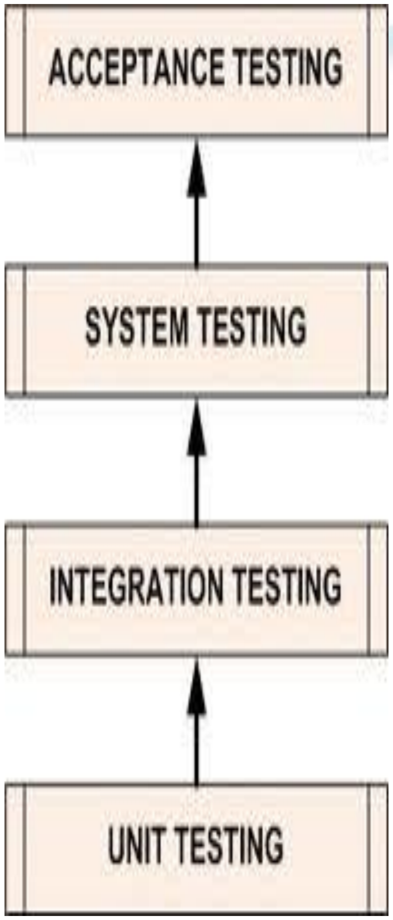
Unit Testing Execution

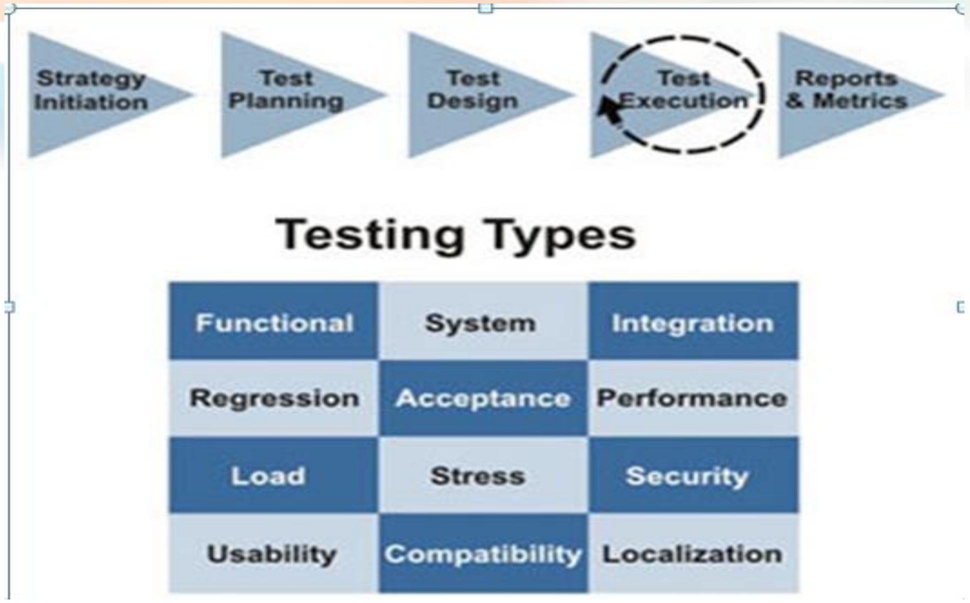
Instructor << >>

Agenda

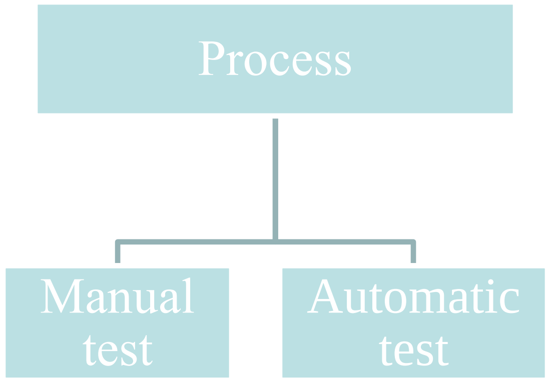
- Software testing levels
- Manual unit testing
- Unit Testing based on UT cases
- Automated Unit Testing
- Automated Unit Testing with NUnit
- Automated Tests vs. Manual Tests
- Best Practices

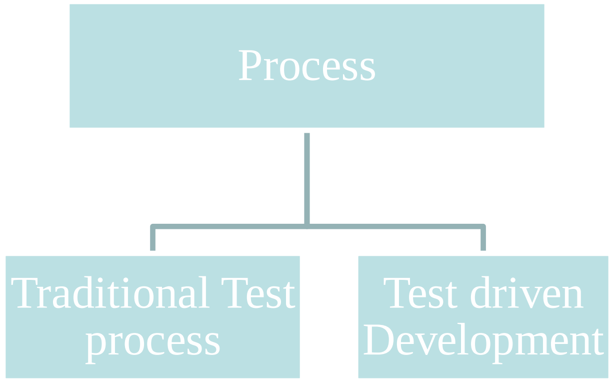
Too many of Software Testing Levels





Functional	System	Integration
Regression	Acceptance	Performance
Load	Stress	Security
Usability	Compatibility	Localization





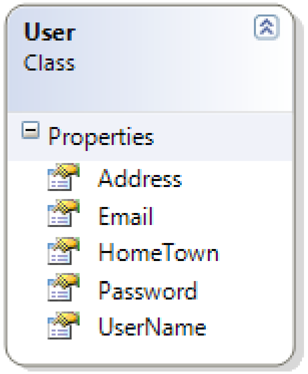
How we test this function?

Requirement :

- Write a module to add an User to DataBase

Business rule :

- Email can not be duplicated
- Email must be in valid form
- UserName ‘s length must be > 8
- UserName can not be dupplicated
- Password length must be > 8



