CPSC 210 Software Construction
Final Exam Study Guide

1. At the start of each reading, there is a list of learning goals. Review these goals and make sure you can do the actions listed. For example, the reading entitled Object-Oriented Design II lists as a learning goal:
    - Use a design pattern to solve a design problem with desired properties.

2. Go through the posted exercise sets and their solutions. Review in-class exercises and their solutions.

3. The final exam will have a heavy emphasis on design. Approximately 75% of the marks on the exam will be for design-based activities. The remainder will be for code-writing exercises.

4. The final exam will be comprehensive, although there will be a greater emphasis on material covered since the last midterm and on material related to object-oriented design.

5. Here is a list of concepts/activities that may be covered on the final exam.
    - Extract models from code such as: intra-method control flow diagram, inter-method control flow diagrams (such as a call graph), type hierarchy diagram,…
    - Given a UML class diagram, communicate concisely and precisely about what the model communicates about the design of the system.
    - Design robust classes, including the design of exception classes and exception handling.
    - Design and implement unit tests for a given method specification.
    - Given a problem statement in English, produce an object-oriented design using the responsibility-driven design approach. In particular, you should be able to:
        i. Identify candidate classes.
        ii. Identify responsibilities and collaborations for those classes.
        iii. Produce a UML class diagram that describes the relationships between the classes in the design and includes significant fields and methods.
    - Define and apply the design principles discussed this term including: coupling, cohesion, the Open-Closed Principle, the Liskov Substitution Principle, …
    - Define and apply any of the design patterns discussed this term including: Composite Pattern, Decorator Pattern, Factory Pattern, … This includes implementing key aspects of these patterns in Java.
    - Given a UML class diagram, produce Java code that represents the design presented in the diagram. In particular, you should be able to:
        i. Correctly implement unidirectional and bi-directional associations, aggregations and compositions, inheritance and implements relationships.
        ii. Select an appropriate data structure from the Java Collection Framework (JCF) to represent an association, aggregation or composition based on the UML diagram and/or the problem statement for the design.

- Extract a containment hierarchy for code for an existing Swing GUI.
- Modify code for an existing GUI to add event handlers to respond to specific events.
- Implement a linked list, including common operations on lists, in Java.

6. The following concepts will not be covered:
    - Memory management in Java and Java garbage collection
    - Using Models in Swing programming
    - The Singleton Pattern
    - Designing code to produce a GUI with Swing