CSE70: Lecture 7

- Project
  - Are you following the practices?
  - Are you doing testing?
- Checklist is out
- Change in the schedule
- For Today:
  - Design and Refactoring (from the reader)
Design review

- OOD
  - Class
    - Members/properties
  - Object
  - Abstraction, Encapsulation, Information Hiding
  - Inheritance
  - Associations
Collaboration

- Request of
  - information
  - service

- Communication
  - Messages

- UML
  - Sequence, communication (collaboration)
Persistence

• What is *persistence*?
• Persistent and Transient classes
  – domain?
  – interface?
• Persistent and Transient associations
• How can you make an object *persistent*?
  – In Java?
  – In c++?
Coupling and Cohesion

• Coupling
  – High/Tight
    • Pros and cons?
  – Loose
    • Pros and cons?

• Cohesion
  – class
  – interface
  – routine/method
Polymorphism and Interfaces

• Inheritance
  – … relationship

• Polymorphism
  – Treat equally instances of different classes
  – Instances of different classes perform *(differently)* the *same* action
  – Implemented with
    • … (in Java)
    • … (in c++)
  – Interfaces
Components

- Modular
  - Complete
  - Composable

- Independent
  - Development
  - Upgradeable
  - Purchasable
Refactoring

• Change the code, but not the *observable* behavior
  – Well defined steps
  – *Design improvement*
  – A code transformation

• Goal
  – Make code easier to understand and modify

• Two hats
Refactoring: why

- Improve the design
  - Decay is cumulative
  - Decay leads to duplication
- Improve readability
- Find bugs
- Increase productivity
Refactoring: when

• The rule of three
• Before changing the behavior
  – Add function
  – Fix a bug
• During a review
• When there are bad smells in the code
Refactoring: problems

- Databases
- Changing Interfaces
- Difficult changes
- Better to rewrite
  - When the code doesn’t work
- Approaching deadline
Bad smells

- Duplicated Code
- Long Method
- Large Class
- Long Parameter List
- Divergent Change
- Shotgun Surgery
- Feature Envy
- Data Clumps
- Primitive Obsession
- Switch Statements
- Parallel Inheritance Hierarchy

- Lazy Class
- Speculative Generality
- Temporary Fields
- Message Chains
- Middle Man
- Inappropriate intimacy
- Alternative Classes with Different Interfaces
- Incomplete Library Class
- Data Class
- Refused Bequest
- Comments
In the Lab

• UAT
• Concordion
  – HTML description of the requirement
  – Java/JUnit fixture
Summary

• Design
  – OOD
  – Persistence
  – Coupling and cohesion
  – Components

• Refactoring

• Next time
  – Refactoring
  – The cathedral and the bazaar (by Eric S. Raymond)