Manual Unit Testing

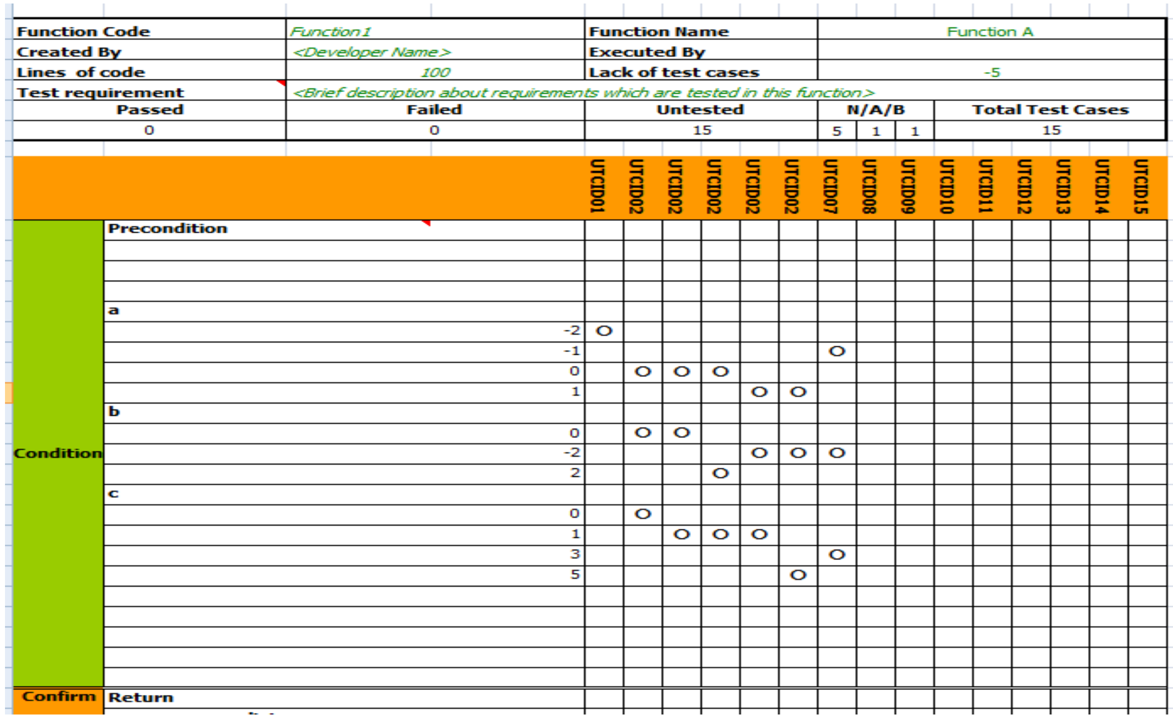
- Write code
- Uploading the code to some place
- Build it
- Running the code manually (in many cases filling up forms etc step by step)
- Check Log files, Database, External Services, Values of variable names, Output on the screen etc
- If it does not work, repeat the above process

Manual Unit Testing - Limitation

- Developer nhớ được trường hợp nào thì test trường hợp đó
- Đến cuối dự án số lượng test case càng lúc càng nhiều, khả năng cover của lập trình viên giảm xuống!
- Nhiều test case bị trùng lắp
- Nhiều test case bị lack
- Team lead không thể review hết được
- ➔ Kết quả dự án chỉ trông chờ vào tester!!!!
- ➔ Rất nhiều lỗi phát sinh sau khi system test, đa phần các lỗi xuất phát do Dev test không kỹ từ lúc Unit Test!

Unit Testing based on UT cases

- Để giải quyết vấn đề trên Mỗi khi developer test xong phải viết tài liệu mô tả test case trên word hoặc excel !
- Điều này giúp team rất dễ dàng review.. Tuy nhiên, cách làm này sẽ phát sinh ra rất nhiều hạn chế!



Function Code	Function1	Function Name	Function A		
Created By	<Developer Name>	Executed By			
Lines of code	100	Lack of test cases	-5		
Test requirement	<Brief description about requirements which are tested in this function>				
Passed	Failed	Untested	N/A/B		Total Test Cases
0	0	15	5	1	1	15

			UTCID01	UTCID02	UTCID02	UTCID02	UTCID02	UTCID02	UTCID07	UTCID08	UTCID09	UTCID10	UTCID11	UTCID12	UTCID13	UTCID14	UTCID15
	Precondition																
	a																
		-2	O														
		-1							O								
		0		O	O	O											
		1					O	O									
	b																
		0		O	O												
Condition		-2					O	O	O								
		2				O											
	c																
		0		O													
		1			O	O	O										
		3							O								
		5						O									
Confirm	Return																

Unit Testing based on UT cases

- Các dự án lớn thì số lượng tài liệu test case thường cũng rất lớn!
- Các dự án lớn thì requirement thường hay thay đổi
- Mỗi khi requirement thay đổi➔ Phải sửa code➔ phải cập nhật lại tài liệu testcase ➔và lại manual retest , rất tốn effort➔Càng đến cuối dự án, lượng việc sinh ra càng nhiều , viết test case document trở thành “địa ngục “ thực sự !➔ dev không còn đủ effort update test case document, tài liệu nhanh chóng bị lạc hậu, hoặc việc update chỉ là đối phó!
- Một số trường hợp không thể dùng Excel Unit TestCase

So, what is the solution?

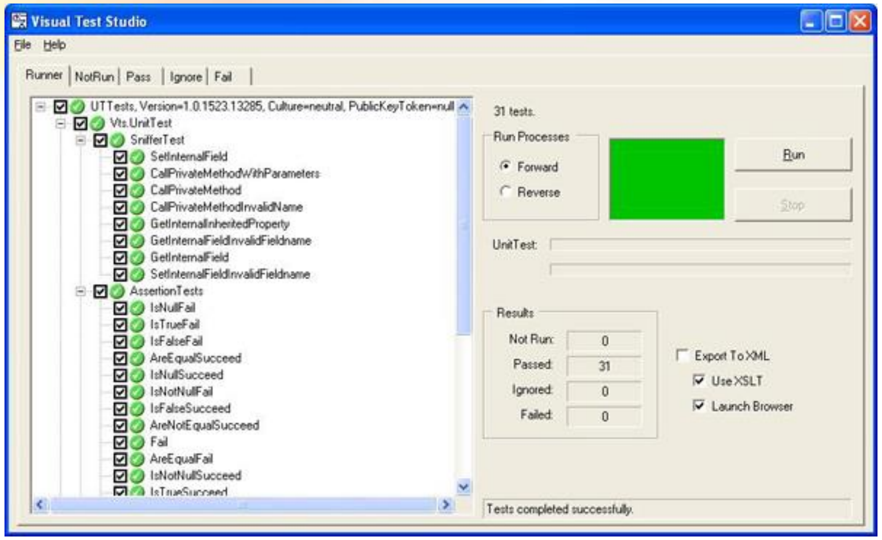
Automated Unit Testing
First Step

- Coding Process with Automated Unit Tests
  - Write code
  - Write one or more test cases script
  - Auto-compile and run
  - If tests fail -> make appropriate modifications
  - If tests pass -> repeat for next method

Automated Unit Testing

Common Tools

- UT Tools for references:
  - Java: JUnit, J2MEUnit
  - C/C++: cppUnit
  - Python: pyUnit
  - Perl: PerlUnit
  - Visual Basic: vbUnit
  - C# .NET: Nunit,csUnit



```
UTTests, Version=1.0.1523.13285, Culture=neutral, PublicKeyToken=null
  Vts.UnitTest
    SnifferTest
      SetInternalField
      CallPrivateMethodWithParameters
      CallPrivateMethod
      CallPrivateMethodInvalidName
      GetInternalInheritedProperty
      GetInternalFieldInvalidFieldname
      GetInternalField
      SetInternalFieldInvalidFieldname
    AssertionTests
      IsNullFail
      IsTrueFail
      IsFalseFail
      AreEqualSucceed
      IsNullSucceed
      IsNotNullFail
      IsFalseSucceed
      AreNotEqualSucceed
      Fail
      AreEqualFail
      IsNotNullSucceed
      IsTrueSucceed
```

31 tests.

