Read more: <http://nsubstitute.github.io/help.html>

Пример использования NSubstitute для стаба

```
private readonly IRoots _mockRoots = Substitute.For<IRoots>();

[Test]
public void TestMoreThanThreeCoefficients()
{
    double[] coeffs = { 1, 3, -5, -15, 4, 12 };
    double[] validRoots = { -3, -2, -1, 1, 2 };

    _mockRoots.Calculate(coeffs).Returns(validRoots);

    var solver = new EquationSolver(_mockRoots);

    double[] roots = solver.Solve(coeffs);

    Assert.That(roots, Is.EquivalentTo(validRoots));
}
```