1. Basic Course Information

1.1 When and Where

This course is online and has no regularly scheduled lectures. There will be a limited number of attendance-optional network conferences (announced on Blackboard).

1.2 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 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 in C++ and Java, 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.

1.3 Textbooks

Required:

  - You may choose to buy the non-custom edition of the course text online
- Any reasonable C++ text that covers classes and inheritance (e.g., your CS250 or 333 textbook)

Optional:


2. Communications

2.1 Instructor

<table>
<thead>
<tr>
<th>Thomas Kennedy</th>
<th>Dragas 1100H</th>
</tr>
</thead>
<tbody>
<tr>
<td>(757) 683–7725</td>
<td><a href="mailto:tkennedy@cs.odu.edu">tkennedy@cs.odu.edu</a></td>
</tr>
</tbody>
</table>

Make sure to include the course name **CS330** in the subject line of any email related to this course.

Office Hours

My general office hours are available at [http://www.cs.odu.edu/~tkennedy/fall16](http://www.cs.odu.edu/~tkennedy/fall16). Instructions for scheduling a formal appointment are listed on the same page. Note that my office hours include both live and web-conference based appointments.

General questions about course content and reports of website problems should normally be asked in the Forums on Blackboard.
Questions about grades, how to solve assignments and other graded activities should be send to tkennedy@cs.odu.edu.

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

3. Course Prerequisites

The prerequisites for this course are:

- **CS 250**, Problem Solving and Programming II, or **CS 333** Problem Solving and Programming in C++
- **CS 252**, Introduction to Unix for Programmers

Note that, if it has been some time since you took CS 250 or 333, or if you received weak grades in them, then 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.

Students who have successfully completed those prerequisites should have acquired the following knowledge and skills:

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.

3.2 C++

I will assume that you are familiar with the basics of C++, including

- the various C++ 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 classes for implementing abstract data types.

3.3 Java

No prior knowledge of Java is assumed.

• In general, CS students at the 300 level should be able to pick up new programming languages with only moderate effort. Because Java is closely related to C++, this is particularly true of students moving from C++ to Java.

• If you believe that you need a more structured aid to learning Java, the 1-credit course CS382 presents the basics of the language in a self-paced form. In fact, much of the course material for CS382 will be listed as required reading for this course.

• When the CS 330 Outline page lists a section of 382 for readings, students are expected to both read the 382 lecture notes and to do the associated 382 labs. You will not be expected to do the 382 assignments or to take the 382 exam.

You may, if you wish, choose to register for CS382 and earn credit for it. In that case, you will be doing largely the same set of readings, but will then be obligated to do the assignments and exam for that course.

• CS 330 will pick up with Java topics more advanced than are currently covered in 382, including multi-threading and the development of GUIs and in general will explore how to use Java in a truly Object-Oriented style.

3.4 Unix

All students in the course will receive accounts on the CS Dept. Unix 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 the use of software available only on the Linux servers. Others may require (or, at least, be simplified by) use of the X windowing system.

3.5 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.

4. Assignments

Assignments for this course will include *programming assignments* (in C++ and Java), which must be done on an individual basis, and *design assignments*, which may be done in small teams.
4.1 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.

4.2 Assignment Grading

Assignments will be turned in through the CS submission system, rather than through Blackboard–more information is available here. Most of the assignments will be graded by an automatic grader. The results will be sent to your CS email account. Unless the assignment explicitly states otherwise, you may submit a total of three times per assignment; the instructor will take the last of the marks, although you may request that your score be “rolled back” to an earlier one. You may NOT submit after viewing the sample solution.

4.3 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 whitebox testing and unit testing. These are topics discussed in CS 250. Take the first week to review these concepts. You will find the necessary materials on the CS 333 course site.

Difficulties with the tests performed by the Auto-Grader should be addressed after the first assignment submission. If a program fails a test, there is usually an edge—or corner—case for which you—the student—did not account.

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.

4.4 Computer Accounts

Students will need an account on the CS Dept. Unix network to participate in this class. This account is unrelated to any University-wide account you may have from the 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.
4.5 Compilers

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 versions used are

- C++: g++ 4.8.1
- Java: 1.7.0_55

5. Review Sessions

I conduct Review Sessions at selected times during the semester–approximately once every two weeks. Instructions regarding how to attend live sessions and view recorded reviews are available here. Attendance for live sessions is not mandatory. You are expected to view the recordings of these Review Sessions.

6. Course Policies

6.1 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.

6.2 Academic Honesty

Everything turned in for grading in this course must be your own work. Some assignments may be done by small teams, in which case the submitted material must be the work of only those team members. Such assignments will be clearly marked as team assignments. Where teams are permitted, specific guidelines will be given in the online assignment description. In the absence of any such explicit statement, an assignment must be performed by a single individual.

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.

Students who contribute to violations by sharing their code/designs with others may be found to be in violation of this policy. This includes showing material to other students in person and posting code and designs in any public area, whether physical or on the internet. Students are expected to use standard Unix protection mechanisms (chmod) to keep their assignments from being read by their classmates.
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. The same guideline applies to discussions, whether face-to-face or on-line, with anyone other than the course instructor and TAs – general aid on the subject matter of the course is OK. Specific discussions of solutions to any graded activity are forbidden.

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.

Violations will be reported to the Office of Student Conduct and Academic Integrity.

### 6.3 Grading

<table>
<thead>
<tr>
<th>Assignments:</th>
<th>40%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Midterm Exam:</td>
<td>25%</td>
</tr>
<tr>
<td>Final Exam:</td>
<td>35%</td>
</tr>
</tbody>
</table>

It is my general policy that, should a student perform significantly better (a rise in class rank of at least 1.25 standard deviations) 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).

For the truly curious, some [further information](#) on my approach to grading is available.

### 6.4 Exams

The midterm and final exam will be administered online.

### 7. 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.
http://www.odu.edu/educationalaccessibility/