Run Processes: [✓ Forward] Reverse

UnitTest:

Results

Not Run:	0
Passed:	31
Ignored:	0
Failed:	0

Export To XML; [✓ Use XSLT]; [✓ Launch Browser]

Tests completed successfully.

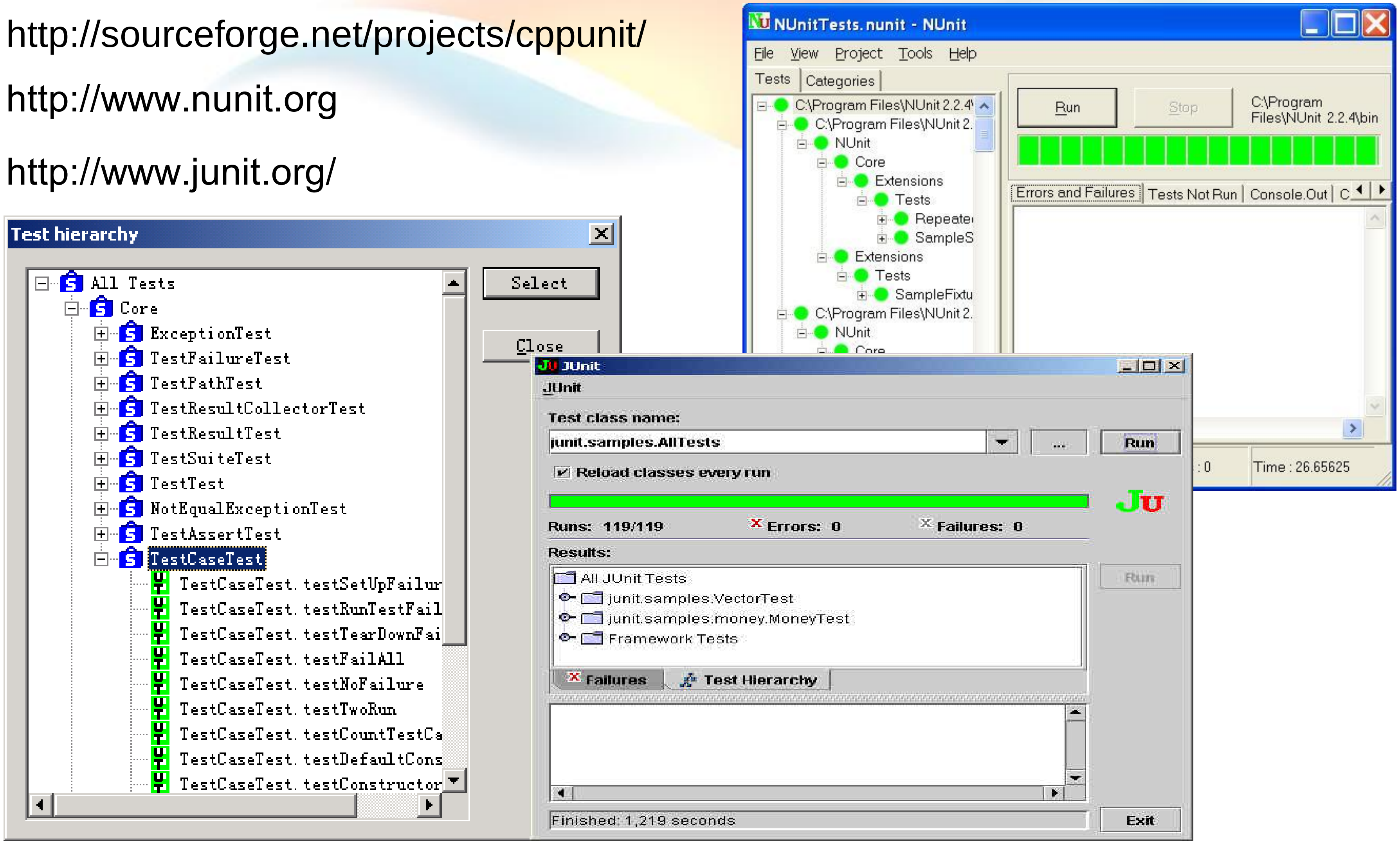
- Refferences:
  - ★ http://www.testingfaqs.org/t-unit.html
  - ★ www.junit.org
  - ★ http://www.codeproject.com/gen/design/autp5.asp

Automated Unit Testing
Common Tools

http://sourceforge.net/projects/cppunit/

http://www.nunit.org

http://www.junit.org/



Test hierarchy

```
All Tests
  Core
    ExceptionTest
    TestFailureTest
    TestPathTest
    TestResultCollectorTest
    TestResultTest
    TestSuiteTest
    TestTest
    NotEqualExceptionTest
    TestAssertTest
    TestCaseTest
      TestCaseTest.testSetUpFailur
      TestCaseTest.testRunTestFail
      TestCaseTest.testTearDownFai
      TestCaseTest.testFailAll
      TestCaseTest.testNoFailure
      TestCaseTest.testTwoRun
      TestCaseTest.testCountTestCa
      TestCaseTest.testDefaultCons
      TestCaseTest.testConstructor
```

NUnitTests.nunit - NUnit

```
C:\Program Files\NUnit 2.2.4\
  C:\Program Files\NUnit 2.
    NUnit
      Core
        Extensions
          Tests
            Repeate
            SampleS
      Extensions
        Tests
          SampleFixtu
  C:\Program Files\NUnit 2.
    NUnit
      Core
```

C:\Program Files\NUnit 2.2.4\bin

Time : 26.65625

JUnit

Test class name: junit.samples.AllTests

☑ Reload classes every run

Runs: 119/119 Errors: 0 Failures: 0

Results:

```
All JUnit Tests
  junit.samples.VectorTest
  junit.samples.money.MoneyTest
  Framework Tests
```

