Listen

Module Three

Learning Objectives

By the end of this module, you will meet these learning objectives:

- ✓ Analyze requirements to determine what to test and why
- ✓ Employ unit testing best practices to verify and validate given requirements
- ✓ Write unit tests to verify and validate code

Module Overview

In the last module, you learned about the different testing techniques and philosophies. But how do you get started writing a test? You have learned about black box and white box testing. In addition, you learned about regression, integration, and unit testing, but now it is time to write some tests. Unit testing is a powerful but simple technique with which the developer breaks down objects into smaller, testable pieces of inputs and outputs. Feature tests can also be created with unit tests. A feature test will contain multiple objects with object-to-object interactions. For example, writing a test to assign a class or enrolling a student to a class may include several objects, such as a student object and a class object. In your milestone assignment for this module, you will test features such as these. You will also discuss the importance of requirements in a discussion.

For the next three modules, we will focus on JUnit testing. What is JUnit? JUnit is the standard Java unit testing framework. It provides a rich interface for developing simple to complex tests. In addition, all of the unit testing principles you learned from previous modules can be implemented with JUnit. Furthermore, you can use third-party mock object libraries that will allow you to create more complex tests without needing to instantiate every object.

JUnit promotes the approach of “code a little, test a little” to increase programmer productivity and decrease the time spent debugging. This is also related to the idea of creating modular, testable code and decreasing technical debt, which we talked about in Module One. JUnit has many positive features: It is open source and simple; it provides annotations to identify test methods; tests can be organized into test suites; JUnit tests can be run automatically; and the tests are automatically checked with no need to manually read through a report of test results. JUnit also has built-in libraries that allow a developer to set up scenarios before the tests begin. In addition, those scenarios can be cleaned up after the test has been executed. Let's take a look at the JUnit testing basics for testing software. The biggest idea in this approach is to start simple and work your way up to testing the more complex pieces.

Module at a Glance

This is the recommended plan for completing the reading assignments and activities within the module. Additional information can be found in the module Resources section and on the module table of contents page.

1. Review the Module Three resources.
2. Post your initial response to this week's discussion (initial post due Thursday).
3. Submit the Module Three milestone.
4. Post peer responses to the discussion.