Syllabus (Fall 2016)

1 Official Yale course listing

CPSC 427 01 (10516) /
CPSC 527 01 (12424)
Object-Oriented Programming
Michael Fischer
MW 4.00–5.15 AKW 200

Fall 2016
Final exam scheduled (Group 37)
12/17/2016 S 7.00p

Object-oriented programming as a means to efficient, reliable, modular, reusable code. Use of classes, derivation, templates, name-hiding, exceptions, polymorphic functions, and other features of C++.

After CPSC 223.

2 Course Description

2.1 Industrial Strength Programming

Programming, broadly defined, is the activity of getting a computer to perform a desired task. It generally takes the form of constructing an artifact, called a *program*, which in turn instructs the computer in the sequence of steps needed to carry out the task. The program is expressed in a *programming language*.

The primary requirement of any program is that it *correctly* performs the desired task. For many small applications, the task is easily described, and the goal of the programmer is to get the job done as quickly and easily as possible. This is typical of homework assignments in beginning programming courses. The specification is provided by the instructor, and the overriding concern is to have something to turn in by the deadline. Small throwaway jobs occur in the real world, too, where the primary object is to get answers quickly to one-off problems. Succinct interpretive languages such as Python and JavaScript are often well-suited to such applications.

By way of contrast, large software systems typically have lifetimes measured in decades. They are built by teams of programmers. The programmers who create them are often not the same people who maintain them over the years. The systems must run correctly in a variety of hardware and software environments. They are not static but are continually evolving to meet changing needs.

Experience shows that a disciplined software development process is needed in order to end up with a system that meets its specifications and is useful in the real world. How to specify, build, manage, and maintain large systems is the realm of *software engineering*, covered by Yale course CPSC 439.

This course focuses on the design and implementation component of the development process. In addition to functional correctness, goals for industrial strength software include:

Verifiability It's not enough that the program *be* correct; there must also be confidence that it *is* correct. Such confidence is achieved through a combination of proper use of language constructs such as type systems to enforce constraints, formal verification tools, human understanding of the code, and testing.

Efficiency The program must satisfactorily perform its task within the available resources such as time, memory, communication bandwidth, etc.

Understandability and maintainability The program must be understandable by humans other than the original developers. This requires that the program be modular, clean, elegant, and well documented. As in any design process, style and judgment are important to the finished product.

Deployability The program's dependencies on its environment must be clearly specified and easily satisfied on target systems. Maintaining backwards compatibility as new features are added aids in achieving this goal.

Reliability, robustness, resilience, and safety These all refer to the program's ability to respond appropriately to errors during execution and to changes in requirements during the program's lifecycle. The program must handle known risks predictabily and unknown risks gracefully. It is rarely acceptable in large systems to ignore error conditions or for the program to crash.

Security This is the program's ability to resist a malicious adversary who deliberately attempts to cause it to fail or misbehave or to otherwise violate its intended properties.

Since good software is expensive to construct and maintain, an additional goal is that it be *reusable* in other applications.

2.2 Topics and Emphasis

This course will use the version of the C++ programming language known as C++14. C++ is large and complicated. It contains many features that *allow* it to be used to satisfy the requirements stated above, but they don't *ensure* good code. These same features can be misused to create code that is opaque and indecipherable. A good understanding of the language is necessary in order to get the benefits that it can provide.

The course will cover many C++ concepts, explaining for each what the purpose is, some of its intended uses, and some of the pitfalls it presents. Topics include design principles and standards, the C/C++ memory model, objects and classes, constructors and destructors, types, casts and conversions, operator definitions, name-hiding, restrictions on data modification, derivation and inheritance, abstract classes, polymorphism, virtual functions, multiple inheritance, templates, and exceptions. The course will make use of some important class libraries such as the C++ Standard Library, Boost, GTK+, and gtkmm. Other topics as time permits include advanced design patterns, programming for efficiency and testability, performance measurement, and debugging.

These topics will be reinforced by frequent programming assignments. The goal of the assignments will be to learn how to apply the principles for good design to actual code. Submissions will be judged on their design as well as on their functionality.

3 Course materials

Textbook:

- A. Fischer, *Exploring C++*, manuscript, 2009. This will be available for free on the course web site.

C++ Reference: A good source of reference material on C++ is the web site `http://www.cplusplus.com`. It is generally up to date and accurate, and I encourage you to become familiar with it.

Beware of the many web sites out there that have C++ information that is outdated, incomplete, or just plain wrong. Not only are many people misinformed about how the language actually works, but some important things have changed with the advent of C++14, so you should also be careful about trusting out-of-date information.

Other Materials: There are many books on C++ directed at different audiences. A list of some of these materials will be put on the course web site but will not be required.

Course Websites: This class will use two websites:

- Classes∗v2: `https://classesv2.yale.edu/portal/site/cpsc427_f16`
- Zoo website: `http://zoo.cs.yale.edu/classes/cs427/2016f/index.html`

Classes∗v2 will be used for homework assignments and submissions, grading feedback, and emailed announcements. The Zoo website will be used for the syllabus, handouts, lecture notes, general announcements, and other course-related materials. Note that there is a link to the Zoo website in the left-hand column of each Classes∗v2 course page.

4 Course Mechanics

Prerequisite: The prerequisite for this course is CPSC 223 (Data Structures) or equivalent. The ability to write significant programs in the C or Java programming language is required. The course also assumes a familiarity with basic computer science concepts such as are covered in CPSC 201 and CPSC 202. Graduate students should have an equivalent background.

Requirements: Course requirements include programming assignments and/or written problem sets (∼45%), a midterm exam (∼20%), and a final exam (∼35%). The approximate weights of each in determining the course grade are subject to change depending on the number and difficulty of the assignments actually given. Graduate students taking the course will be expected to perform at a higher level than undergraduates and may be required to do additional work.