Finished: 1,219 seconds

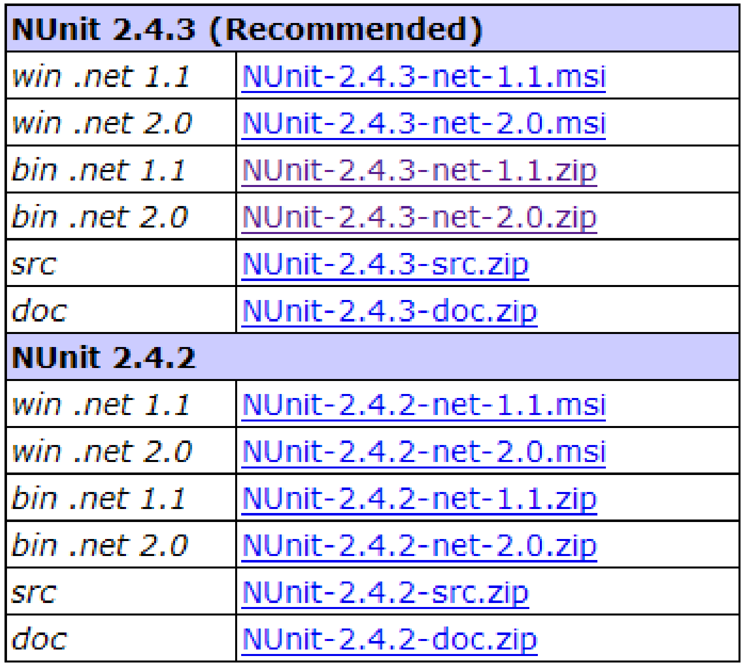
Automated Unit Testing Demo

Automated Unit Testing with NUnit
What is NUnit?

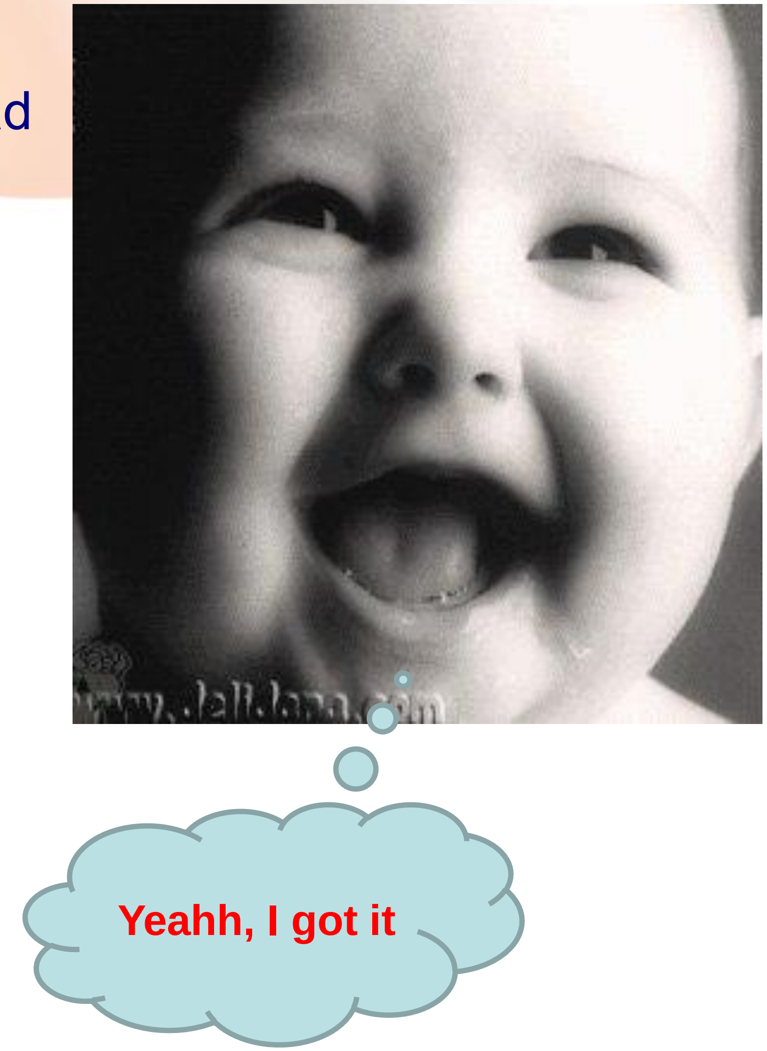
- NUnit – an open source test tool for .NET
- Useful for development and regression
- Leads to a design-for-test approach
- Tests can be written in VB.NET or C#

Automated Unit Testing with NUnit
Where to get NUnit?

