COMP/EECE 4081

Final Study Guide

Fall 2012

This document contains a list of the topics and skills that will be covered on the final. For problems that require reading/interpreting/modifying/writing code, you need not memorize the exact APIs covered, but you should be able to understand/apply the APIs if I provide you a listing of their classes and method signatures.

This study guide does not cover material included in the midterm study guide; however, some midterm material may appear on the final.

System Sequence Diagrams (SSDs)

- Know the conventions and notations of SSDs.
- Be able to draw an SSD base on a use case or other description of user-system interaction.
- Know why creating SSDs is useful (hint: system operations and object collaborations).

Domain Models

- Know the notations and conventions of domain models (represented with class diagrams).
    - Know how to choose between modeling a thing as a attribute or a class.
    - Know how to model generalization.
        - Know/apply the 100% Rule.
        - Kow/apply the Is-a Rule.
        - Know/apply guidelines on when to model a subclass.
    - Know how to model abstract classes.
    - Know how to model composition.
        - Know the semantics implied by composition.
- Know how to create a domain model (class diagram) based on descriptive text.
    - Know/apply noun-phrase identification technique.
    - Know/apply the "think like a mapmaker" technique.
- Know how to read/interpret a domain model.
- Know why creating domain models is useful.

Design Class Diagrams

- Know how design class diagrams are different from domain models.
- Know why creating design class diagrams is useful.
- Know how design can be seen as a refinement process and how design class diagrams fit into the process.
- Know the notations and conventions of domain models (as covered in lecture).
    - Know how to distinguish between data types and non-data types.
- Know how to create a design class diagram from descriptive text and how to reverse engineer a design class diagram from code.

Design Sequence Diagrams

- Know why creating sequence diagrams is useful for design.
- Know the notations and conventions of sequence diagrams (as covered in lecture).

Object-Oriented Design

- Know/apply responsibility driven design.
- Know/apply the GRASP patterns covered in lecture:
  - Creator Pattern
  - Information Expert Pattern
  - Low Coupling Pattern
  - High Cohesion Pattern
  - Indirection Pattern
  - Protected Variations Pattern
  - Polymorphism Pattern
- Understand the relationship between coupling and cohesion.
- Know/apply the Law of Demeter.

Ethics and Software Reliability

- Understand that all "real" software contains bugs, and the types of moral/ethical challenges this creates.
- Be able to apply an ethical perspective to analyze difficult moral questions.

Verification and Validation

- Be able to define these terms: verification, validation, defect/bug/fault, error, failure, test case, test suite.
- Understand the "testing problem" and why it exists.
- Know what unit testing, integration testing, and system testing are, and how they're different.
- Know what blackbox testing and whitebox testing are, and how they're different.
- Know what regression testing is.
- Know what test-driven development (TDD) is, how to apply it, and what its pros/cons are.
- Know common criteria for selecting blackbox test cases.
- Know how to interpret/create control flow graphs.
- Know/apply the following whitebox testing techniques: statement coverage, condition coverage, and path coverage.
- Know about code reviews.
- Know how to use the JUnit API to write test cases.