CS 4240: Principles of Software Design

Course Introduction

Tom Horton
horton.uva@gmail.com

Official Course Description:

- This course focuses on techniques for software design in the development of large and complex software systems.
- Topics will include software architecture, modeling (including UML), object-oriented design patterns, and processes for carrying out analysis and design.
- More advanced or recent developments may be included at the instructor's discretion.
- The course will balance an emphasis on design principles with an understanding of how to apply techniques and methods to create successful software systems.

Prerequisite:

- CS 216/2150 with a C- or better. Or:
  - at least two semesters experience in OO programming, in Java,…
  - with an understanding of inheritance, interfaces and polymorphism, plus…
  - understanding of basic data structures and libraries that support them.

Grading: HWs, Project

- Some aspects of this still TBD
- Homeworks (30%): a set of 3 to 6.
  - Some possibly done in pairs.
  - Project structure may affect number.
- Project (25%): Groups of 3.
- Balance of grade percentage may be adjusted.

SW Design Portfolio

- HWs and project will require some kind of report.
- Will be collected together to form a software design portfolio.
- Might be useful in job interviews.
- The point: I want your class work-products to be in a form that <u>could</u> demonstrate you have design skills

Class Participation

- I do expect you to attend class!
- Participation penalty: up to 5%
  - Occasional quizzes, exercises, activities during classes. Record your participation.
  - Maybe 10 or so total.
  - No penalty for missing a few.
  - Email me about reasonable absences.

Grading: Exams

- Exam 1: 20%. Tuesday, Sep 28. (Drop deadline is Oct. 5.)
- Exam 2: 20%. Tuesday, Nov. 16. (W/D deadline is Nov. 12.)
- Final Quiz: 5%. Take-home. Issued Tues., December 7 (last day of class), due by Monday, Dec. 13.
- (Possible alternative. 3 exams, the last during the final exam session, 9am, Dec. 17.)

Readings:

- You don’t have to buy a text book, but…
- Required reading using books and articles on-line or on-reserve
  - Some of these are in Safari on-line library, accessible with virginia.edu IP address
  - VPN or read on grounds

First Reading Assignment

- Chapter 1 of *Design Patterns Explained: A New Perspective on Object-Oriented Design* (2nd edn).
- By Alan Shalloway and James Trott.
- By Tuesday, August 31

Languages, Tools, Etc.

- Documents submitted in PDF
- Mix of Collab and webpages for course-site
- Collab will be used for submission
  - Files bundled with Zip or tar
- Drawing tool or UML tool (more later)
- Programming language(s)….

Java

- We'll use Java a lot at first. Why?
  - We all know it. It's a solid OO language.
  - Rich set of libraries and frameworks.
  - A *lingua franca* in OO writings.
  - Widely used (e.g. Android)
  - Strong tool support: IDEs, GUI, code generation, reverse engineering
- Others? C#, C++, Objective C, Python, Ruby
  - Project?

Eclipse Etc.

- I’ll encourage you to use Eclipse
  - Others possible: Netbeans, IntelliJ
- Explore large applications (hundreds of files, complex inheritance hierachies)
- Run JUnit tests
- Integrate with version control (svn), build tools (ant)
- Execute refactoring operations
- Debug
- Integrate with servers (e.g. Tomcat)

A Course Emphasis This Term:

- Professional SW Engineering Skills
- SW <u>Construction</u> tools
  - Build scripts. Why? ant with Java
  - Unit tests. JUnit. Test-first development.
  - Use of libraries. E.g. log4j, java.concurrent, others
  - Version control. Subversion, Redmine

Less Emphasis This Term

- Building according to a process
  - CS3240 does a lot of that
- We'll talk about it
  - Context for design
  - Requirements and design
- But the project will be less about this than, say, recent offerings of CS3240
- No Unified Process. Maybe a little agile.

Back to the Project

- Will emphasize forming and documenting a design
- Implementation to demonstrate design’s success.
- Team-based development
- System as part of larger code-base, made up of components, etc.
  - Not from 100% from scratch

Question:

- What kind of project interests you?

Course Topics (part 1):

- Context for design
- Design principles
  - Modularity, etc.
  - Functional design
  - (Briefly) Non-OO design
- Code Smells, Refactoring
- Object-oriented design
  - OO Analysis
  - OO modeling: Unified Modeling Language (UML)

Course Topics (part 2):

- Object-oriented Design (cont’d)
  - Abstraction, Inheritance, Interfaces
  - Packages
  - Libraries, Frameworks
- Design Patterns
- Software Architecture
  - Higher-level, system level
  - Plug-ins (Eclipse, Firefox, etc.)
- Case studies: code examples

Possible Advanced Topics:

- Some flexibility:
  - User-interface design?
  - Concurrent systems?
  - Web-based systems? Ruby on Rails?
  - Non-OO design? (C, web languages)

What Is Software Design?

- What would you say?

Class Activity: Groups of 3

- Mod 0 Groups:
  - List two things you do when you “do SW design”
- Mod 1 Groups:
  - List some things that are part of a SW design
- Mod 2 Groups:
  - List who might use design “outputs” and for what

What is Software Design?

- Maybe different ways to think about it?
    - Goals
    - Activities
    - Inputs, Outputs
    - Techniques, Skills
    - Principles
    - Descriptions

Your Answers:

What makes a design “good”?

- Qualities? Principles or rules?

Your Answers:

Someone’s Answers….

- Book: *Java Design: Building Better Apps & Applets* (2/e, 1999)
- Peter Coad and Mark Mayfield
- The book proposes that:
  Java has features support good OO design principles

Coad's book: design activities

- Design activities:
  1. Identify purpose and features
  2. Select classes
  3. Sketch a user-interface (UI)
  4. Work out dynamics with scenarios
  5. Build a class diagram

Coad’s book: design principles

- Design principles
  1. Design with composition rather than inheritance
  2. Design with interfaces
  3. Design in interfaces
  4. Design with notification