- Let's go to website: http://www.nunit.org/index.php?p=download

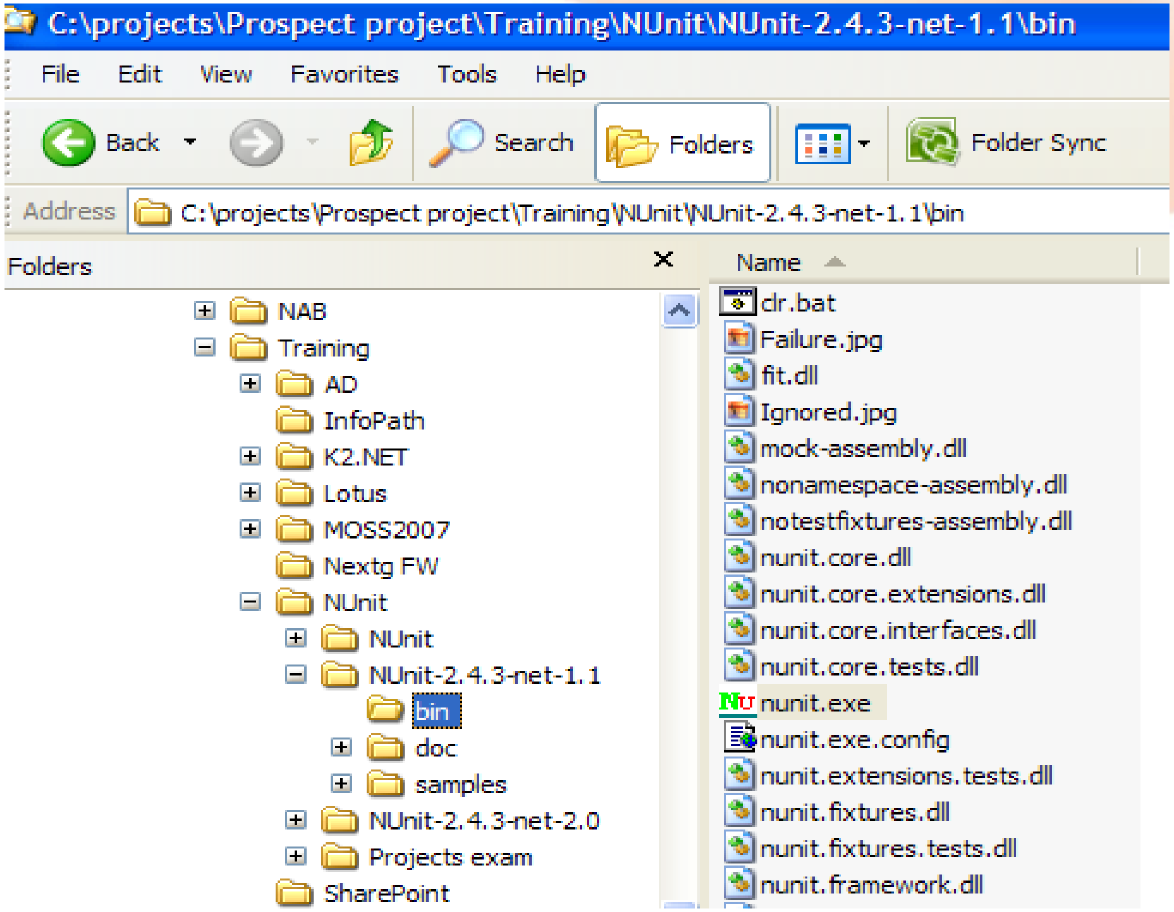


NUnit 2.4.3 (Recommended)	
win .net 1.1	NUnit-2.4.3-net-1.1.msi
win .net 2.0	NUnit-2.4.3-net-2.0.msi
bin .net 1.1	NUnit-2.4.3-net-1.1.zip
bin .net 2.0	NUnit-2.4.3-net-2.0.zip
src	NUnit-2.4.3-src.zip
doc	NUnit-2.4.3-doc.zip
NUnit 2.4.2	
win .net 1.1	NUnit-2.4.2-net-1.1.msi
win .net 2.0	NUnit-2.4.2-net-2.0.msi
bin .net 1.1	NUnit-2.4.2-net-1.1.zip
bin .net 2.0	NUnit-2.4.2-net-2.0.zip
src	NUnit-2.4.2-src.zip
doc	NUnit-2.4.2-doc.zip



Automated Unit Testing with NUnit

Where to get Nunit? - Extract to any folder

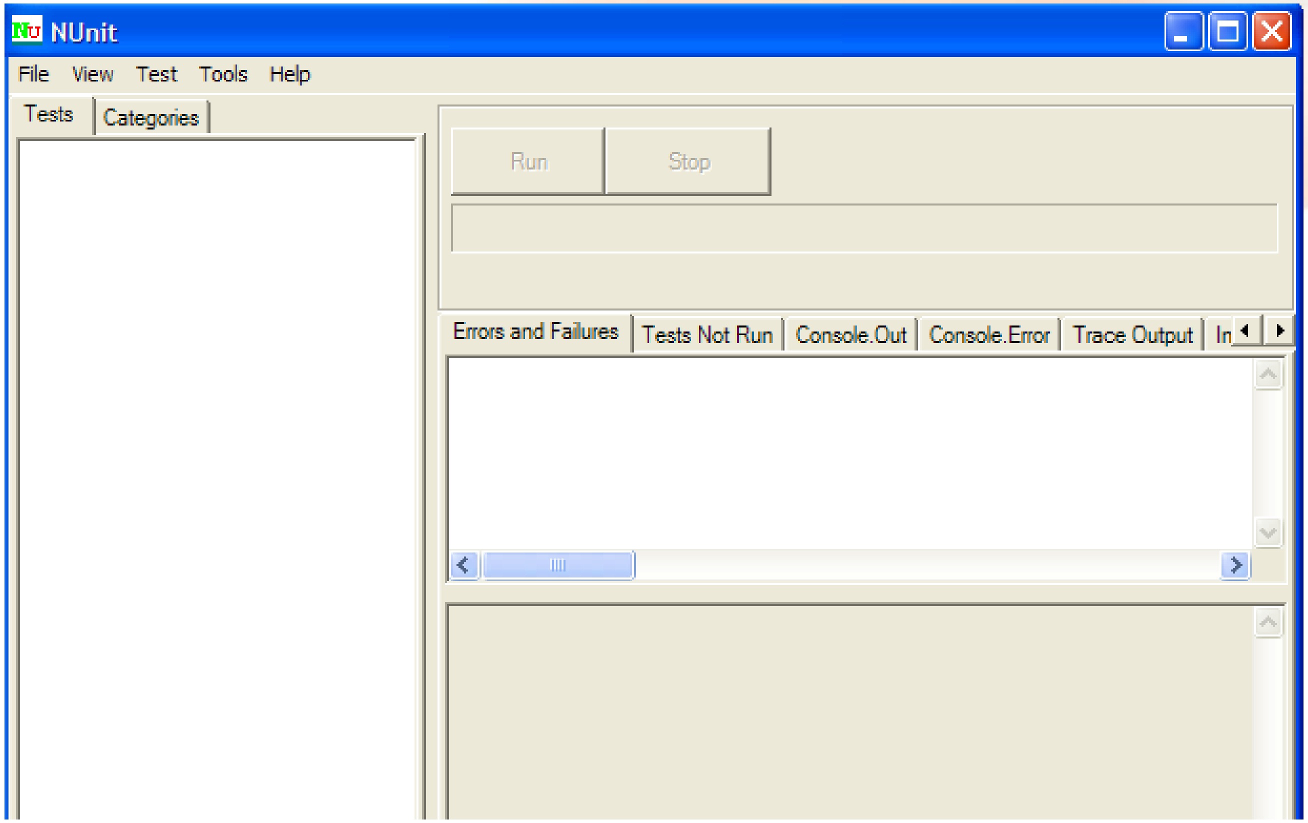

C:\projects\Prospect project\Training\NUnit\NUnit-2.4.3-net-1.1\bin
File Edit View Favorites Tools Help
Back
Search
Folders
Folder Sync
Address
Folders
Name

C:\projects\Prospect project\Training\NUnit\NUnit-2.4.3-net-1.1\bin

- NAB
- Training
  - AD
  - InfoPath
  - K2.NET
  - Lotus
  - MOSS2007
  - Nextg FW
  - NUnit
    - NUnit
    - NUnit-2.4.3-net-1.1
      - bin
      - doc
      - samples
    - NUnit-2.4.3-net-2.0
    - Projects exam
  - SharePoint

```
clr.bat
Failure.jpg
fit.dll
Ignored.jpg
mock-assembly.dll
nonamespace-assembly.dll
notestfixtures-assembly.dll
nunit.core.dll
nunit.core.extensions.dll
nunit.core.interfaces.dll
nunit.core.tests.dll
nunit.exe
nunit.exe.config
nunit.extensions.tests.dll
nunit.fixtures.dll
nunit.fixtures.tests.dll
nunit.framework.dll
```

Automated Unit Testing with Nunit
Screens of tool


NUnit
File View Test Tools Help
Tests Categories
Run Stop
Errors and Failures | Tests Not Run | Console.Out | Console.Error | Trace Output | In

Automated Unit Testing with Nunit
How to use NUnit?

