COSC436/716
Final Exam Information

Notes:

- You may have one sheet of 8.5”x11” paper (both sides) to use as a “Cheat sheet”.
  - **Any sheets used must be turned in with the exam** (so make a copy in advance if you want to keep yours).

- The ***focus*** will be on the concepts covered in the last half of the semester, with only a few exceptions, as noted in the topic list below.

- The exam will consist of:
  - A few multiple-choice questions
  - A few short answer questions (covering concepts outlined below)
  - Show a UML class diagram for a small set of given classes provided in code.
  - Modify an existing sample program fragment to utilize a specific design pattern.

Topics

- The general programming concepts and OOP features from Exam 1
  - These are all programming language constructs we’ve seen (or will see) concerning design pattern implementation:
    - Interfaces, Abstract Classes, Concrete classes.
      - Polymorphism
    - Inner classes
    - Anonymous classes
      - Anonymous classes are often direct “instantiations” of Interfaces that are only created as a parameter to a method in another object.
      - For some “official” examples, see: https://docs.oracle.com/javase/tutorial/java/javaOO/anonymousclasses.html

- UML Class Diagrams
  - Meaning and usage of Diagramming notation/symbols

- Design Patterns
  - Design Pattern Concepts
    - Categories of Patterns: Creational, Structural, Behavioral
      - Link: The 3 Types of Design Patterns All Developers Should Know
    - FYI: Software Design Patterns: A Guide
  - Design Patterns to know:

    While you should understand the intent and usage of the following, I have bolded the ones that I may ask you to implement on the final exam.
    - Adapter (Tutorials Point) (Refactoring Guru)
    - Builder (Tutorials Point) (Refactoring Guru)
    - **Command (Tutorials Point) (Refactoring Guru)**
    - **Decorator (Tutorials Point) (Refactoring Guru)**
    - Factory Method / Abstract Factory (Tutorials Point) (Refactoring Guru)
    - ***Iterator (Tutorials Point) (Refactoring Guru)***
    - ***Observer (Tutorials Point) (Refactoring Guru)***
    - Singleton (Tutorials Point) (Refactoring Guru)
    - ***State (Tutorials Point) (Refactoring Guru)***
    - ***Strategy (Tutorials Point) (Refactoring Guru)***
    - Visitor pattern (Tutorials Point) (Refactoring Guru)
    - Null Object (Tutorials Point) (Refactoring Guru)
      - NOTE: Implemented as the Optional/Option/Maybe type in many modern languages. (Some Code Examples)
  - *Other Patterns: While not covered in detail, understand the intent/purpose of the following:*
    - Chain of Responsibility (Tutorials Point) (Refactoring Guru)
    - Proxy (Tutorials Point) (Refactoring Guru)
    - Composite Pattern (Tutorials Point) (Refactoring Guru)