Assignments and other announcements: Written problem sets and programming assignments will be posted from time to time on the handouts page of the Zoo website and will be announced using Classes∗v2. Other course announcements will be posted on the Zoo website home page. It is your responsibility to check these pages frequently.

Help with technical questions: The graduate teaching assistants and undergraduate course graders will be holding scheduled office hours during the term as posted on the course web site. You are encouraged to meet with them with questions about the lectures, textbook, problem sets, and C++ programming generally. You may also send questions by email. Please copy the instructor on all such emails. This will often result in a quicker response, since whoever is available at the time can decide to answer it. The response will also go to both TA and instructor so both will know that the questions have been addressed.

Other Questions: All questions about assessment and grading should be taken first to the TA. If the TA is unable to resolve your questions to your satisfaction, or if you wish to talk to me privately about any matter, you are always welcome to contact me, either by email or in person. Email is the preferred way to arrange an appointment with me.

5 Policies

Late policy: Assignments will be due at 11:55 pm on the night of the stated due date. Late work will generally be subject to a penalty of 5% per day late unless accompanied by a Dean's excuse. A 2-hour grace period following the *original* due date will be granted during which no late penalty will be assessed. However, there will be no grace period in counting the number of days late for assignments turned in after the grace period. Work more than 4 days late will not be accepted, but alternative means for making up missed work may be arranged on an individual basis with a Dean's excuse.

Please contact the instructor as soon as you find out that you are unable to submit work on time or to attend a scheduled exam so that suitable makeup arrangements can be made.

Policy on Working Together: This course follows the Yale College Undergraduate Regulations and the Yale Graduate School Professional Ethics and Regulations policies regarding cheating, plagiarism, and documentation, with which you should familiarize yourself. Briefly, if you use someone else's work, you must acknowledge it. If it's a piece of code, place the acknowledgment in your source file and explain clearly what parts are not your own. Similarly, if it's in a paper, the acknowledgment belongs in the paper itself. All work not so acknowledged must be your own.

You may of course discuss the lectures and readings with your classmates in order to improve your understanding of the subject matter. Helping each other learn to use the tools in the Zoo is also okay. However, the design and implementation of all programs and all submitted work must be your own except where other sources are explicitly noted.

You must never let another student see your work, either before or after the due date of the assignment. Sometimes you may be tempted to "help" your friends by letting them see your solution. Don't! This doesn't help them. To the contrary, it allows them to avoid the hard work of learning the material and deprives them of the educational experience they came to Yale to get.

You are always free (and encouraged) to come in and ask the TA or instructor for help about anything concerning the course. Please talk to the instructor if you have any questions about this policy.

Avoiding Plagiarism: You may neither copy from another student nor permit your own work to be copied, unless explicit permission is given for such collaborations. If your work is found in the possession of another student, you and the other student are equally guilty of plagiarism. To avoid unintended involvement in plagiarism, *your work should never be in the possession of another student*. Do not ask someone else to deliver or pick up your work. Do not let another student "borrow" your code to compare with theirs. Keep your files protected so that others cannot read them and carefully guard your password. Do not leave printed work in public areas such as the Zoo or in accessible wastebaskets. If you think your password may have been compromised, you must change it immediately and notify the instructor.

Policy on Computer Problems: The Yale College policy on "Use of Computers and Postponement of Work" in the Yale College Programs of Study, Academic Regulations, applies to this course.

It is reproduced below.

> "Problems that may arise from the use of computers, software, and printers normally are not considered legitimate reasons for the postponement of work. A student who uses computers is responsible for operating them properly and completing work on time. (It is expected that a student will exercise reasonable prudence to safeguard materials, including saving data on removable disks at frequent intervals and making duplicate copies of work files.) Any computer work should be completed well in advance of the deadline in order to avoid last-minute technical problems as well as delays caused by heavy demand on shared computer resources in Yale College."

Particularly relevant for this course are the cautions against leaving a programming assignment to the last minute when machines might be busy, printers broken, and so forth, and about safeguarding your data.

Policy on Technology in the Classroom: Cell phones are not to be used in class. Tablets and laptops are allowed only for course-related activities such as note-taking, reading slides and other materials from the course website, and quick internet searches on topics relevant to the lecture. Their use must limited so as to not distract you from paying attention in class. If in doubt, ask the instructor or TA first. Games, instant-messaging, reading email, and other diversions are not permitted. You may be asked to leave the class if these rules are not followed.

6 Computing Facilities

The Zoo: This course will use the Computer Science Department's educational computing facility, affectionately known as the Zoo. This facility contains modern workstations running Fedora Linux 24. You will need to use these machines to prepare coursework. Look at

```
http://zoo.cs.yale.edu/help/
```

for information on getting started if you are new to the Zoo.

These days, most of you have your own laptops and may be wondering why you should be bothered with using a new computer system. The answer is because code development software is still not completely compatible across multiple platforms. If it works on your Mac or Windows PC but fails when the graders run it on the Zoo, you will lose points. If you ask for help with compiler errors on your personal machine, we might not be in a position to answer your questions. In short, develop your code on the Zoo! Regardless of where the code is developed, *your assignments will be graded according to how well they work on the Zoo*. Submission of assignments will be through Classes∗v2.

The Zoo machines support remote access via the SSH and VNC protocols. These enable you to do your work remotely when it is inconvenient to go in person to the Zoo. Instructions on how to configure your machine for remote access will be posted to the course web site.

Course directory: The shared course directory, `/c/cs427`, is located on the Zoo server. You can access it from your Zoo course account. It will contain any software needed for this course and miscellaneous documentation and files. Public files there can be accessed via the web as well as from a Zoo node.