CS201 - Assignment 3, Part 1
Due: Friday February 28, *at the beginning of class*

One of the keys to writing good code is testing your code. This assignment is going to introduce you and get you setup to one framework for making this easier called `JUnit`.

1 `JUnit`

A "unit" test is a test that tests a very small, portion of code, often some functionality of a single function/method. `JUnit` is a framework for that allows you to write meaningful tests, run them regularly and easily identify which parts of your code might be problematic.

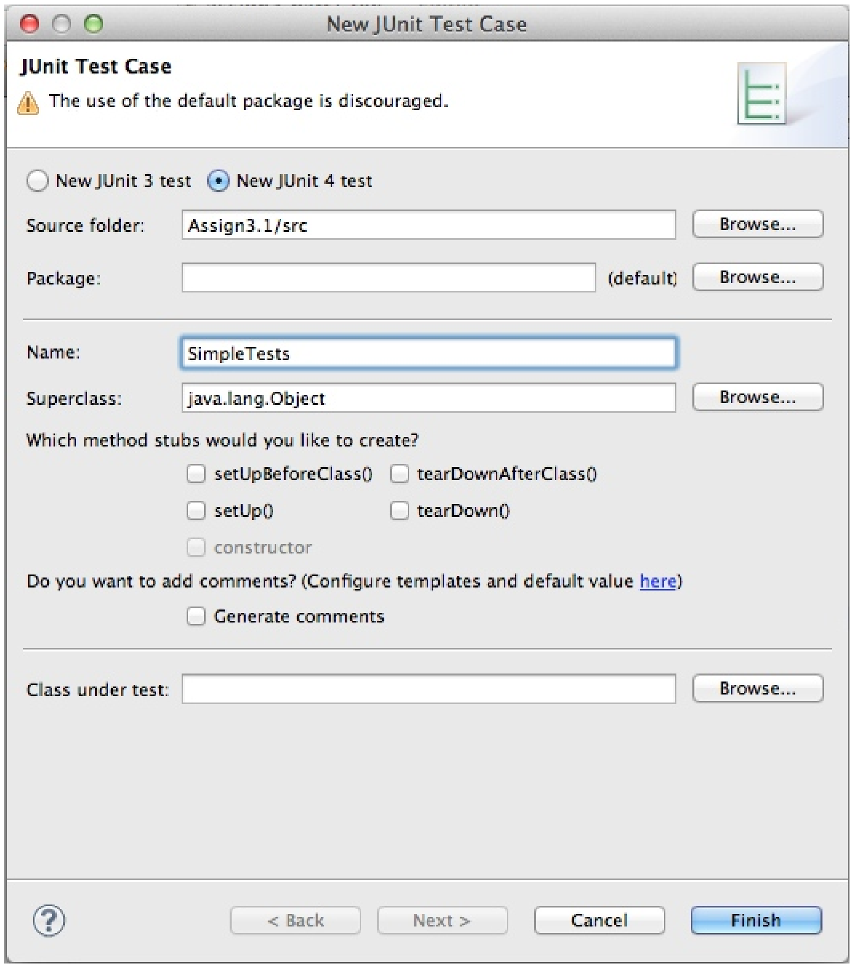
`JUnit` comes installed with Eclipse already. To make sure you can run `JUnit` properly, create a new Java project called "assign3.1". Inside, that create a class called `Simple` and copy and paste the code below into that class:

```
public class Simple {
    private boolean simple;

    public Simple(boolean b){
        simple = b;
    }

    public boolean isSimple(){
        return simple;
    }

    public String toString(){
        String returnMe = "";

        if( simple ){
            return "I'm simple";
        }

        return returnMe;
    }
}
```

Take a look at this class and make sure you understand what it's doing.

Now, we're going to create a few tests to check to see that this class is doing what we expect it to be doing.

To create a new set of tests, right click on your "assign3.1" project and then select `New` → `JUnit Test case`. This should pop up a dialog box that looks something like:



The use of the default package is discouraged.

New JUnit 3 test [✓ New JUnit 4 test]

Source folder: Assign3.1/src

Package: (default)

Name: SimpleTests

Superclass: java.lang.Object

Which method stubs would you like to create?

