Testing Study Guide | CS 61B Spring 2018

sp18.datastructur.es/materials/lectures/lec7/lec7

Lecture Code

Code from this lecture available at

https://github.com/Berkeley-CS61B/lectureCode-sp18/tree/master/testing.

Overview

Why Test Code? In the real world chances are you won't have an autograder. When your code gets deployed into production, it is important that you know that it will work for simple cases as well as strange edge cases.

Test-Driven Development When provided an autograder, it is very easy to go "autograder happy". Instead of actually understanding the spec and the requirements for a project, a student may write some base implementation, smash their code against the autograder, fix some parts, and repeat until a test is passed. This process tends to be a bit lengthy and really is not the best use of time. We will introduce a new programming method, Test-Driven Development (TDD) where the programmer writes the tests for a function BEFORE the actual function is written. Since unit tests are written before the function is, it becomes much easier to isolate errors in your code. Additionally, writing unit test requires that you have a relatively solid understanding of the task that you are undertaking. A drawback of this method is that it can be fairly slow and also sometimes it can be easy to forget to test how functions interact with each other.

JUnit Tests JUnit is a package that can is used to debug programs in Java. An example function that comes from JUnit is `assertEquals(expected, actual)`. This function asserts true if expected and actual have the same value and false otherwise. There are a bunch of other JUnit functions such as `assertEquals`, `assertFalse`, and `assertNotNull`.

When writing JUnit tests, it is good practice to write '@Test' above the functionthat is testing. This allows for all your test methods to be run non statically.

Exercises

C Level

1. In general, is it good to write tests that test your entire program? How about for specific functions?

B Level

1. Write a testing method that will take in 2 arrays and see if they are equal. These arrays can have nested arrays and those nested arrays can have nested arrays and so forth.

2. If we have 2 classes that have identical qualities, will `assertEquals(Object o1, Object o2)` assert true or false?