CS 330: Syllabus (Fall 2025)

Sarah Hosni

Last modified: Aug 18, 2025

Contents:

1 Basic Course Information

The course schedule and website are located at https://www.cs.odu.edu/~tkennedy/cs330/shosni/latest.

1.1 Catalog Description

Laboratory work required. The techniques and idioms of object-oriented programming in C++ and Java. Methods of object-oriented analysis and design with the Unified Modeling Language. Multi-thread programs, synchronization, and graphic user interfaces

1.2 Overall Description

Throughout this course we will discuss various best practices and object-oriented design patterns. As part of this journey… we will endeavor to do the following:

1. develop the skill set to learn new languages *on the fly* based on existing knowledge
2. leverage existing knowledge to reason about code written in an unfamiliar language
3. analyze design patterns that are common between languages
4. identify *emergent* properties of object-oriented code

1.3 Prerequisites

The prerequisites for this course are:

- CS 250, Programming with C++; or CS 251, Programming and Problem Solving in Java; or CS 253, Transfer Credit for Programming with Python
- CS 252, Introduction to Unix for Programmers

Note that, if it has been some time since you took CS 251, or if you received weak grades in the course, you may need to do some review work to prepare for this class.

If you are a transfer student who took equivalent courses elsewhere, you would do well to review the material on those course websites and look for topics that may not have been covered in your prior courses, because course "equivalence" is often a very rough approximation. **Pay particular attention to the material on design, testing, and debugging**, as these are often given short shrift at other institutions.

Either way, if you need to review any of the prerequisite topics described below, the time to do so is early in the semester, *before* you need it to understand the lectures or are called to use it in assignments.

1.3.1 General Programming Knowledge

Students should be familiar with certain basic programming techniques that are largely independent of any specific programming language:

- using editors, compilers and other basic software development tools.
- basic software design (i.e., stepwise refinement and top-down design)
- software testing, including the use of scaffolding code (stubs and drivers), selection of test cases for black-box testing, and head to head testing.
- debugging, including the use of debugging output, the use of automatic debuggers to set breakpoints and trace program execution, and the general process of reasoning backwards from failure locations to the faulty code responsible for the failure.

1.3.2 C++ Knowledge

No prior knowledge of C++ is assumed. C++ will be discussed as a point of comparison (i.e., for perspective and context). However, you will **not** be tested on your ability to write C++ code.

1.3.3 Java Knowledge

I will assume that you are familiar with the basics of Java. This includes:

- the various Java statements and control-flow constructs,
- the built-in data types,
- the use of arrays, pointers, pointers to arrays, and linked lists,
- the use and writing of functions, and
- the basic use of structs and/or classes for implementing abstract data types.

We will discuss how to write Java code, along with best practices.

1.3.4 Python 3 Knowledge

In general, CS students at the 300 level should be able to pick up new programming languages with only moderate effort.

Prior knowledge of Python is neither expected nor assumed.

We will discuss how to write and design Pythonic (idiomatic) Python code.

1.3.5 Rust Knowledge

Prior knowledge of Rust is neither expected nor assumed.

We will discuss how to write Rust code for context and perspective (with an emphasis on design patterns).

1.3.6 Unix/Linux

All students in the course will receive accounts on the CS Dept. network, and knowledge of how to work with the Linux servers is part of the course prerequisites. This course does not require familiarity with shell scripting. All other topics in CS 252 are required.

> Some assignments will require (or, at least, be simplified by) the use of software available on the Linux servers.

1.3.7 General Computer Literacy

You will be studying techniques in this course for preparing professional-quality software documentation. The key embedded word in “software documentation” is “document”. Students taking this course should be able to use word processors and other common tools to produce good quality documents, including mixing text and graphics in a natural and professional manner.

1.3.8 Web Section

- CRNs 17646, 17648, and 17649

This course is online and has no regularly scheduled lectures.

1.4 Instructor

General Information	
Instructor	Sarah Hosni
Office	Dragas

Phone #	
Email	shosni@odu.edu

Important: All email related to this course should have the phrase **CS 330** somewhere in the subject line. This flags your message in my mailbox for faster attention.

1.4.1 Office Hours

I plan to use Zoom for my office hours twice a week.

