Unit Testing Your Classes
2014-03-05
Unit Testing and JUnit

- Good tutorial here that we'll follow: http://www.vogella.com/tutorials/JUnit/article.html

- A unit test is a piece of code written by a developer that executes a specific functionality (usually a single method) to be tested.

- The % of code that is tested by unit tests is called test coverage. Good to shoot for 100% coverage.
Why Unit Tests?

- They ensure your code works as intended.
- Help ensure that code still works as intended when you modify other code (bug fixing or extending functionality).
- High coverage → develop features without having to perform lots of manual tests (and easily make changes).
Other Kinds of Tests

- Integration/Functional testing – test the behavior of a component or integration between a set of components.
- Let you translate a “user story” (like a use case for a customer) into a test suite – resemble user interaction
- Performance testing – benchmark component
What to test?

- Some developers aim to test every statement in their code.
- Can generally ignore getters and setters.
- Any method other than these should probably be tested.
- Especially important to have robust tests for critical or complex parts of your code.
JUnit Tests

• Current version is JUnit4, DrJava ships with Version 3 by default.
• Can download junit-4.*.jar from junit.org
• Change jar file to use in Preferences → Junit
• May also need to add junit jar file to class path (also through preferences).
• Examples of version 3 and 4.
JUnit Tests

• Usually write at least one test function for each non-trivial function in your code.

• May have multiple test functions for single function depending on branches, different possible outcomes.

• JUnit assumes test methods can be executed in any order → tests should not depend on each other.
Assert Statements

- `assertTrue/assertFalse` – checks that a boolean condition is true/false
- `assertEquals(expected, actual)` – tests that two values are the same
- `assertNull/assertNotNull` – checks that the object is/is-not null
- `assertSame/assertEquals` – checks that both variables do/do not refer to same object
setUp and tearDown

- A setUp method can be used to set up data fixtures.
- setUp is run before each test case (method)
- tearDown can be used, is called after each test method.
- In JUnit4, use @Before, @After annotations instead.
- also have @BeforeClass, @AfterClass (run once for the whole test class)