Testing & QA

Functional Testing Types

- **Unit tests:** Tests an individual unit of the software to make sure it performs appropriately.
- **Integration tests:** Takes multiple individual units of the software and tests them as a group to make sure they interact appropriately.
- **Smoke tests:** Tests the major pieces of the software in a non-comprehensive manner to ensure the software works well enough (or is not riddled with too many issues) to move on to additional tests.
- **Regression tests:** Tests that code changes do not have a negative effect on the functionality of the software.
- **User acceptance tests:** Often the last step before software goes live, user acceptance testsmake sure the software meets user needs. End users typically perform these tests during a Beta period.

Unit Testing Frameworks

- MSTest (VS 2012)
- NUnit
- xUnit.net
- NSpec

NUnit

```
namespace NUnit.Tests
{
  using System;
  using NUnit.Framework;

  [TestFixture]
  public class SuccessTests
  {
    [Test] public void Add()
    { /* ... */ }

    public void TestSubtract()
    { /* backwards compatibility */ }
  }
}
```

SetUp & TearDown

```
namespace NUnit.Tests
{
  using System;
  using NUnit.Framework;

  [TestFixture]
  public class SuccessTests
  {
    [SetUp] public void Init()
    { /* ... */ }

    [TearDown] public void Cleanup()
    { /* ... */ }

    [Test] public void Add()
    { /* ... */ }
  }
}
```

Assert

- `AreEqual(object expected, object actual)`
- `AreEqual<T>(T expected, T actual)`
- `AreNotEqual(object expected, object actual)`
- `AreNotEqual<T>(T expected, T actual)`
- `AreSame(object expected, object actual)`
- `AreNotSame(object expected, object actual)`:
- `Equals(object objA, object objB)`: проверяет на равенство оба объекта
- `IsFalse(bool condition)`
- `IsTrue(bool condition)`
- `IsNull(object value)`
- `IsInstanceOfType(object value, Type expectedType)`

Mocking Techniques

- Stubbing
- Mocking
- Fake

Moq - mocking framework

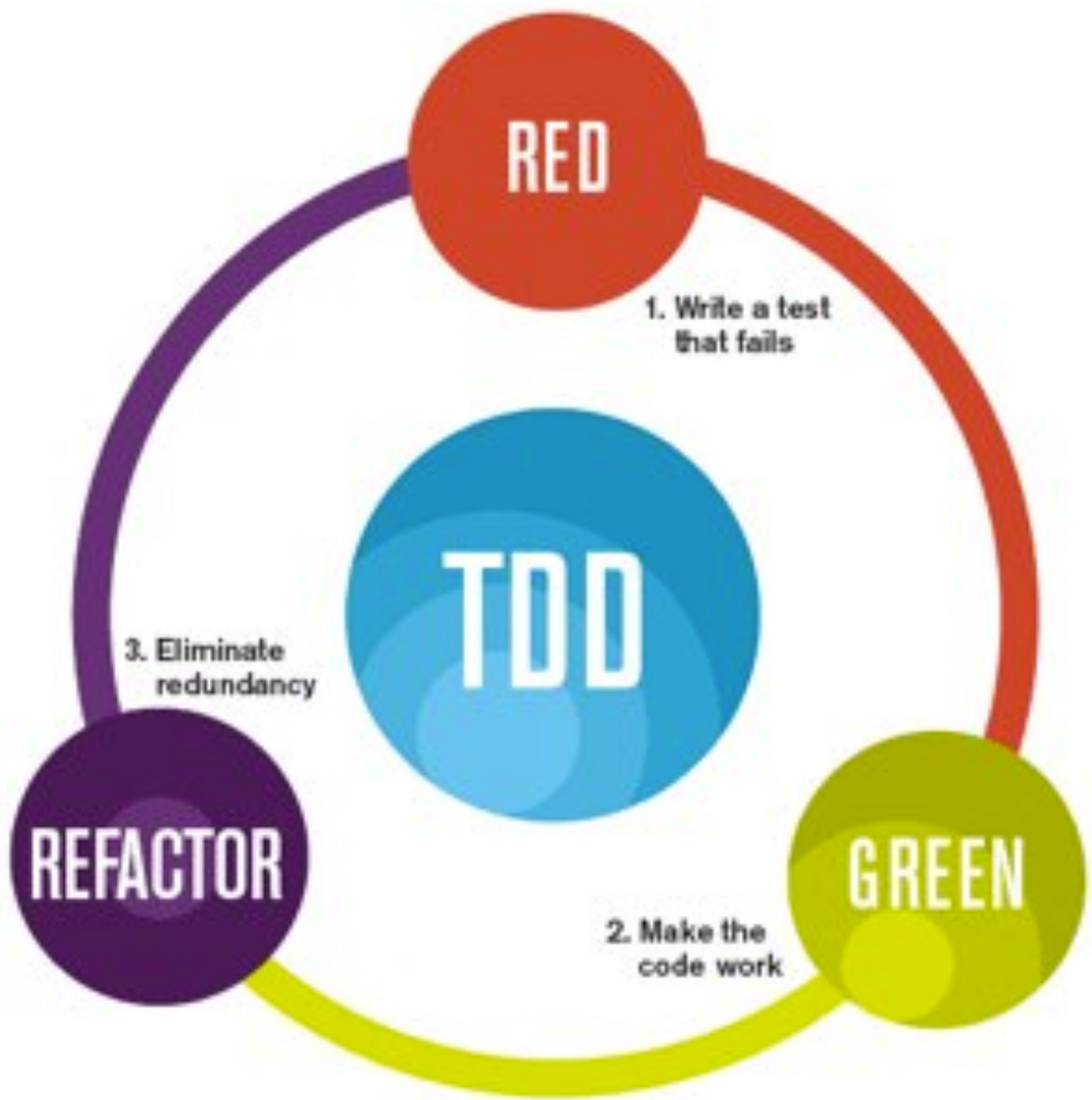
```
public interface IFoo
{
    Bar Bar { get; set; }
    string Name { get; set; }
    int Value { get; set; }
    bool DoSomething(string value);
    bool DoSomething(int number, string value);
    string DoSomethingStringy(string value);
    bool TryParse(string value, out string outputValue);
    bool Submit(ref Bar bar);
    int GetCount();
    bool Add(int value);
}

var mock = new Mock<IFoo>();
```

TDD

Test-driven development (**TDD**) is a software development process that relies on the repetition of a very short development cycle: requirements are turned into very specific test cases, then the software is improved to pass the new tests, only.

TDD



The mantra of Test-Driven Development (TDD) is "red, green, refactor."

Code Coverage

Test coverage is a measure used to describe the degree to which the source code of a program is executed when a particular test suite runs. A program with high test coverage, measured as a percentage, has had more of its source code executed during testing, which suggests it has a lower chance of containing undetected software bugs compared to a program with low test coverage

Code Coverage

- https://www.hanselman.com/blog/AltCoverAndReportGeneratorGiveAmazingCodeCoverageOnNETCore.aspx

- https://www.hanselman.com/blog/AutomaticUnitTestingInNETCorePlusCodeCoverageInVisualStudioCode.aspx