Module 3 (Part II)

Unit Testing and Unit Testing Frameworks

❖ Introduction

- ***Unit testing*** is a type of software testing where individual units or components of a software are tested
- The purpose is to validate that each unit of the software code performs as expected
- Unit Testing is done during the development (coding phase) of an application by the developers
- Unit Tests isolate a section of code and verify its correctness
- A unit may be an individual function, method, procedure, module, or object

- ***Unit test frameworks*** are software tools to support writing and running unit tests, including a foundation on which to build tests and the functionality to execute the tests and report their results
- They are not solely tools for testing; they can also be used as development tools
- Unit test frameworks can contribute to almost every stage of software development, including software architecture and design, code implementation and debugging, performance optimization, and quality assurance

- Unit tests usually are developed concurrently with production code, but are not built into the final software product
- The relationship of unit tests to production code is shown in Figure 1-1

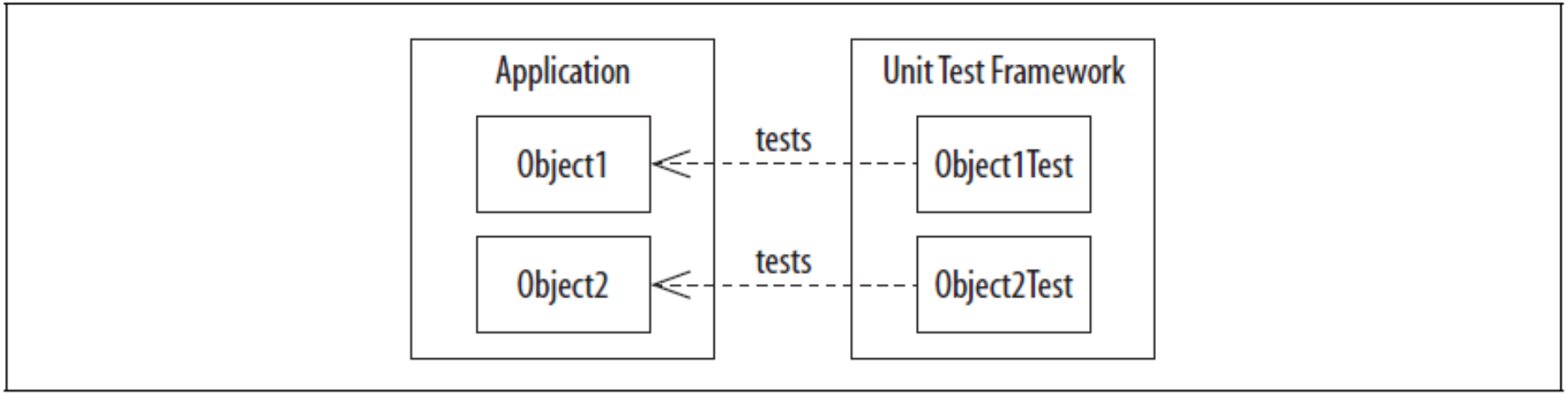

Figure 1-1. Production application and unit test framework

- An application is built from software objects linked together
- The unit tests use the application's objects, but exist inside the unit test framework
- A single unit test should test a particular behavior within the production code
- Its success or failure validates a single unit of code
- By starting with tests of the most basic functionality, then gradually building to tests of compound objects and behaviors
- A unit test framework can be used to verify very complex architectures

❖ Test Driven Development (TDD)

- The key rule of TDD can be summarized as "test twice, code once," by analogy to the carpenter's rule of "measure twice, cut once."
- "Test twice, code once" refers to the three-step procedure involved in any code change:

  1. Write a test of the new code and see it fail.
  2. Write the new code, doing "the simplest thing that could possibly worl
  3. See the test succeed, and refactor the code.

  These three basic steps are the *TDD cycle*.

- If TDD is followed rigorously, the code should never be left in a state in which a unit test fails

❖ Unit Testing and Quality Assurance

- Unit test frameworks are valuable when used for automated software testing as part of a quality assurance (QA) process
- In many software development groups, the QA process starts when new code is submitted, built, and unit tested
- Often, the unit tests include not only programmer tests, but also acceptance tests designed or written by the QA team
- If all the unit tests succeed, the code is provisionally accepted and sent to a QA engineer for inspection and testing

- Running the full suite of unit tests as the first step in QA has many benefits
- Unit testing doesn’t replace all other types of testing
- Unit tests are a powerful tool for QA
- Developers who use test-centric development report dramatic improvements in software quality, speed of development, and ability to make significant design changes on the fly
- These speed and quality advantages rapidly become apparent from the QA perspective as well

❖ Homegrown Unit Testing

- Writing simple tests comes naturally to most programmers
- The classic beginner exercise of writing a three-line program that prints "Hello world!" is a basic unit test of the development language and environment
- Find a software shop with no unit test framework in place and you may see developers writing their own little "toy programs" or "test utilities" to try out new code
- The sad thing about this approach is that the toy programs are thrown away once the developer is done with them
- Later, when something breaks, someone has to laboriously debug the production code, without benefit of the developer's test

- Just as many developers take the initiative and write test programs to try out small pieces of code, it’s common to find developers putting together basic, home-grown unit test frameworks that take care of their testing needs
- A test framework can be just a few lines of code to run unit tests and report the results
- Even a very simple framework can be the foundation for thorough testing of complex applications