- Tuesdays: 12:00 AM to 01:00 PM
- Thursdays: 11:00 AM to 12:00 PM

My office hours are typically 15-30 minute Zoom meetings. Please email me (shosni@odu.edu) to schedule a time slot when needed.

Questions about grades, how to solve assignments and other graded activities must be sent to shosni@odu.edu.

For more discussion on course communications, please refer to the Communications policy.

2 Required Materials

2.1 Textbooks

Most assigned reading will be from books and other sources available on-line.

Required:

- **There is no new required textbook for this course.**
- You should already have a Java text that covers classes and inheritance. (e.g., your CS251 textbook).

2.2 Supplemental Materials

All supplemental resources and materials will be available in Canvas.

2.3 Technology Requirements

2.3.1 Computer Accounts

Students will need an account on the CS Dept. Linux network to participate in this class. This account is unrelated to any University-wide account you may have from ODU's Information Technology Services (ITS).

If you have had a CS Unix account in the recent past, you should find it still active with your login name, password, and files unchanged. If you have had an account and it has not been restored, contact the CS Dept systems staff at root@cs.odu.edu requesting that it be restored.

If you do not yet have such an account, go to the CS Dept. home page and look for "Account Creation" under "Online Services". All students in this course are responsible for making sure they have a working CS Unix account **prior to** the first assignment.

2.3.2 Compilers and Interpreters

The "official" environment in which students' programming assignments will be evaluated is defined by our Dept. Linux servers. It is the student's responsibility to be sure that their code compiles and executes using the compilers and run-time environment provided there. As of this writing, the compiler and run-time environment versions used are…

- C++: g++ 11.4.0
- Java: openjdk version "21.0.8"
- Python: 3.11 & 3.13

You may want to install uv. It will handle the creation of Python virtual environments and the installation of any required libraries. Most Python examples can be run with…

uv run [program].py

…in place of…

python3.11 [program].py

3 Course Objectives

This course will explore the techniques of object-oriented programming, analysis, 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 object-oriented programming languages and will learn the constructs that differentiate such languages from others. This course will explore the idioms and styles of object-oriented programming in Java, Python, and Rust with emphasis upon how these contribute to reusable software components.

Students will learn how to use object-oriented techniques in support of programming. In particular, students will be introduced to the process of object-oriented analysis as a means of understanding an unfamiliar problem domain.

Students will learn to build and use models, expressed in the Unified Modeling Language (UML) to codify and evaluate that understanding and to evolve system requirements. Students will learn how to use those models to facilitate a smooth transition from analysis to design and from there to implementation.

4 Course Schedule

The course schedule is available in Canvas under *Modules* and *Calendar*. An alternate schedule/outline can be accessed at https://www.cs.odu.edu/~tkennedy/cs330/shosni/latest/Directory/outline/index.html.

5 Grading

Category	Weight
Assignments	40%
Module Review Quizzes	50%
Final Exam	10%

Example 1: Grading Schema

Percent	Letter Grade	4pt Value
≥85	A	4.0

≥80.5	A-	3.7
≥74.5	B+	3.3
≥70	B	3.0
≥60	B-	2.7
≥59.5	C+	2.3
≥55	C	2.0
≥50.5	C-	1.7
≥44.5	D+	1.3

≥40		D	1.0
N/A		D-	0.7
<39	<	F	0.0

Your *Final Course Grade* will be computed by both:

- dropping two Assignment grades
- dropping two Module Review Quiz grades

5.1 Assignment Drop

All final grade computations are based on 4pt grades as listed under *Grades* in Canvas. The computation for dropping assignment grades is

```
((sum of all assignment grades) - (lowest two assignment grades)) / (number of assignments - 2)
```

5.2 Assignments

Assignments for this course will include *programming assignments*, which must be done on an individual basis.

5.2.1 Assignment Grading

Assignments will be turned in through Canvas (unless otherwise noted in an Assignment Prompt).

Most of the assignments will be graded by an automatic grader. The results will be listed in either a rubric or feedback for your submission in Canvas.

Unless the assignment explicitly states otherwise, you may submit an unlimited number of times. You may NOT submit after viewing the sample solution.