- Create a test case base on NUnit framework
- Deploy and Run

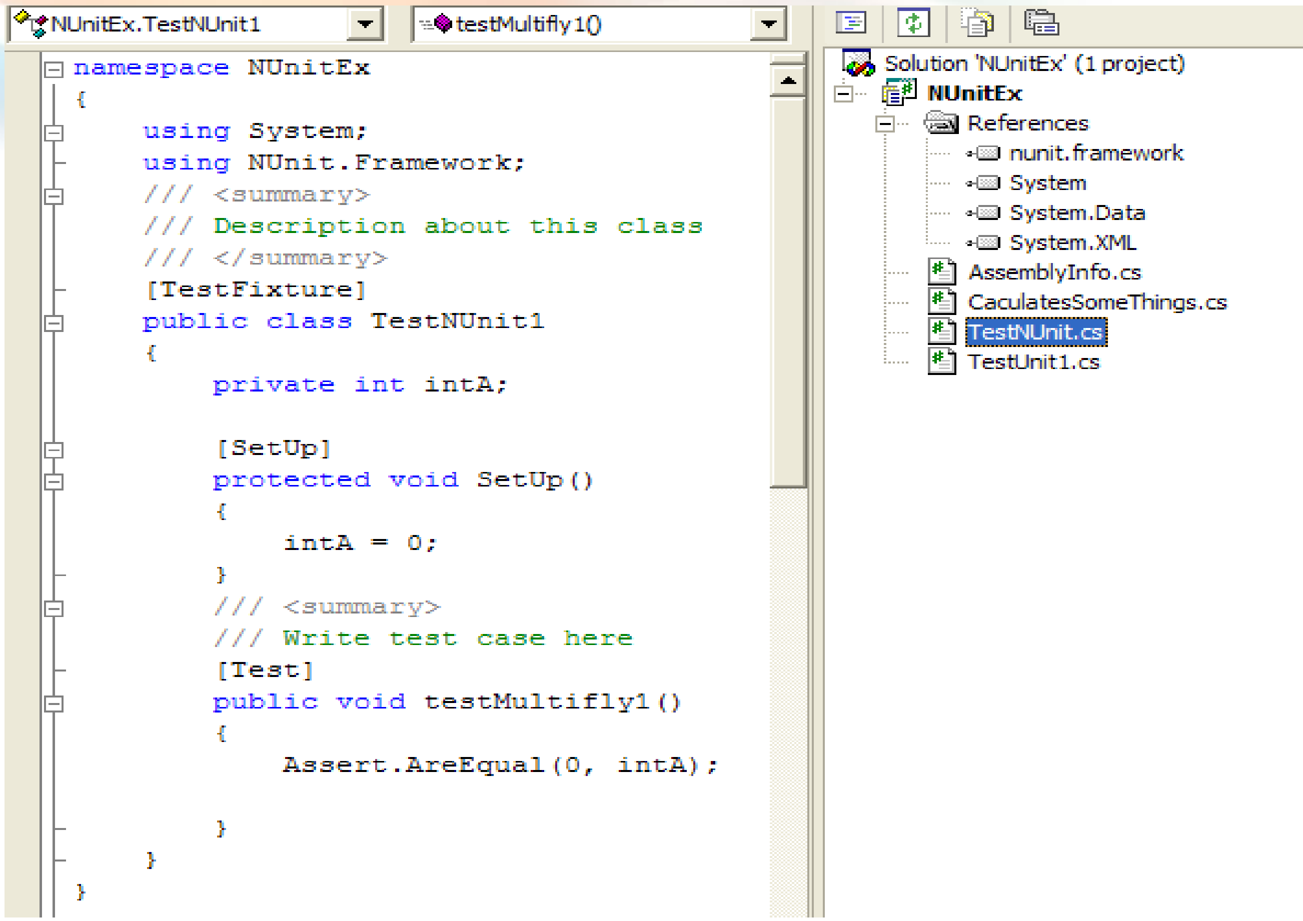
Automated Unit Testing with Nunit
Create a test case

- Step 1: Create a Class
- Step 2: Add a reference nunit.Framework.dll to this class
- Step 3: Add a reference to *.dll contains function which you want to do Unit test
- Step 4: Restructure class following Nunit frame work
- Step 5: Write a test case

Let’s me go…

Automated Unit Testing with Nunit
Create a test case

- Step 1: Create a Class
- Step 2: Add a reference nunit.Framework.dll to this class
- Step 3: Add a reference to *.dll contains function which you want to do Unit test
- Step 4: Restructure class following Nunit frame work



```
namespace NUnitEx
{
    using System;
    using NUnit.Framework;
    /// <summary>
    /// Description about this class
    /// </summary>
    [TestFixture]
    public class TestNUnit1
    {
        private int intA;

        [SetUp]
        protected void SetUp()
        {
            intA = 0;
        }
        /// <summary>
        /// Write test case here
        [Test]
        public void testMultifly1()
        {
            Assert.AreEqual(0, intA);

        }
    }
}
```

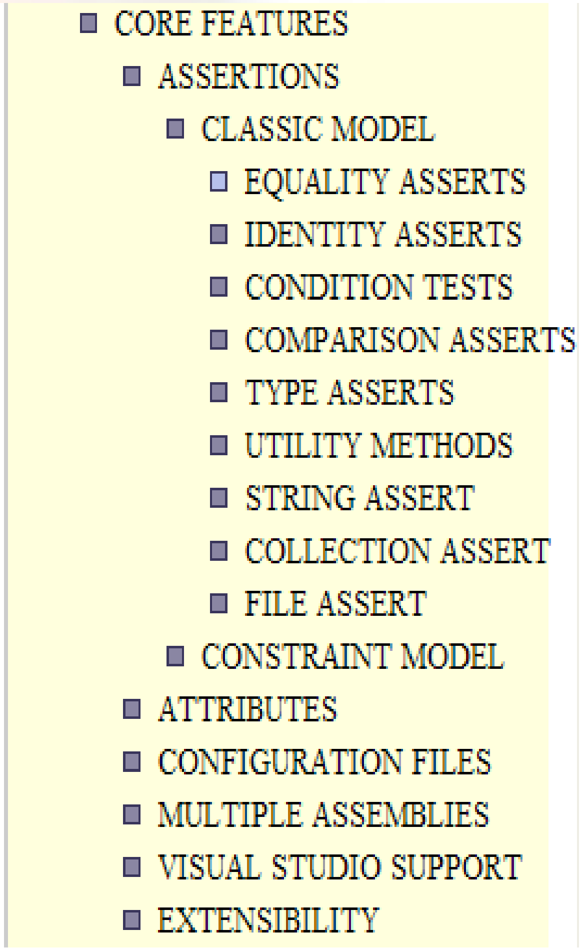
Automated Unit Testing with Nunit
Create a test case

