Midterm 1 Review

Concepts

A. UML Class Diagrams
    1. Components: Class, Association (including association name), Multiplicity Constraints, General Constraints, Generalization/Specialization, aggregation/composition, attributes
    2. Conceptual Model
    3. Design Model
        a. at higher level Many to Many often preserves or replicates associations or aggregations
        b. 1 to many, many to 1, and 1 to 1, (also 0:*) usually become attributes

B. Design Principles
    1. Single Responsibility Principle/Cohesion
    2. Information Hiding
        a. Spec/Abstract View
            1. Domain
                a. atomic
                    1. restricted atomic
                b. composite/aggregation
                c. structured (set, ordered set, multi-set, sequence, tree, map, graph)
                d. Invariants
                    1. Instance
                    2. Class Invariants
                        a. Treating the class as an object
            2. Behavior/Method Specification
                a. Pre-condition
                b. Post-condition
                c. What about static methods?
        b. Implementation
        c. In languages: C++ JavaProblems
            1. Spec vs. Implementation
                a. C++ .h vs .cpp files
                b. Java: yntactically it really doesn't
                    1. Javadoc
                    2. Use of Interfaces and Javadoc
                c. Pre-conditions/Post-conditions
                d. Domain definition
                    1. Invariants
                    2. Class Invariants
    3. Coupling/Cohesion
        a. Class perspective
        b. Method perspective
        c. Less Coupling == Higher Cohesion

C. Generalization/Specialization
    1. Conceptual
    2. Implementation

a. Inheritance
1. Why this is not real generalization/specialization
b. Composition
3. Design by Contract
a. Contract perspective
b. Pre-condition
1. Who is responsible?
2. Use of assertions as defensive programming
a. Frowned upon by some
c. Post-condition
1. Who is responsible?
d. Math equation
4. Good practice
a. Access a class only through methods
b. Every field is private
c. Why are protected fields in Java a little bit of a problem
a. Does making all fields private solve the problem?
d. Don't let names expost unnecessary detail

D. Patterns -- How used and why useful?
1. Proxy
a. Remote proxy
2. Command
3. State Pattern
4. Façade Pattern
5. Observer Pattern
6. Singleton
7. Visitor Pattern
8. Chain of responsibility
9. Question perspectives
1. What is the problem?
a. Give a specific example
2. Describe general solution using UML like diagram
a. Use UML to describe specific solution

E. Layers
1. Benefits
a. Layer reuse, modification, replacement
b. Reduce dependency (how?)
c. Easier to understand
2. Behavior
a. Down to Bottom then Up (Scenario I)
b. Down to Intermediate Level then up (Scenario II)
c. From Bottom to Top (Scenario III)

H. Dependency Inversion – disconnect to minimal abstraction
1. Don't depend on Implementation, depend on minimal abstraction (specification)
a. Usually based on method invocations

1. Specified by interface

b. Down Calls

1. Controller to Server
    a. IServer or ServerInterface or ServerSpecification

c. Up Calls
1. Can use same pattern
2. Why is Observer Pattern an example of dependency Inversion***
    a. Observable doesn't know how many observers there are.
    b. Observers defines "method" to be called by observable
    c. Subject (Observable) not tightly connected to Observer

I MVC

1. What are the model, view, and controller and their responsibilities
    a. A controller is often many controllers each with own "View" perspective and "Model" Perspective
2. Two views
    a. V <-> C <->M – often called Model View Presenter
    b. V -> C -> M -> V
    c. How are connections made
        1. Call backs (handlers)
        2. Observer Pattern
    d. How does a "Server" fit in?