5.2.2 Auto-Grader & Testing

Test driven development is a topic of particular import–not only in academia, but in industry.

You will be expected to make use of Blackbox Testing. This is a topic of particular import. Blackbox testing is covered in CS 250–a prerequisite for this course. You will also need to make use of white-box testing and unit testing.

All tests are designed by me–the instructor. The Auto-Grader runs tests that I use to evaluate my solution. These tests evaluate mechanics of import–e.g., dynamic binding and function overloading.

Be systematic in all changes to your assignment solution and modifications to your tests.

- Do not haphazardly make changes to an assignment and resubmit hoping for a better grade.
- Treat each submission attempt as your final submission.
- Ask for guidance before each subsequent submission.

5.3 Exams

The Final Exam will be administered online via Canvas.

5.4 Incomplete Grades

A grade of “I” indicates assigned work yet to be completed in a given course, or absence from the final examination, and is assigned only upon instructor approval of a student request. The “I” grade may be awarded only in exceptional circumstances beyond the student’s control. The “I” grade becomes an “F” if not removed by the day grades are due for following term based on specific criteria: Incomplete, Withdraws and Z grades..

6 Expectations

6.1 Assignment Expectations

It is my expectation that you have completed approximately 70% of each assignment once half the allotted time has passed. For a two week assignment this would be one week. This will allow you sufficient time to address any issues, refine your testing process, and discuss your solution with me during my office hours.

I expect every student to discuss each assignment with me at least once.

6.2 Recorded Lectures & GitHub Repositories

There are two (2) sets of examples referenced in this course.

1. **CS 330 - Examples**
   Recorded lectures are included in the course outline. The full set of recordings (and associated example code) is available on GitHub.
2. **Foundational Examples**
   These examples are not usually referenced in the course outline. This is a collection of small examples the demonstrate select concepts and mechanics used in larger course discussions (e.g., command line arguments). There are usually a C++, Java, Python, and Rust versions of every example. These examples are available on GitHub.

Download the repositories (either by cloning the repos or downloading zip files) to a location of your choosing.

Where you save the example code is up to you. Choose somewhere convenient (i.e., somewhere that you can open, edit, and run the code).

7 Course Policies

7.1 Due Dates and Late Submissions

Late papers, assignments, projects, and make-up exams will not normally be permitted.

Exceptions will be made only in situations of unusual and unforeseeable circumstances beyond the student's control, and such arrangements must be made prior to the due date in any situations where the conflict is foreseeable.

"*I've fallen behind and can't catch up*", "*I'm having a busier semester than I expected*", or "*I registered for too many classes this semester*" are *not* grounds for an extension.

Extensions to due dates will not be granted simply to allow "porting" from one system to another. "*But I had it working on my home PC!*" is not an acceptable excuse.

7.2 Academic Honesty

Everything turned in for grading in this course must be your own work. If an assignment is **explicitly** described as a team assignment, it must be the work of the team members only.

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 Office of Student Conduct & Academic Integrity for consideration for possible punitive action.

Students who contribute to violations by sharing their code/designs with others may be subject to the same penalties.

- 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, at the very least.

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.

> Students 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.

7.2.1 Use of Online Resources

You may **not** post details of course assignments, projects, or tests at online Forums, Bulletin Boards, Homework sites, etc., soliciting help.

You may use information that you have not solicited but have located, subject to the following restrictions:

- Just as when writing a paper, if you use someone else's ideas, you must cite your sources appropriately. Within code, such citations appear in comments.
  Example:

```
⋮
double x = 23.0;
double xsqrt = sqrt(x);
// Search algorithm based upon code by S Zeil at
//
https://www.cs.odu.edu/~zeil/cs361/latest/Public/functionAnalysis/index.html#orderedsequential
search
int loc = 0;
while (loc < arraySize && numbers[loc] < xsqrt)
⋮
```

-
- Just as when writing a paper, if you use someone else's words (code), you must cite your sources appropriately **and** mark the quoted text. Within code, such citations appear in comments.
  Example:

```
⋮
double x = 23.0;
double xsqrt = sqrt(x);

// Begin quoted code from  S Zeil at
//
https://www.cs.odu.edu/~zeil/cs361/latest/Public/functionAnalysis/index.html#orderedsequentialsearch
int loc = 0;
while (loc < listLength && list[loc] < searchItem)
{
   ++loc;
}
// End quoted code

⋮
```

