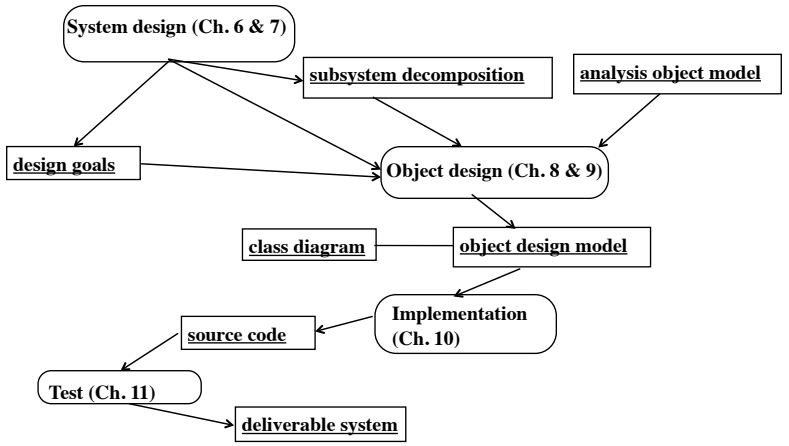
Testing:JUnit

CS 4354
Fall 2012

Jill Seaman

Review: Object-Oriented Development, part 2



An overview of Testing

- **Testing** is the process of finding differences between the expected behavior specified by system models and the observed behavior of the implemented system.
- **Unit testing:** individual program units or object classes are tested.
  --should focus on testing the functionality of objects.
- **Component testing:** several individual units are integrated to create composite components.
  --should focus on testing component/subsystem interfaces.
- **System testing:** all of the components in a system are integrated and the system is tested as a whole.
  --should focus on testing component interactions.
- **Performance testing** finds differences between nonfunctional requirements and actual system performance.

Goal of Testing

- **Goal** of testing is to identify faults and then to fix them.
  - ✦An attempt to show that the system is inconsistent with the system models.

Unit testing

- Unit testing is the process of testing individual components in isolation.
- **Goal**: complete test coverage of a class:
  - ✦Testing all operations associated with an object
  - ✦Setting and interrogating all object attributes
  - ✦Exercising the object in all possible states

Automation

- Automation of executing test cases has many benefits
  - ✦Fewer errors than manual testing
  - ✦Ensure that changing the source code does not introduce an error that would be exposed by a test case.
  - ✦The code is tested more frequently and errors can be detected earlier
- Disadvantage to automation of testing
  - ✦ It takes a while to set up the testing infrastructure.

JUnit

- A Java framework for writing and running unit tests
- Written by Kent Beck and Erich Gamma (Design Pattern authors)
- Written with “test first” and pattern-based development in mind
  - ✦Tests written before (or after) code
  - ✦Allows for regression testing
  - ✦Facilitates refactoring
- JUnit is Open Source
  - ✦www.junit.org
  - ✦JUnit Version 4, released Mar 2006

JUnit: test cases

- JUnit 4.x uses annotations to identify methods that are test methods.
- To write a test with JUnit:
  - ✦Annotate a method with @Test
  - ✦Use a method provided by JUnit to check the expected result of the code execution versus the actual result
    - ➡assertEquals(a,b)
    - ➡assertTrue(b)
    - ➡assertFalse(b)
    - ➡assertNotNull(x)

JUnit: running test cases

- To run your tests you can use
  - ✦Eclipse or
  - ✦NetBeans or
  - ✦org.junit.runner.JUnitCore
- Can be invoked manually by running the test class or automated by using a script (like ant)

JUnit 4 demo with eclipse

- Unit to be tested
  - ✦Team (from assignment 6), method: Team (League league)
  - ✦We want to make sure that when we construct a team using the constructor that the bidirectionality constraint holds:
    - ➡The team.league is league AND league.teams contains the team.
  - ✦To make the test fail initially we'll comment this line out of the Team constructor:

```
   public Team(League league) {
      this.league = league;
//    this.league.addTeam(this);

      lineups = new HashSet<Lineup>();
   }
```


JUnit 4 demo with eclipse: Create test class

- create a new source folder for the test:
  - ✦right (or ctrl) click the project, select New -> Source Folder, call it test
- create the test case class:
  - ✦right (ctrl) click on Team, select New -> JUnit Test Case
  - ✦select "New JUnit 4 test" and set source folder to test,
  - ✦press Next, select method(s) to test (Test(League)), press Finish
    - ➡ Note: if JUnit 4 is not on the build path, you'll be prompted to add it.
  - ✦Now you should have a class/file called TeamTest.java in the test source folder.

JUnit 4 demo with eclipse: Add test code, Run test

- In TeamTest.java, method testTeam(), add code:

```
League l = new League();
Team t = new Team(l);
assertEquals(t.getLeague(),l);
assertTrue(l.getTeams().contains(t));
```

  - ✦Note @Test before method indicates the method is a test method.
- Run the test case:
  - ✦right (ctrl) click on your new test class and select Run-As → JUnit Test.
  - ✦ It fails.
  - ✦Uncomment out the line we changed. It passes.

JUnit 4 demo with eclipse: The complete TeamTest.java file

- TeamTest.java

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

import org.junit.Test;

public class TeamTest {

   @Test
   public void testTeam() {
      League l = new League();
      Team t = new Team(l);
      assertEquals(t.getLeague(),l);
      assertTrue(l.getTeams().contains(t));
      }
}
```