Syllabus

Instructor information

Course Instructor: Kristina Striegnitz (Call me "Kristina.")

Email: striegnk@union.edu

Office: Steinmetz Hall 227

Need help? You can

- use the Piazza forum
- talk to our CS helpdesk tutors
- come to see me during office hours (or make an appointment)

See Nexus (`https://nexus.union.edu`) for details.

1 What this class is about

CSC 151 is the third core course in the CS major curriculum. Its purpose is to refine your programming and problem-solving skills to gain a sense of "programming maturity". You've gained a lot of powerful tools in your study of programming constructs like loops, conditionals, and arrays. And you have worked in the object-oriented programming world of Java. Now we focus on using those tools in a logical, efficient way to create modular, reusable programs instead of cumbersome, haphazard ones.

We will again be programming in Java, and while we'll be learning some new features of that language, our focus will be on good design, which is applicable to any language.

By the end of the course, you should be proficient in the following:

- top-down design
- development of modular, well-organized, reusable, understandable programs
- a variety of debugging and testing techniques
- space/time trade-offs; other trade-offs between various approaches to solving problems and storing data
- analysis of space/time complexity of an algorithm and its comparison to other algorithms

- a variety of data structures

I hope to at least cover the following topics:

- code reuse through Java interfaces and generics
- mathematical comparison of algorithms and implementations
- data abstraction and information hiding
- linked lists and efficiency trade-offs
- stacks, queues, priority queues
- trees
- graphs
- hashing

2 Prerequisites

You need a C- or better in CSC 120 (Programming on Purpose) to be eligible to take this course. You should already be familiar with Java. See your instructor immediately if you don't meet these prerequisites.

3 Software, Textbook, and Other Resources

If you want to work on your own computer, please make sure that you have VS Code (IDE) and a Java 17 development kit. See Nexus for links to download and install.

You also have access to the CS department's labs:

- Olin 107
- ISEC 051 (aka Pasta lab)
- CS resource room (Steinmetz Hall, 209A, just down the hall from my office)

We will use an online and interactive textbook from Runestone Academy that is available to you for free. You can find a link and registration instructions on Nexus.

An additional book that is good, but not required, is: Michael Main, Data Structures and Other Objects Using Java., 4th ed, Pearson, 2012. This is the book that this course has used for many years. However, it has gotten very expensive.

4 What do I need to do to succeed?

Participate actively. It is important that you engage with the material. Just sitting in class, is not enough. You have to think about and work through examples and problems. In class you will sometimes work in small groups and sometimes in full class discussions. In order to get the most out of these discussions, you should participate actively, even if you don't know the answer or understand everything completely. Showing that you are thinking is more important than being able to answer all questions perfectly.

Complete the assigned readings and activities. I will assign readings, worksheets, and small programming exercises throughout the course. These are an opportunity for you to deepen your understanding of the material and to critically evaluate for which parts of the material you may need some clarification. Complete them as I assign them and follow up on any questions during office hours or at helpdesk. Note: Cramming the day before the exam does not work very well for this course.

Check the Nexus page and the Piazza class forum regularly. Where regularly means at least once a day. This is important to not miss any assignments and announcements.

Get started on the programming projects early. There will be several bigger programming projects throughout the term. These projects will allow you to see the concepts you are learning about in action within the context of a somewhat larger project. You should expect to need multiple work sessions to finish these projects. So make sure to get started on them as soon as they are assigned so that you have enough time and have time to ask questions if you get stuck.

Stay focused. There's a lot to learn when it comes to program design, so your job is to stay engaged. If you're using a laptop/tablet to take notes, that's fine, but class time is for class work. Using your computer for other things like other classes' work, checking social media sites, reading email, or catching the latest ESPN highlight video is not permitted. This policy extends to cell and smart phones, too. Phones should be in silent mode and out of sight (mine *and yours*).

Class goes by quick. Stay focused. If you're done with an in-class exercise, don't allow yourself to get bored. Extend the exercise. Or help out your neighbor.

Ask questions and seek help. By now, you have probably noticed that in computer science classes one thing builds on the next. We start with simple things and build on them and combine them to more and more complex ideas. It is, therefore, important that you don't fall behind. Keep up with all readings

and assignments and, most important of all, if at any point you notice that something is not quite clear, ***ask.*** It is completely normal to have some questions sometimes. So don't hesitate to ask. I expect to see all of you in my office hours at some point during the term. And some of you might decide to attend office hours every week, and that is fine, too.

Here are some ways in which you can ask for help:

- Use our class forum.
- Come to office hours.
- Go to helpdesk to get help from more advanced CS students. Helpdesk will start in week 2.

5 How will my work in this course be assessed?

The following components will contribute to your final grade for this class with roughly the indicated weight.

- projects: 40%
- midterm exam: 20%
- final exam: 25%
- engagement and practice: 15%

Note that you must get a C- or better in this course in order to take any other course that requires Data Structures as a prerequisite.

Exams. There will be a midterm exam and a final exam during exam week. Check Nexus for the dates. The final will be cumulative. On exams, you will be responsible for all material covered in the readings, lectures, and projects.