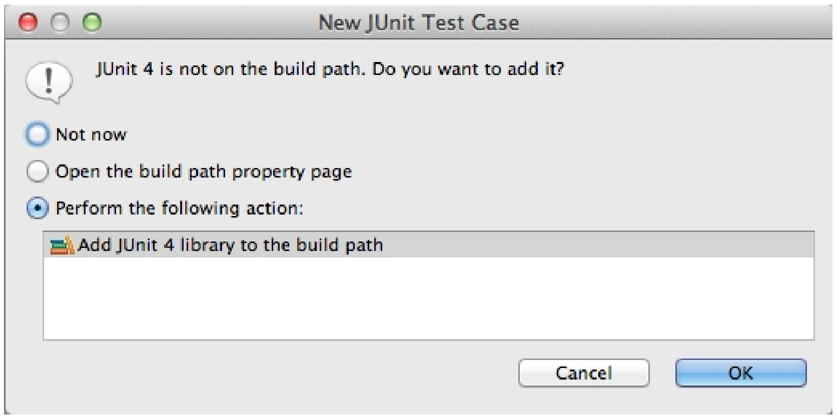
setUpBeforeClass() tearDownAfterClass() setUp() tearDown() constructor

Do you want to add comments? (Configure templates and default value here)

Generate comments

Class under test:

Under the "Name" field, put the name "SimpleTest" (like I've done in the screenshot above). Then press `Finish`. When you do, you should see a window pop-up:



JUnit 4 is not on the build path. Do you want to add it?

Not now

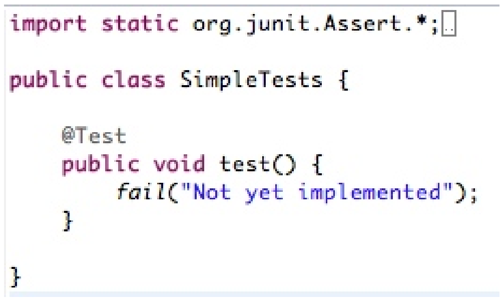
Open the build path property page

[✓ Perform the following action:]

Add JUnit 4 library to the build path

Click OK.

You should now see a class called SimpleTest that looks something like:



```
import static org.junit.Assert.*;

public class SimpleTests {

    @Test
    public void test() {
        fail("Not yet implemented");
    }

}
```

A unit testing class is like a normal Java class, however, it does not have a constructor and the methods that we write are treated as tests. Tests are `public void` methods that are annotated above the method with "@Test". We then can use special method calls inside these methods that check to see that answers are what we expect them to be. These are called "assert" statements, because they assert what we think should be true if everything is functioning properly.

Delete the method "test()" that is auto-generated for us in the `SimpleTest` class and then copy and paste these two tests:

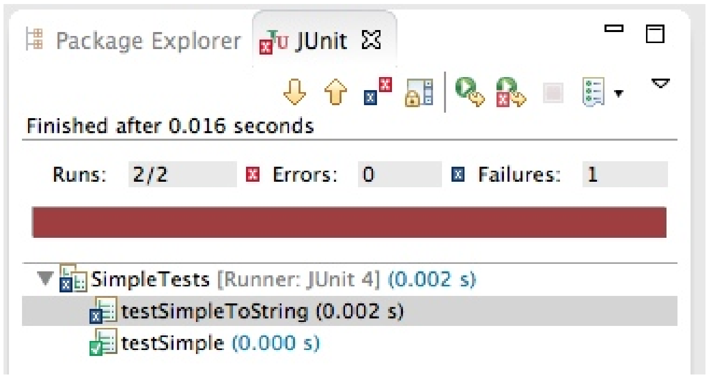
```
@Test
public void testSimple() {
    Simple s = new Simple(true);
    Simple s2 = new Simple(false);

    assertTrue(s.isSimple());
    assertFalse(s2.isSimple());
}

@Test
public void testSimpleToString() {
    Simple s = new Simple(true);
    Simple s2 = new Simple(false);

    assertEquals(s.toString(), "I'm simple");
    assertEquals(s2.toString(), "I'm not simple");
}
```

The first test checks to make sure that newly created Simple objects remember their state properly and then the second test checks to make sure that toString is working correctly. These tests use three different assert statements:

– `assertTrue`: takes a single parameter that is a `boolean` and checks to make sure that it is `true`. If the argument is not true (i.e. `false`), then the test that contains the assert statement will fail.