- Step 5: Write a test case
  - Each test case will be a function/method of class
  - Must have attribute [Test] above a function/method
  - Ex:

```
[Test]
public void testCase1()
{
            Assert.AreEqual(0, intA);
}

[Test]
public void testCase2()
{
            Assert.AreEqual(0, divides(intA, intB));
}
```

Automated Unit Testing with Nunit
Core Features

- Core Features to code a test case
  - **Assertions**
    - Equality Asssserts:
      - Ex: Assert.AreEqual(int expected, int actual);
    - Condition Tests:
      - Ex: Assert.IsTrue(bool condition);
    - Comparrison Asserts
      - Ex: Assert.Greater(int arg1, int arg2);
    - Type Asserts
      - Ex: Assert.IsInstanceOfType(Type expected, object actual);
    - Utility methods
      - Ex: Assert.Fail();
    - String Assert
      - Ex: StringAssert.Contains(string expected, string actual);
    - Collection Asserts
      - Ex: CollectionAssert.AreEqual(Collection expected, Collection actual);
  - **Attributes**

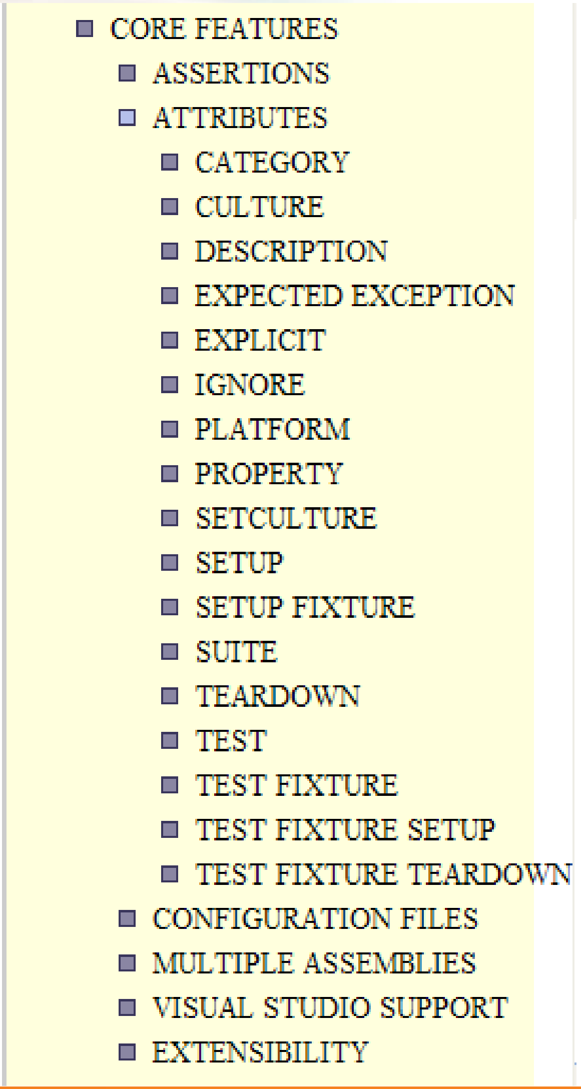


- CORE FEATURES
  - ASSERTIONS
    - CLASSIC MODEL
      - EQUALITY ASSERTS
      - IDENTITY ASSERTS
      - CONDITION TESTS
      - COMPARISON ASSERTS
      - TYPE ASSERTS
      - UTILITY METHODS
      - STRING ASSERT
      - COLLECTION ASSERT
      - FILE ASSERT
    - CONSTRAINT MODEL
  - ATTRIBUTES
  - CONFIGURATION FILES
  - MULTIPLE ASSEMBLIES
  - VISUAL STUDIO SUPPORT
  - EXTENSIBILITY

Automated Unit Testing with Nunit Core Features

★ **Attributes**

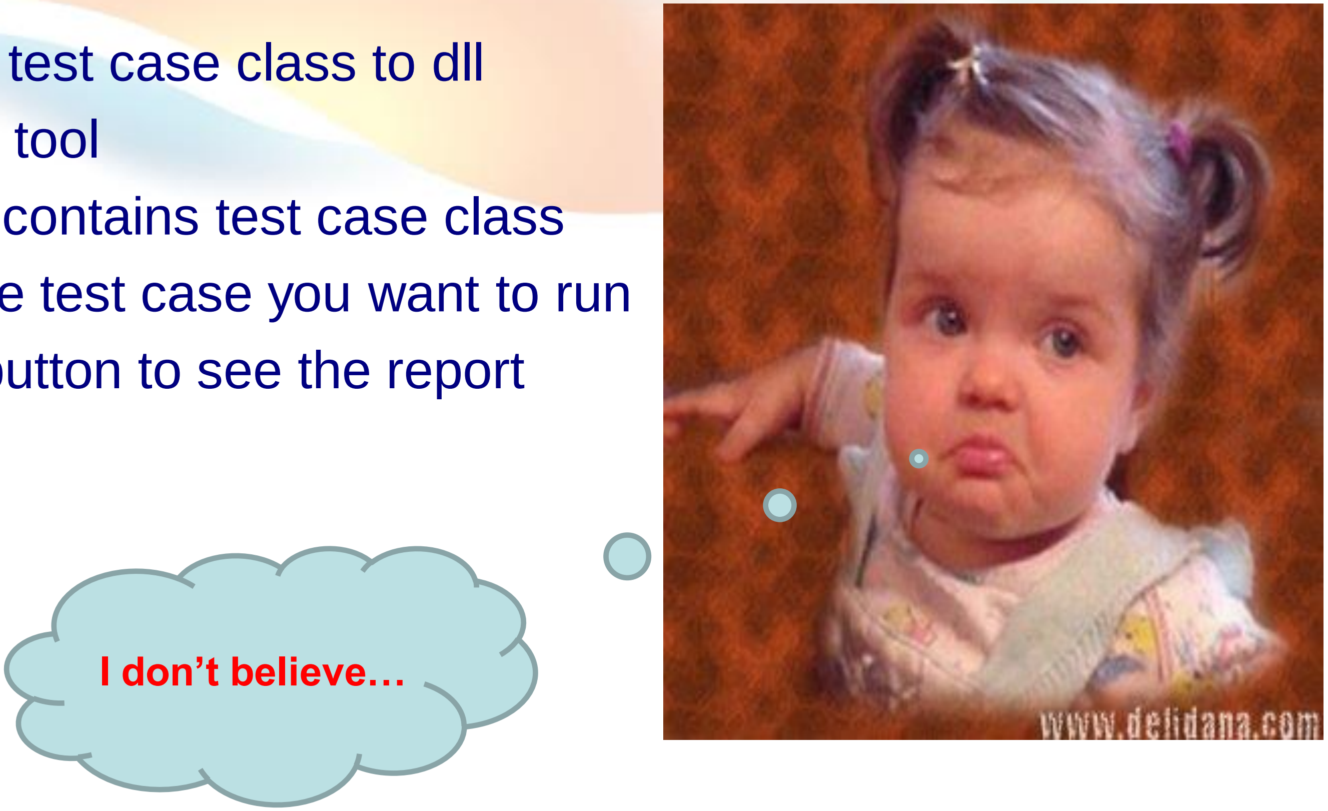
```
[TestFixture]
[Category("TestUnitExample")]
    public class TestNUnit
    {
            private int intA;
            private int intB;
            private CaculatesSomeThings objCal;

            [SetUp]
            protected void SetUp()
            {
                    intA = 0;
                    intB = 0;
                    objCal = new CaculatesSomeThings();
            }

            [Test]
            Public void TestCase1()
            {
                    Assert.AreEqual(0, objCal.Multifly(intA, intB))
            }
```



