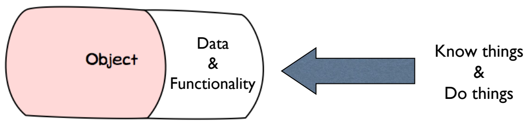
Unit Testing and Test-driven Development

by Andrew Cain and Willem van Straten

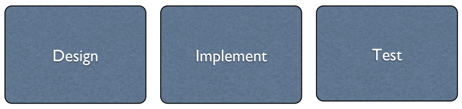
SWINBURNE UNIVERSITY OF TECHNOLOGY

Object Oriented Programming

Object oriented programming involves creating objects that know and do things

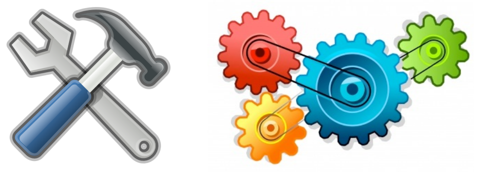


Developers use tools and processes to help guide the creation of programs



Getting programs to work correctly can be tricky at the best of times

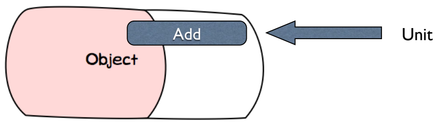
The right tools and processes will help ensure you get a working product



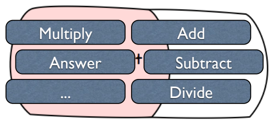
Use Unit Testing tools to help design and build your programs

Verify object functionality with unit tests

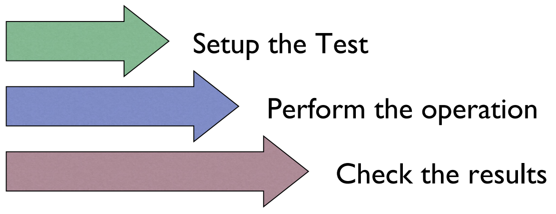
Units represent the smallest testable part of your program



Use many small tests to check as much of the program functionality as possible

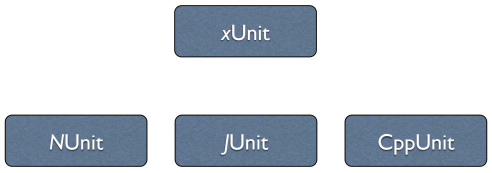


Each test checks if that part of the functionality is working correctly

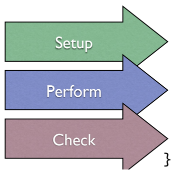


Speed up testing with automated unit testing tools

The *x*Unit framework provides tools to perform unit testing in many languages

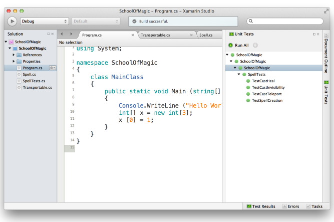


Create test fixtures that contain unit tests



```
[TestFixture()]
public class TestCalc
{
  [Test()]
  public void TestPush ()
  {
    RpnCalculator c = new RpnCalculator();
    int actual;

    c.Push(5);
    actual = c.Answer();

    Assert.AreEqual(5, actual, "Test push 5");
  }
}
```

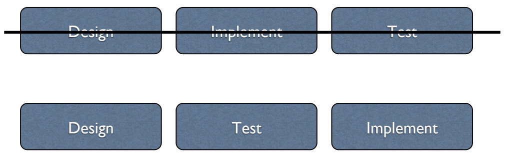
Use the tools to run all of the tests each time you make changes



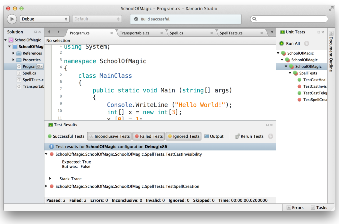
```
using System;

namespace SchoolOfMagic
{
    class MainClass
    {
        public static void Main (string[]
        {
            Console.WriteLine ("Hello Wor
            int[] x = new int[3];
            x [0] = 1;
        }
    }
}
```

Step it up with test driven development

Write the tests first!



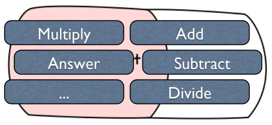
Add features only where the tests fail: create a test, watch it fail, make it work!



```
using System;

namespace SchoolOfMagic
{
    class MainClass
    {
        public static void Main (string[] args)
        {
            Console.WriteLine ("Hello World!");
            int[] x = new int[3];
```

```
Test results for SchoolOfMagic configuration Debug|x86
SchoolOfMagic.SchoolOfMagic.SchoolOfMagic.SpellTests.TestCastInvisibility
    Expected: True
    But was:  False

  Stack Trace
SchoolOfMagic.SchoolOfMagic.SchoolOfMagic.SpellTests.TestSpellCreation

Passed: 2   Failed: 2   Errors: 0   Inconclusive: 0   Invalid: 0   Ignored: 0   Skipped: 0   Time: 00:00:00.0200000
```

Add tests to expand the program’s functionality



Will unit testing help ensure you get the right results?

The right tools and processes can help you get the right results

Use Unit Testing tools to help design and build your programs

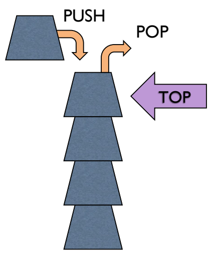
Build the program right with Unit Testing and Test Driven Development

Unit Testing: Postfix Notation Calculator

Postfix Notation: each operator follows its operands

Infix Notation	Postfix Notation
1 + 1	1, 1, +
1 + 2 * 3	1, 2, 3, *, +
1 * 2 + 3	1, 2, *, 3, +

Stack:



Example: 1 + 2 * 3 + 4

Operation	Stack	Action
Push 1	1	1 pushed to top
Push 2	1, 2	2 pushed to top
Push 3	1, 2, 3	3 pushed to top
*	1, 6	Pop 2, Pop 3, Push 2 * 3
+	7	Pop 1, Pop 6, Push 1 + 6
Push 4	7, 4	4 pushed to top
+	11	Pop 7, Pop 4, Push 7 + 4
=		Pop and return 11

This Week's Tasks

Pass Task 5 - Shape Drawer
Pass Task 6 - Unit Testing Shape
Pass Task 7 - Unit Testing the Spells
Pass Task 8 - Documenting the Spell Class