- `assertFalse`: just like `assertTrue` except passing is `false` and failing is `true`.
- `assertEquals`: takes two parameters and checks to make sure that they are equal to each other. If they are not, then the test that contains the assert statement will fail.

Should these tests pass? To run these tests, just click on the the green run arrow. When you do, you'll see the `JUnit` window open where the package explorer is (generally on the left) and should see:



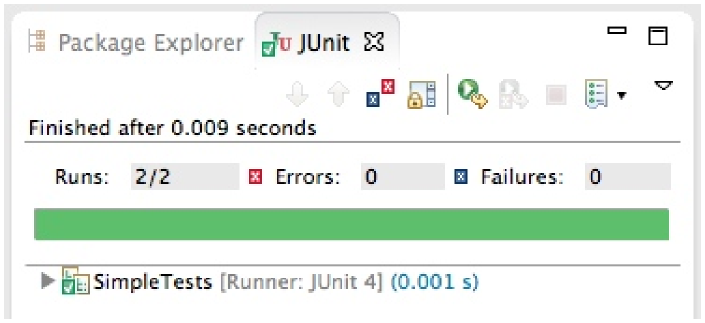
Finished after 0.016 seconds

Runs: 2/2 Errors: 0 Failures: 1

SimpleTests [Runner: JUnit 4] (0.002 s)
testSimpleToString (0.002 s)
testSimple (0.000 s)

The red bar indicates that one or more of your tests failed :(If you look below you can see that the `testSimple` test failed (green check mark), but the `testSimpleToString` check did not pass. If you double-click on this failing test it will take you to the assert statement that is failing.

Fix the `Simple` class so the `toString` method does the right thing, that is, returns "I'm not simple" when it is constructed with `false`. Then, go back to the `SimpleTest` class and run it again and you should see that it passes:

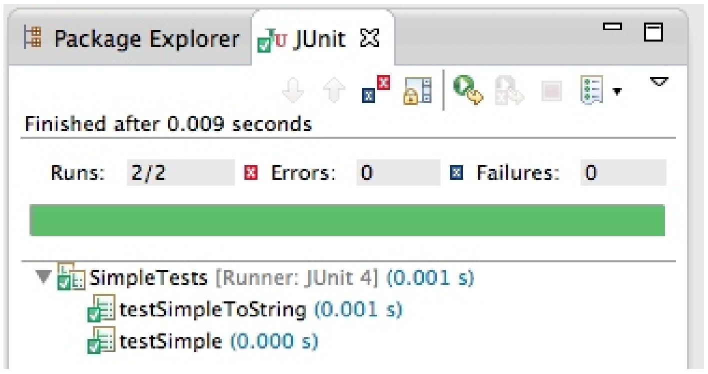


Finished after 0.009 seconds

Runs: 2/2 Errors: 0 Failures: 0

SimpleTests [Runner: JUnit 4] (0.001 s)

If you want to see all of the individual tests passing, you can click the arrow to the left of `SimpleTests`:



Finished after 0.009 seconds

Runs: 2/2 Errors: 0 Failures: 0

SimpleTests [Runner: JUnit 4] (0.001 s)
testSimpleToString (0.001 s)
testSimple (0.000 s)

The power of JUnit is that you can run these tests over and over again as you add new code. This makes sure that the new things you add didn't break anything that was working already!

2 Your Turn

- Copy and paste your `WarmUp` class from **Assignment 1** into your `Assign3.1` project.
- Create a new JUnit test class call `WarmUpTest`.
- Write at least two test methods that test different methods from your `WarmUp` class. Remember, to call `static` methods in another class you write the class name then a dot and then the name of the method. For example, to call the `isOdd` method you would write `WarmUp.isOdd`.

3 What to submit

Generate a `jar` file from your `Assign3.1` project and submit that online as "3.1".