Lecture 15:
Code Reviews (cont)
Unit Testing

Notes:
- HW 9 out

Exercise

Q: Create an initial *checklist* for your code reviews
- What should reviewers be looking for?
- How will this change how you write code (if at all)?

Q: How will your group keep track of issues?
- How will your team capture the issues
- And check whether they have been resolved
GitHub supports integrated issue tracking

- add new issues
- assign to specific people
- add a tag (bug, enhancement, question, etc.)
- comment on issues
- can close issues

What to use issue tracking for ... 

- Code reviews (add issues for things found)
  - can also use pull requests
  - can also comment on versions of files (history view)
- Tasks (add an issue for each task)
- Bugs (when a bug is found, add & assign a bug)
- General todo's
Code review exercise (Part 2) ...

Do pass around reviews (everyone has code reviewed)
  – reviewer reviews without walkthrough
  – use the checklist
  – meet and discuss issues/defects

As you go:
1. consider checklist revisions
2. keep track of issues & needed changes (e.g., in GitHub)

Add some issues to GitHub (or find another approach …)

Unit Testing (a type of white box testing)

Test individual “components” (“units”) of the code
  – smallest unit is usually a function (class method)
  – can also include a sequence of function calls

Each unit is tested in “isolation”
  – tests don’t depend on other tests
  – sometimes shared code to set up a set of tests

Unit test cases can vary
  – often just given inputs, check correct outputs
  – but other types as well: abnormal cases (exceptions), calls
    other methods properly, data access, etc.
Unit Testing implies Automated Testing

In automated testing …
- test cases ("test suites") executed and verified by a program
- vs "manual" testing … execute and verify test cases by hand

Automated tests are written as code
- leads to additional coding (for testing)
- this code not part of the production/deployed system

Automated Testing

Some potential disadvantages of automated tests
- slows development speed (more code to write) … alternative?
- can be tricky in some cases (e.g., GUIs)

Advantages of automated testing
- easier to run regression tests!
- tests can be run quickly and consistently
- code serves to document the test (and possibly the API)

*In practice, code w/o unit tests is highly suspect!*
Automated Testing Frameworks

Lots of tools exist …

- We’ll look at JUnit (www.junit.org)
- an xUnit testing framework for Java
- first widely used unit testing framework

Many unit testing tools like JUnit for various languages

- NUnit (.Net), CppUnitTest, PyUnit, Test::Unit (Ruby)
- Built into XCode
- JavaScript has many (Jasmine, QUnit, Mocha, Buster.JS, YUI)
- PHPUnit

JUnit example ...

JUnit Extreme Basics

For each class MyClass, create a MyClassTest class file ...

```java
import static org.junit.Assert.assertEquals;
import junit.org.Test;

public class MyClassTest {
    @Test
    public void descriptiveNameForATestofFunctionF() {
        MyClass c = new MyClass();
        result = c.f(params);
        assertEquals(result, whatValueShouldBe);
    }
    ...
}
```
JUnit Extreme Basics

Compile test suite (class) ...
   javac -cp .:junit-4.12.jar MyClassTest

Run tests
   java -cp .:junit-4.12.jar:hamcrest-core-1.3.jar
   org.junit.runner.JunitCore MyClassTest

Note: Most IDE’s have this stuff built in

JUnit Extreme Basics

Output (test succeeds)
   JUnit version 4.12
         .
   Time: 0.006
   OK (1 test)

Output (test fails)
   JUnit version 4.12
         .E
   There was 1 failure:
   java.lang.AssertionError ...
   FAILURES!
   Test run: 1, Failures: 1
**UI automated testing frameworks**

Automate UI actions and run tests over them
- Many tools, not always free …

Some free Tools
- Silenium tools for web apps
- XCode has a built in UI tool

Note: Black-box testing, not white-box (unit) tests

---

**Exercise ...**

Break into “subgroups” and do the exercise …