ECE 3574: Applied Software Design

Chris Wyatt

Fall 2018

Welcome to ECE 3574: Applied Software Design

- CRNs 82992 and 82994
- Website: https://filebox.ece.vt.edu/~ECE3574
- Instructor: Chris Wyatt, clwyatt@vt.edu
- TAs: Ashin Thomas (ashinmarin@vt.edu) and Sampanna Khu (sampanna@vt.edu)

Today's Schedule:

- Description of the course
- Administrative details
- Expectations
- Course Tools Setup

Communication

Course Website: https://filebox.ece.vt.edu/~ECE3574

- syllabus, schedule, notes, etc.
- primary way materials are *distributed*.

GitHub: https://github.com

- distribute starter code
- demonstrate progress
- share code with instructors/TAs
- how you submit your assignments

Canvas: https://vt.instructure.com

- grades posted

Piazza: https://piazza.com/

- use it to ask (and answer) questions, to the class or to the instructors privately
- forum/wiki like software for QA, polls, announcements

Course Objectives

Having successfully completed this course, you should be able to:

- Use software design patterns and application programming interface (API) specifications to implement efficient and portable software.
- Design and implement multi-threaded and multi-process applications that rely on standardized inter-process communication and synchronization mechanisms.
- Design and implement complex software applications based on portable software frameworks and event-driven programming.
- Design, implement, and perform testing strategies including unit and integration testing.

Course Topics

We will be covering the following core topics.

- Generics and containers
- Inheritance and polymorphism
- Unit and integration testing
- Design patterns
- Using class-based software libraries
- Event-driven programming
- Concurrency: processes and threads
- Communication using shared memory and messages

Prerequisites

- ECE 2574 Data Structures and Algorithms. You are expected to be competent in the basics of programming with C++ and the use of data structures and algorithms to solve problems.
- It is helpful to be familiar with Unix systems (e.g. taken 2524) but it is not considered a prerequisite.

Text and Resources

Readings will be assigned from the following books. They can be read online through the library, although there is limited access, or you can purchase your own (both are very good and offer solid, practical advice on programming). There will also be assigned readings from various online sources.

- Clean Code, Robert C. Martin, Prentice Hall 2009
- The Pragmatic Programmer, Andrew Hunt and David Thomas, Addison-Wesley, 2000

Additional Resources

- Pro Git book https://git-scm.com/book/en/v2
- CMake Tutorial http://www.cmake.org/cmake-tutorial/
- C++ Reference http://en.cppreference.com/w/
- QT Documentation http://doc.qt.io

Software

A modern C++ compiler with sufficient C++11 support is required. Only specified compiler extensions and libraries may be used. We will use the open source CMake application for managing the build process (see www.cmake.org) and the git source code management tool (see git-scm.com).

Recommended Compilers:

- GCC >= 4.8
- Clang >= 3.5
- VC++ >= 19 (e.g. Visual Studio 2017)

Development Environments

- For development you can use your favorite editor and a command console to invoke the compiler toolchain, or you can use any integrated development environment (IDE) supported by CMake, including Visual Studio on Windows and XCode on the Mac.
- There are several other options including QT Creator and CLion.
- Use whatever works for you but note the project defines a reference environment using a virtual machine that, **for grading purposes**, is the final arbiter of working code.

Whatever environment you choose, you should be adept at using it to write and debug code.

Grading

Coursework consists of in-class exercises, a project with 5 milestones, and a final exam. The grades will be computed as follows:

- Exercises: 10%
- Project Milestones: 30%
- Final Project: 45%
- Final Exam: 15%

All graded work, other than the in-class exercises, is expected to be the original work of the individual student.

Exercises

The exercises are worked through in-class after a short lecture on the material for that day, and are due by midnight the day of class.

Project

https://filebox.ece.vt.edu/~ECE3574/project/index.html

- The project is divided into milestones with requirements that correspond to the material covered to that point in the course.
- These milestones have explicit due dates worth 6 percent each and are to encourage you to not procrastinate and keep up.
- The final project submission is due the last day of classes.

No project work will be accepted past the due date.

Auto-grader and Milestone Feedback

- For each milestone there is an auto-grader available: https:://grader.ece.vt.edu
- When you first sign-in using your PID/password and 2FA it will ask you for a username. **Please use your PID**.
- Each milestone will have an entry where you enter a git commit hash or tag. The grader will pull your code from GitHub, run my tests for that milestone and give you feedback.
- You will also receive more detailed feedback about your code within 2 weeks of the Milestone due date.

Prior Expectations

You are expected to understand basic computer organization and C++ syntax from ECE 1574

- Types, including references and pointers
- How to write and call a function, passing arguments and returning values
- How to write and use a basic class
- How to read and write files, including parsing techniques
- How to do proper memory allocation and handling

Prior Expectations

You are expected to understand selection and use of common data structures and algorithms from ECE 2574

- Array-based and Link-based lists, stacks, queues, deques, and priority queues
- Tree and Hash Table based dictionaries
- Algorithms and used for sorting and searching

Prior Expectations

You are expected to be able to write, compile, and debug C++ code using good engineering practices.

- Perform the mechanics of compiling a program
- Understand how to read and correct compile-time errors
- Understand runtime-errors and how to identify why they are occurring
- Use an incremental development technique
- Have good debugging skills (hypothesis testing)
- Be able to identify and use appropriate reference material to solve problems

Honor Code

We encourage you to help each other, but there are rules:

- Project work must be primarily an individual effort. You are encouraged to discuss approaches with other students but your code must be your own.
- You may not use materials produced as course work by others, whether in this or previous semesters, nor may you provide work for others to use.
- As a general rule, when helping another student, neither your solution or theirs should be visible. Make a habit of closing your laptops and use paper or a whiteboard.
- You can use material from external sources (e.g. StackOverflow), but only with proper attribution and integration into your code base.
- You can use any code provided by this semester's 3574 Instructor or TAs without attribution.

We do use a code comparison system.

Honor Code Example #1

Sue and Bob sit at the same table in the Library working on their project. They talk about general approaches and draw some diagrams and pseudo-code on a nearby whiteboard. Sue finds a very useful class in the C++ standard library and sends Bob a link to the reference page.

OK?

Honor Code Example #2

Jane and Lee are good friends and like to work together on classwork. As they work together Jane reads her code from her screen aloud so that Lee can follow along and type it in on his laptop.

OK?

Honor Code Example #3

Jake has been working hard but is struggling with his classwork and is on academic probation. He hires a programmer from a freelance website to write a solution for him to use, but only as a reference. He makes many improvements to the code including better variable names and adds lots of comments.

OK?

Tips for success

Here are the tips I have for doing well in the course.

- Attend class and do the exercises
- Start work on milestones as soon as they are released and ask questions as soon as you have them
- Attend office hours if you get behind. If you can't make these then contact me to setup one-off meetings.
- Find a peer or peers to work with (just take care with the honor code)

Ultimately however if you cannot consistently devote at least 12 hours per week on the course it is unlikely you will do well.

Milestone 0

https://filebox.ece.vt.edu/~ECE3574/project/milestone0.html

- The goal of this first milestone is to get used to the course work-flow and be sure you understand how to submit code for grading.
- You will have to read the starter code and make some modifications and extensions.
- Due 9/7 by 11:59 pm

Questions?

Exercise 01: Setup

See Website.

Next Actions

- Make sure you complete today's Exercise 01.
- Read Chapter 1 and 2 of the Pro Git Book (you can skip sections 1.2 and 1.5)
- Start on Milestone 0 (at least read the description and related material)