EE 602
Java Assignment #2
Hualiang Li

What is software test? What are main types of tests?

Software unit tests help the developer to verify that the logic of a piece of the program is correct.

Running tests automatically helps to identify software regressions introduced by changes in the source code. Having a high test coverage of your code allows you to continue developing features without having to perform lots of manual tests. Main types of tests include unit test, integration test, component interface test, system test and operational acceptance test.

What is a UNIT test? Integration Test? Performance test? behavior test/interaction test? State Testing?

A unit test is a piece of code written by a developer that executes a specific functionality in the code to be tested and asserts a certain behavior or state.

An integration test aims to test the behavior of a component or the integration between a set of components. The term functional test is sometimes used as synonym for integration test. Integration tests check that the whole system works as intended, therefore they are reducing the need for intensive manual tests.

Performance tests are used to benchmark software components repeatedly. Their purpose is to ensure that the code under test runs fast enough even if it's under high load.

A test is a behavior test (also called interaction test) if it checks if certain methods were called with the correct input parameters.

A behavior test does not validate the result of a method call. State testing is about validating the result. Behavior testing is about testing the behavior of the application under test.

Where to put test class?

Typical, unit tests are created in a separate project or separate source folder to keep the test code separate from the real code.

How to create a JUNIT test case?

A JUnit test is a method contained in a class which is only used for testing. This is called a Test class. To write a JUnit 4 test you annotate a method with the @org.junit.Test annotation.

This method executes the code under test. You use an assert method, provided by

JUnit or another assert framework, to check an expected result versus the actual result. These method calls are typically called asserts or assert statements.

How to name it?

The class name followed by “Test”

What is a build path?

Locations of external Java class libraries needed to compile the application.

How to select he methods to test?

@Test
public void method()
The @Test annotation identifies a method as a test method.
@Test (expected = Exception.class)
Fails if the method does not throw the named exception.
@Test(timeout=100)
Fails if the method takes longer than 100 milliseconds.
@Before
public void method()
This method is executed before each test. It is used to prepare the test environment (e.g., read input data, initialize the class).
@After
public void method()
This method is executed after each test. It is used to cleanup the test environment (e.g., delete temporary data, restore defaults). It can also save memory by cleaning up expensive memory structures.
@BeforeClass
public static void method()
This method is executed once, before the start of all tests. It is used to perform time intensive activities, for example, to connect to a database. Methods marked with this annotation need to be defined as static to work with JUnit.
@AfterClass
public static void method()
This method is executed once, after all tests have been finished. It is used to perform clean-up activities, for example, to disconnect from a database. Methods annotated with this annotation need to be defined as static to work with JUnit.
@Ignore or @Ignore("Why disabled")
Ignores the test method. This is useful when the underlying code has been changed and the test case has not yet been adapted. Or if the execution time of this test is too long to be included. It is best practice to provide the optional description, why the

test is disabled.

Where is assertEquals() from ? How to use it to test?

It is from the class “Test”, the super class of testMaxHeapPriorityQueue. Because assertEquals() is a static method, we do not have to use object.method syntax, we can simply call it as assertEquals(expected, actual). Another way to use it is to use the method class.method, in this case, we can use testMaxHeapPriorityQueue.assertEquals(expected, actual). But, we are not inside this class, we do not have to specify which class it belong to.