Unit Testing

BY EVAN OLDS

CPT S 422

Unit Testing

- Testing “individual units” of source code. This could be functions, entire classes, or entire modules/components.
  - Generally considered to be white-box testing
- White-box (or glass-box) testing is testing that requires knowledge of (and usually access to) the code
- In contrast, black-box testing is done without internal knowledge of the product code
  - Tests the product against the end user, external specification
- Building a unit test
  - Set of inputs and expect outputs needed for tests
  - Unit tests not only testing the code to see if it crashes or not, should be testing to see if it matches the component specification
  - In this sense you are also testing the component specification too

Unit Testing

- Building a unit test
  - Set of inputs and expect outputs needed for tests
  - Unit tests not only testing the code to see if it crashes or not, should be testing to see if it matches the component specification
  - In this sense you are also testing the component specification too

Unit Testing Approaches

- Several unit testing frameworks are built right into modern IDEs
  - Visual Studio has them for C++ and C#
  - NUnit is a unit testing framework for .NET
    - Built into MonoDevelop and Xamarin Studio
- Advantages to using built-in stuff
  - Click a button to run all tests
  - Don't have to write test harness yourself
- Disadvantages to using built-in stuff
  - What are they (there is at least 1 major one)?
  - Discuss in class

What to test?

- Anything really, no preset definition of what a unit test does
- Common approaches
  - Include test with inputs and expected outputs for:
    - Common use cases
    - Edge cases (“minimums and maximums”)
  - Not only test things that are expected to succeed but also things that are expected to fail
- We will see several scenarios through the semester and discuss appropriate unit tests for those scenarios