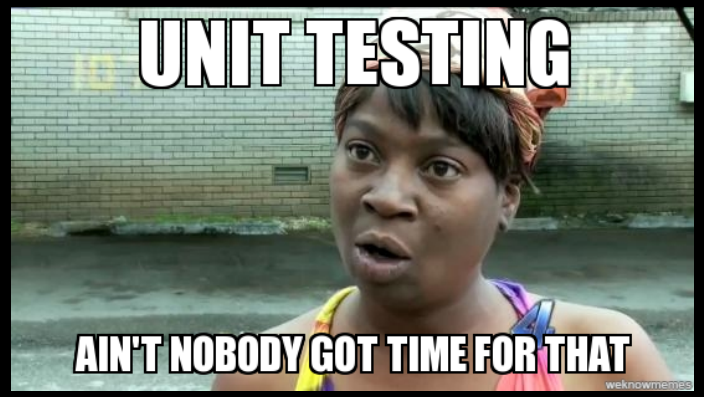

QUICK START TO UNIT TESTING

Object-oriented Programming

Edison Lascano, ESPE

Based on the slides of Dr. Stephen Clyde

PROBLEM: HOW CAN "I" KNOW MY CODE IS CORRECT?

Consider the following class:

```
public class Triangle
  {
     public double[] SideLengths { get; set; }

     public double ComputeArea()
     {
        double halfOfPerimeter = SideLengths.Sum()/2;

        return Math.Sqrt(       halfOfPerimeter *
                                (halfOfPerimeter - SideLengths[0])*
                                (halfOfPerimeter - SideLengths[1])*
                                (halfOfPerimeter - SideLengths[2]));
     }
  }
```

Validation:
Is it doing the right thing?

Verification:
Is it doing that thing right?

SAMPLE TEST CASES

SideLengths = …

{ 3.0, 4.0, 5.0 }
{ 3.432, 4.525, 5.236 }
{ 2.0, 3.0 }
{ 1.0, 1.0, 1.0, 1.0 }
{-1.0, 2.0, 3.0 }

{10, 2, 2}
{2, 2, 2}
{4, 4, 3}
{null, 3, 2}
{2, null, 3}

{A, 2, 3}
{}
null
?? More ??

CONDUCTING A TEST

Three general steps:

Setup:	Ensure the system is in a known state, or at least the component you are testing is in a known state, e.g., create a triangle object with specific sides
Stimulate:	Execute the thing you are testing, e.g., the ComputeArea() method
Observe:	Compare the actual results with the expected results. The results may be the return value of a method, or more generally, the system's new state

MAKING A TEST EXECUTABLE

All three steps for specific test cases can be captured as an executable method or function:

```
public void Triangle_TestSimpleScalene()
{
   // Setup
   Triangle t = new Triangle() {SideLengths = new[] {3.0, 4.0, 5.0}};

   // Stimulus
   double area = t.ComputeArea();

   // Observation
   if(6.0 != area)
       throw new Exception($"Got an area of {area}, when expect 6.0");
}
```

MAKING A TEST EXECUTABLE

Encapsulation of comparisons into Assertions:

```
public void Triangle_TestSimpleScalene()
{
    // Setup
    Triangle t = new Triangle() {SideLengths = new[] {3.0, 4.0, 5.0}};

    // Stimulus
    double area = t.ComputeArea();

    // Observation
    Assert.AreEqual(6.0, area);
    Assert.AreEqual(3.0, SideLength[0]);
}
```

USING TESTING FRAMEWORKS

- A testing framework provides tools for
  - Setting up test cases
  - Making observations, e.g., the Assert class
  - Organizing test cases
  - Executing test cases
- Examples:
  - Java: JUnit
  - C# (or more generally, anything .Net) : VS Unit Testing Framework, NUnit
  - See https://en.wikipedia.org/wiki/List_of_unit_testing_frameworks for a catalog of frameworks by language