

Object-Oriented Programming and Reuse Syllabus

1 Basic Course Information

Objectives:

This course will explore the techniques of object-oriented programming and design. The emphasis will be upon the development of clean interfaces that permit easy modification and reuse of software components. Other techniques, drawn from outside the object-oriented approach, that significantly contribute to this goal will also be discussed. Students will gain facility in an object-oriented programming language and will learn the constructs that differentiate such languages from others.

This course will explore the idioms and styles of object-oriented programming, with emphasis upon how these contribute to reusable software components. Object-oriented design will also be discussed as an integral part of this process.

Comparisons of similar components from different sources such as the Booch, g++lib, NIH-Lib, and `std` libraries, will be used to explore alternative choices in interface design and to prompt discussion of how these affect utility.

Alternate and future directions of OOP languages including Java, the Common Lisp Object System, and persistent programming languages will also be explored to help highlight the shortcomings of current languages and of the OO approach in general.

Required Texts:

Horstmann, *Practical Object-Oriented Development in C++ and Java*, 1997, John Wiley & Sons

Fowler & Scott, *UML Distilled*, 1997, Addison-Wesley

1.1 Prerequisites:

1. This is *not* a beginning programming course in C++, but a course for students wishing to explore the possibilities offered by object-oriented programming beyond the rudiments of basic programming. Familiarity with an object-oriented programming language (preferably C++ or Java) is assumed. It is possible for extremely skilled programmers to teach themselves C++ for the first time while taking this class, but this is not recommended

and students who attempt this must take the responsibility for bringing themselves up to speed quickly.¹

2. Data structures. Students should be familiar with the standard data structures, algorithms, and abstractions that would be presented in a course such as CS361 or CS359. Students should be comfortable with defining new data types, the use of pointers, and conventional programming techniques for processing arrays, lists, trees, and hash tables. Students need not be capable of easily recreating the more complex algorithms from CS361 (e.g., balanced trees, complex graph algorithms) but should have an awareness of them and know when such algorithms would be useful.

2 Communications

2.1 Instructor:

Steven J Zeil
248D Education Building e-mail: zeil@cs.odu.edu
(757) 683-4928 Fax: (757) 683-4900

Course Repository and Home Page:

Most on-line materials for this course can be found at <http://www.cs.odu.edu/~zeil/cs412/>.

Files related to assignments, however, will be available via anonymous ftp at <ftp.cs.odu.edu> in `/pub/zeil/cs412`. Students logged into a CS Dept. Unix machine will also be able to find these files at `~zeil/cs412`.

Office Hours:

Weekly office hours are posted on a calendar accessible from the course web page.

e-mail:

I will use electronic mail on the CS Dept. network for timely communication, especially of clarification/corrections/changes to homework or projects. Students should check their e-mail on a regular basis.

3 Assignments

Students should expect assignments approximately every two weeks. Many of these will involve programming in C++ and Java.

All students in the course will receive accounts on the CS Dept. network of Unix workstations. Students who are unfamiliar with Unix must make it their own responsibility to learn how to use it. A Unix tutorial is available.

¹And, in fairness, most students who attempt this do not do at all well.

C++

C++ is a language that is still undergoing rapid change. Consequently, code accepted by one C++ compiler may fail to compile in another. The “official” compiler for this course is the Free Software Foundation’s `g++` (also known as `gcc` or GNU CC), version 2.95 or EGCS 1.1.2. This is the compiler that the instructor and/or grader will use in evaluating and grading projects. If you have access to other compilers, you may use them, but *you* are responsible for making sure that their projects can be compiled by the instructor and/or the course’s grader using the official compiler. You may want to develop your programs on the most convenient compiler and then port it over to the official environment. Please don’t underestimate the amount of time that may be involved in coping with subtle differences among compilers.

4 Course Policies

Late Submissions:

Late papers and projects and make-up exams will not normally be permitted. I will give appropriate consideration to documented emergencies, but such arrangements must be made *prior to the due date* in any situations where the conflict is foreseeable.

Extensions to due dates will not be granted simply to allow “porting” from one system to another. “But I had it working on my office PC!” is not an acceptable excuse. In a similar vein, anyone who has achieved junior-level or higher status in a CS program should by now be aware that the network and workstations generally get overloaded just before assignments are due, and that machine crashes and down-time are a fact of life in this field. Students are expected to *plan* for these problems. Except in extreme cases, these are not grounds for extension of a due date.

Attendance:

Attendance at classes is not generally required, but students are responsible for all material covered and announcements made in class. Consequently, if you are going to miss class, be sure to get notes, handouts, etc., from another class member.

Academic Honesty:

Everything turned in for grading in this course must be your own work. The instructor reserves the right to question a student orally or in writing and to use his evaluation of the student’s understanding of the assignment and of the submitted solution as evidence of cheating. Violations will be reported to the Honor Council for consideration for punitive action.

By CS Dept. policy, students found to be in violation of this rule will, at the very least, receive a failing grade in the course and may be subject to stiffer penalties. Students who contribute to violations by sharing their code/designs with others are subject to the same penalty. *Students are expected to use standard Unix protection mechanisms (`chmod`) to keep their assignments from being read by their classmates. Failure to do so will result in grade penalties.*

This policy is *not* intended to prevent students from providing legitimate assistance to one another. Students are encouraged to seek/provide one another aid in learning to use the operating system, in issues pertaining to the programming language, or to general issues relating to the course subject matter. Student discussions should avoid, however, explicit discussion of approaches to solving a particular programming assignment, and under no circumstances should students show one another their code for an ongoing assignment, nor discuss such code in detail.

4.1 Grading:

Assignments: 30%

Midterm Exam: 30%

Final Exam: 40%

It is my general policy that, should a student perform significantly better on the final than upon the midterm, or should a student have one project grade that is significantly lower than the rest, to waive that single low grade (adjusting the percentages of the remaining grades accordingly).

5 Course Outline:

In addition to the chapters listed below, additional readings may be made available via the course web page.

Date	Topics	Chapters	
		Fowler	Horstmann
8/28	Course Overview	1	
9/30-9/6	Classification	2	1, 5 ^a
9/11,13	Classes and ADT's	7	2,3,4
9/18,20	Class Relationships	4, 5, 7, 11	5
9/25,27	Inheritance & Dynamic Binding		7,8
10/2,4	Inheritance in Java		9,10
10/9,11	Use Cases	3,6	
10/16	Multiple Inheritance		18
10/18	Midterm Exam		(in class)
10/23,25	Persistence & Memory Mgmt		13,14
10/30-11/1	Polymorphism		15
11/6,8	Functors, Iterators and other "helper" classes		17
11/13,15	Patterns & Frameworks	8	19
11/20,27,29	Multi-Threading		
12/4	Limitations of OO		10
12/6,	reserved for schedule slippage, or review		
12/11	Final Exam, 3:45-6:45		

^a2. Up to "Recognizing Class Relationships".

Other dates of interest:

9/5 Last day to add classes or to drop with tuition refund

10/24 Last day to withdraw without a permanent "WF" on student transcript.