-
- Failure to appropriately cite any such "found code" will be taken as evidence of plagiarism.
- The overall principle stated in the first sentence of this section remains in effect. "Everything turned in for grading in this course must be your own work." If the bulk of your assignment, project, test answer, etc., are copied, even with appropriate citation, to the degree that, in the judgment of the instructor, you have not demonstrated your own knowledge of the course material, you will receive a zero for that submission.

7.2.2 ChatGPT and Similar Tools

ChatGPT and similar tools are not prohibited. However, any use must be properly cited in a form similar to all other online resources.

8 University Policies

8.1 Code of Student Conduct and Academic Integrity

The Office of Student Conduct & Academic Integrity (OSCAI) oversees the administration of the student conduct system, as outlined in the Code of Student Conduct. Old Dominion University is committed to fostering an environment that is: safe and secure, inclusive, and conducive to academic integrity, student engagement, and student success. The University expects students and student organizations/groups to uphold and abide by standards included in the Code of Student Conduct. These standards are embodied within a set of core values that include personal and academic integrity, fairness, respect, community, and responsibility.

8.2 Honor Pledge

By attending Old Dominion University, you have accepted the responsibility to abide by the Honor Pledge:

> I pledge to support the Honor System of Old Dominion University. I will refrain from any form of academic dishonesty or deception, such as cheating or plagiarism. I am aware that as a member of the academic community it is my responsibility to turn in all suspected violations of the Honor Code. I will report to a hearing if summoned.

8.3 University Email Policy

Reformatted to follow https://ww1.odu.edu/about/policiesandprocedures/computing/standards/11/02.

8.3.1 Student Email

With the increasing reliance and acceptance of electronic communication, email is considered an official means for University communication. Old Dominion University provides each student an email account for the purposes of teaching and learning, research, administration, and service. It is important that all students are aware of the expectations associated with email use as outlined in the Student Email Standard.

8.3.2 Email Account Activation

It is the responsibility of every eligible student to activate MIDAS, the Monarch Identification and Authorization System, in order to obtain email access.

8.3.3 Expectations Regarding Use of Email

The email account provided by the University is considered to be an official point of contact for correspondence. Students are expected to check their official e-mail account on a frequent and consistent basis in order to stay current with University communications. Mail sent to the ODU

email address may include notification of University-related actions, including academic, financial, and disciplinary actions. For more information about student email, please visit Student Computing.

8.3.4 Educational Uses of Email

University offices and instructors cannot validate that a communication coming by email is from an ODU student unless it comes from a valid ODU email address. If students send mail from non-ODU email accounts (e.g., Hotmail or Yahoo), faculty and staff are not obligated to respond and may request that official e-mail accounts be used.

8.4 Withdrawal

Enrollment in this course indicates your acceptance of its teaching focus, requirements, and policies. Please review the syllabus and the course requirements as soon as possible. If you believe that the nature of this course does not meet your interests, needs or expectations, if you are not prepared for the amount of work involved – or if you anticipate assignment deadlines or adherence to course policies will constitute an unacceptable hardship for you – you should drop the course by the drop/add deadline, which is listed in the ODU Schedule of Classes. Visit the Office of the University Registrar for more information.

8.5 Educational Accessibility

Old Dominion University is committed to ensuring equal access to all qualified students with disabilities in accordance with the Americans with Disabilities Act. The Office of Educational Accessibility (OEA) is the campus office that works with students who have disabilities to provide and/or arrange reasonable accommodations.

- If you experience a disability which will impact your ability to access any aspect of my class, please present me with an accommodation letter from OEA so that we can work together to ensure that appropriate accommodations are available to you.
- If you feel that you will experience barriers to your ability to learn and/or testing in my class but do not have an accommodation letter, please consider scheduling an appointment with OEA to determine if academic accommodations are necessary.

The Office of Educational Accessibility is located at 1021 Student Success Center and their phone number is (757) 683-4655. Additional information is available at the OEA website