Refactoring, Patterns, and Coupling

9-21-2011

Opening Discussion

- What questions do you have about things?
- Have you been writing any cool code?
- Have you seen anything cool in the news?
- Did you do the curriculum discussion?
- What do you think AI based automation is going to do to the workforce in the coming decade?

Refactoring in TDD

- Rely on your existing tests.
  - This is where the courage of TDD comes into play.
  - Unless you have Eclipse and do auto-refactoring. :)
- Still take small steps and runs tests a lot
- Expect to go from fail to pass.

Levels of Tests

- Unit Test
  - Tests functionality of a single unit of software.
- Integration Test
  - Test interactions between units.
- System Test
  - Test that the whole system meets specified requirements.

Test Suites

- Unit test systems allow you to define test suites.
- For JUnit use this:
  - @RunWith(Suite.class)
  - @Suite.SuiteClasses({Class1.class,Class2.class,...)

Design Patterns

- Christopher Alexander, *Notes on the Synthesis of Form*
- Purpose
  - Control complexity
  - Improve speed
  - Communication

Definitions

- Gamma *et al*.
  - Patterns are descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context.
- Beck *et al*.
  - A design pattern is a particular prose form of recording design information such the designs which have worked well in the past can be applied again in similar situations in the future.

Coupling

- A measure of how strongly dependent on software unit is on others.
- Dependencies are produced by any type of usage.
- You typically want lower/weaker coupling.

Cohesion

- This is a measure of how focused a unit of software is.
- You want high cohesion. Things in a single unit should be closely related to one another.
- Applies at many different “unit” levels.

Law of Demeter

- Do not collaborate with indirect objects.
- Also called “Don't talk to Strangers”.
- Implementation, only call methods on things you have direct access to.
  - When taken a fit too far, this can lead to an opposite smell.

Closing Comments

- Can you see how UML helps for system design after seeing this material?