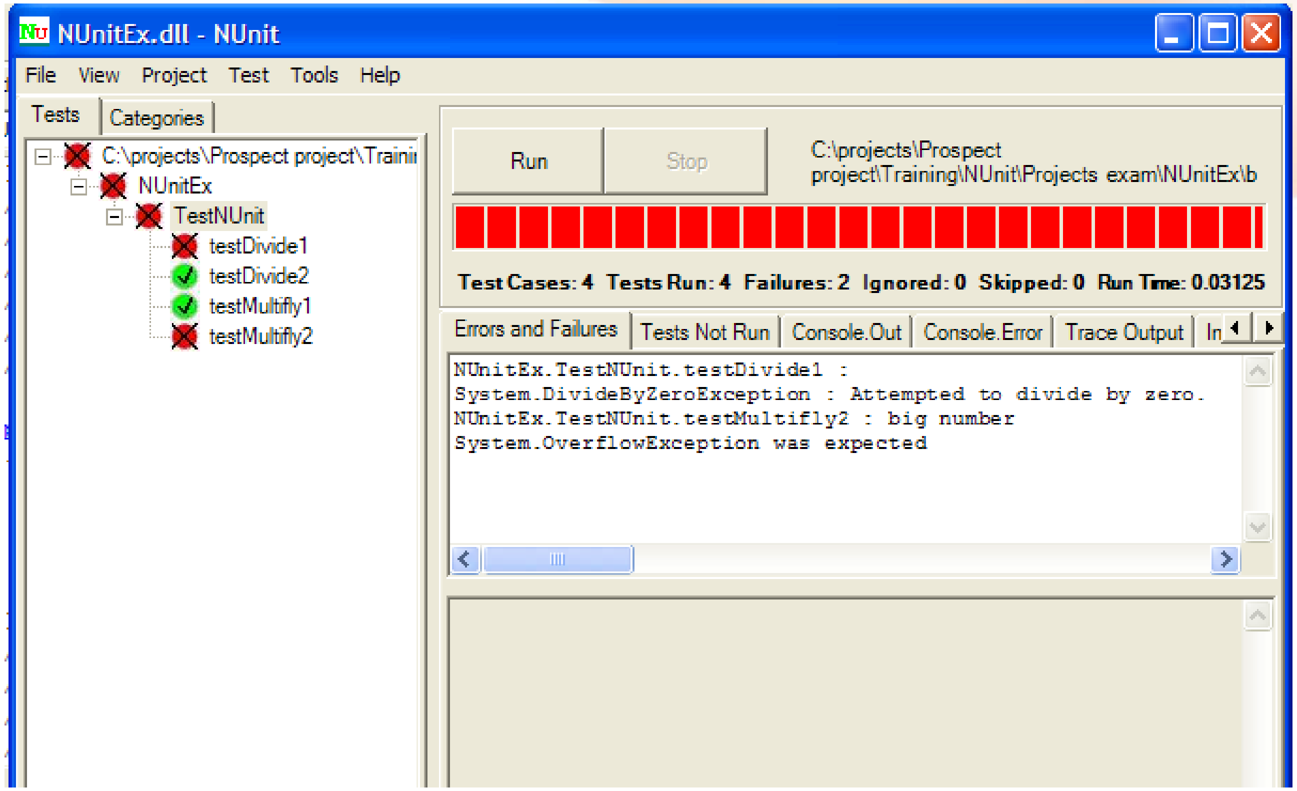
- CORE FEATURES
  - ASSERTIONS
  - ATTRIBUTES
    - CATEGORY
    - CULTURE
    - DESCRIPTION
    - EXPECTED EXCEPTION
    - EXPLICIT
    - IGNORE
    - PLATFORM
    - PROPERTY
    - SETCULTURE
    - SETUP
    - SETUP FIXTURE
    - SUITE
    - TEARDOWN
    - TEST
    - TEST FIXTURE
    - TEST FIXTURE SETUP
    - TEST FIXTURE TEARDOWN
  - CONFIGURATION FILES
  - MULTIPLE ASSEMBLIES
  - VISUAL STUDIO SUPPORT
  - EXTENSIBILITY

Automated Unit Testing with Nunit Deploy and Run

- Step 1: Compile a test case class to dll
- Step 2: Run NUnit tool
- Step 3: Open *.dll contains test case class
- Step 4: Choose the test case you want to run
- Step 5: Click run button to see the report



Automated Unit Testing with Nunit
Deploy and Run



C:\projects\Prospect project\Training\NUnit\Projects exam\NUnitEx\b

Test Cases: 4 Tests Run: 4 Failures: 2 Ignored: 0 Skipped: 0 Run Time: 0.03125

```
NUnitEx.TestNUnit.testDivide1 :
System.DivideByZeroException : Attempted to divide by zero.
NUnitEx.TestNUnit.testMultiply2 : big number
System.OverflowException was expected
```

Q&A