Video Name: JUnit

Topics:
- purpose
- Test-Driven Development (TDD)
- JUnit framework
- Testing static methods

Java Class(es):
- utilities package, MyCalculations
- windchill package, MyTempConverter
- tests package, TestWindchillCalcs

Without a testing framework, users must manually test. This is error prone and tedious.

Vocabulary:
- Unit test. Method that runs one or more tests
- Test class/Test case. Collection of tests run at the same time, includes one or more test methods.
- Test runner. Utility used to run a test suite. Eclipse shows graphical presentation, but some test runners are run from command line.

Test-Driven Development (TDD). May be the most important agile practice.

1. Add a test (or tests) for a new unit of functionality (unit === method). The test specifies the behavior (e.g., if pass in parameter value of x, return value should be y).
2. Run tests, see that they fail. May write stubs so code compiles.
3. Implement new functionality.
4. The tests should immediately pass (i.e., the test specified the behavior; if you have correctly coded the behavior, the test should pass). In practice, tests occasionally need to be modified, as we revise our understanding of what functionality is needed.
5. Refactor - clean code rocks!

Red/Green refactoring. Red is the failing test, green is the implemented, correct code.

Usually put tests in a separate package, as customers do not need these tests.

To create a test case, select TestCase in Eclipse. We are using Junit 4. Eclipse will automatically bring in the needed libraries and will provide one sample test

Each test must have an @Test annotation. The test runner will automatically run every method with an @Test annotation. In other words, you don’t have to add your test methods to some main function, as
the JUnit framework automatically runs every @Test method. NOTE: common error is to forget the annotation, in which case your method will just not run.

Method stub. Will specify the function header (return type, name, parameters). In order to compile, often need to return some default value, such as 0 if the return type is an int.

Software engineering. Longer parameter names are more self-explanatory. Use temperature rather than temp (which could mean temporary).

Part of decision process is deciding whether function should be an instance method or a static/class method. A function that takes all the values it needs, does a calculation, then returns a result, should typically be static. Wind chill calculation is an example (no Windchill object needed!)

How to choose values? Edge conditions are good. In this example, wind speed of 5 is the cutoff (i.e., speeds less than 5 do not affect perceived temperature), so including tests with wind speed of 5, less than 5 and greater than 5 would be good.

Dealing with packages. Since JUnit test is in another package, you need to import the class being tested.

Use asserts to compare expected and actual results:

```java
int expected = -11;
int actual = MyTempConverter.calcWindChill(5, 0);
assertEquals(expected, actual);
```

For simple values, code may be easier to read if you don’t use the local variables:

```java
assertEquals(3, MyTempConverter.calcWindChill(10, 15));
```

It’s possible to run just one test by highlighting the method name and pressing run.