CS 372 - F01
Software Construction – 3 credits
Spring 2019

Instructor: Dr. Chris Hartman
Email: cmhartman@alaska.edu
Office: 525 Duckering
Office Phone: 474-5829 (email is always better)
Office Hours: MWF 2:30-3:30 or by appointment

Prerequisites: CS 311

Text: **Head First Design Patterns** by Elisabeth Freeman, Eric Freeman, Bert Bates and Kathy Sierra

Recommended Books:

Agile Software Development, Principles, Patterns, and Practices by Robert C. Martin

Clean Code: A Handbook of Agile Software Craftsmanship by Robert C. Martin

Code Complete: A Practical Handbook of Software Construction, 2nd ed. by Steve McConnell

The Pragmatic Programmer: From Journeyman to Master by Andrew Hunt and David Thomas

Course BlackBoard site at http://classes.alaska.edu

Schedule: MWF 3:30-4:30 PM, Duckering 536 until Monday, April 29th
Final Exam: 1:00-3:00 Thursday, December 13th

Assessment of the following items will be used in the following proportions to determine student grades.

Continuous Learning	10%
Class Participation	25%
Assignments	25%
Group Projects	30%
Final Exam	10%

Course description:
From the catalog: CS F372 Software Construction
Methods for programming and construction complete computer applications, including refactoring, performance measurement, process documentation, unit testing, version control, integrated development environments, debugging and debuggers, interpreting requirements, and design patterns. Prerequisite: CS 311. (3+0)

This is a required course for all Computer Science students, leading up to the senior capstone sequence of 471/472. In this course we will learn several techniques for writing large-scale programs that lead to better (specifically, more maintainable) software with fewer bugs.

You are expected to be proficient in the material from CS 311 (a prerequisite) such as advanced C++ programming, common data structures and algorithms, and basic software engineering principles, as well as object oriented techniques such as polymorphism and inheritance.

After taking this class, students should:

- Have increased proficiency in software development—specifying, designing, coding, refactoring, testing, debugging, and optimizing—and in performing code review.
- Understand various common software development methodologies.
- Be familiar with the use of a version-control system in managing a software project.
- Understand the different kinds of software testing, how unit tests are developed, and the test-driven development methodology.
- Have a good understanding of object-oriented design, including standard design patterns.

Instructional Methods – Classroom lectures, videos, discussion of external readings and case studies and in-class code review, group presentations.

Self Learning – You will be expected to spend at least two hours a week investigating continuous learning opportunities relevant to the material in this course. YouTube videos of conference talks, podcasts, blogs, etc.

Class Participation – You will be expected to participate in discussions of the reading material and case studies, and to actively participate in code-review sessions.

Group Projects – You will complete three small software development projects, each of which goes all the way from specification and design to coding and testing. There will be one project with groups of two students, one project with groups of three students, and one project with larger groups. You'll have the opportunity to choose whom to work with. Collaboration is encouraged, although each team member is required to participate roughly equally in every activity (design, implementation, test, documentation, presentation), and I may ask for an accounting of what each team member did. Each project will have multiple due dates, with different deliverables. On the preliminary due dates you will turn documents having to do with exploring the problem and initial design decisions. On the final date, you will turn in final design decisions and working code. Code and design documents will be handed in electronically (by committing in the repository before the deadline). Each project will have at least two in-class presentations (discussing design decisions and finally demonstrating working code.) While each team member will not be required to participate in each presentation, each team member must at least give some part of some presentation.

Assignments – Assignments will be required generally every two to three weeks. The assignments will reinforce lecture concepts, introduce material needed for group projects, and demonstrate application of critical thinking skills. Unless otherwise specified, all assignments must be done on an individual basis.

Policies – Examinations **must** be taken at the scheduled time. In particular, there **will be no** early final exams. You may discuss homework and programming assignments with others, but everything you turn in **must** be your own work.

Disabilities Services – The Office of Disability Services implements the Americans with Disabilities Act (ADA), and insures that UAF students have equal access to the campus and course materials. I will work with the Office of Disabilities Services to provide reasonable accommodation to students with disabilities.