Programming projects. Several larger programming projects will be assigned to reinforce the concepts discussed in class. These projects are where you'll get the best chance to practice good design principles. Programming projects are due at the end of the day on their due dates. (See the class schedule linked on Nexus for the planned due dates.) You have three "extension tokens" for this class. Each token will extend a project deadline by 24 hours. You can use the three tokens for three different projects, or you can use two or even all three tokens on the same project, giving you an extension of 48 or 36 hours. **After you have used up your three tokens, no late submissions will be accepted.**

Engagement and practice. Demonstrate engagement with the class and the class material by attending class, coming to class prepared, participating actively in class discussions and group work, being active on the class forum, and completing the textbook readings and embedded exercises as well as any additional homework assignments that I might assign.

6 Academic Integrity

6.1 General Statement

Union College recognizes the need to create an environment of mutual trust as part of its educational mission. Responsible participation in an academic community requires respect for and acknowledgment of the thoughts and work of others, whether expressed in the present or in some distant time and place.

Matriculation at the College is taken to signify implicit agreement with the Academic Honor Code, available at `http://honorcode.union.edu`. It is each student's responsibility to ensure that submitted work is his or her own and does not involve any form of academic misconduct. Students are expected to ask their course instructors for clarification regarding, but not limited to, collaboration, citations, and plagiarism. Ignorance is not an excuse for breaching academic integrity.

What that means for this course

Most importantly that means that any work you turn in has to be your own. In particular, you should not look up solutions to problems (or even parts of problems) online. Of course, you are allowed to use any online resources that are linked from the course Nexus site, but you should not search for (partial) solutions elsewhere on the Internet.

You also should not ask other students in this class (or other people) for solutions to assignments. To make sure that no accidental plagiarism happens you should not look at the code that other students have written before the due date of an assignment or show your code to other students before the due date.

Following these guidelines is not just a matter of obeying the honor code. It's also how you will learn. You can only learn to program by doing it. And a big part of programming is debugging, i.e. analyzing and fixing errors in your code. So, struggling with problems is a normal part of the process. And it is where a lot of the learning happens in CS. (However, sometimes you will need some help getting unstuck. That is normal, too. In those cases, see the information above on where to get help.)

Finally, here are some ways in which you *are* allowed to discuss the course material with other students in the class and other people:

- Once you've struggled with something on your own and you're still getting nowhere, ask for help! Email and visit me, go to the helpdesk, and, yes,

ask your fellow students! They can't show you code, but they can sure write down examples, show you demos, and explain things.

- It's ok to write down an algorithm in English (pseudocode) together. Then go off by yourself to implement it into code.
- It's ok to read and write code together that is not part of an assignment, but that helps demonstrate the concept of what you're being asked to do. Go through an example from class or from the book together, or come up with your own examples. There's no better way to understand something than trying to think up examples in order to teach it to someone else. Try it!

Your goal for any discussion about an assigned problem or programming project should be that you come away with a better understanding of the problem and of possible ways of approaching the problems, so that you can then try out these approaches on your own. You should never leave such a discussion with just an answer, without an understanding of how to arrive at that answer.

You're going to write and see a lot of code in this class. A good question is: what sources can you legally take code from for your projects? It is ok to reuse code ...

- that I hand out in class
- that is part of a demo that I leave on Nexus
- that is part of a homework assignment

It is NOT ok to reuse code ...

- that is part of someone else's homework or project
- that is on the Web/Internet
- that someone or something else wrote for you (i.e. no code written by a friend, a tutor, an AI, ...)
- that is in textbooks (except for our Runestone textbook and other readings that I make available on Nexus)

Here's the bottom line: except for the above, you have to write all the code yourself, from scratch. For everything you turn in, you must explicitly cite any source (like a web page tutorial or a helpdesk person) that you use to help complete an assignment. Again, this is similar to writing an English paper; if you use a quote or material from someone else, you have to give credit where credit is due. Otherwise you are inappropriately plagiarizing or borrowing ideas. You don't have to cite help from me, though.

We have an honor code and I trust y'all to follow it. Read up on it at `http://honorcode.union.edu`. All suspected violations will be reported to the Honor Council chair and Dean of Studies.

7 Special Arrangements

Any student with a documented learning disorder is welcome to talk to me privately about options for the completion of exams and assignments.

Similarly, talk to me as soon as possible if you know that you are going to miss classes or if there are circumstances that are keeping your from completing your work in a timely manner, so that we can discuss how you are going to make up the work.

8 Counseling Resources

As a college student, there may be times when personal stressors interfere with your academic performance and/or negatively impact your daily life. If you or someone you know is experiencing mental health challenges at Union College, please contact the Counseling Center. Please call 518-388-6161 or email hotalinm@union.edu or daym@union.edu to schedule an appointment.

In a crisis situation, or after hours, contact Campus Safety at 518-388-6911. The National Suicide Prevention hotline also offers a 24-hour hotline at 800-273-8255.