Problem Solving Via Object-Oriented Programming

CS10: Winter 2015

About the Course

CS 10 is the second prerequisite for the Computer Science major and minor. It covers fundamental concepts in computer programming, in two broad areas: object-oriented programming and data structures. In particular, we'll use object-oriented programming to implement data structures. We'll understand why the data structures work and analyze their efficiency. We'll also lean heavily on abstraction, so that we can separate the *what* from the *how*.

Now, data structures can be cool on their own, but they're even better when we apply them to solve interesting problems. That's what we do in CS 10.

Here's the ORC description of the course:

> *Motivated by problems that arise in a variety of disciplines, this course examines concepts and develops skills in solving computational problems. Topics covered include abstraction (how to hide details), modularity (how to decompose problems), data structures (how to efficiently organize data), and algorithms (procedures for solving problems). Laboratory assignments are implemented using object-oriented programming techniques.*

Instructor

Travis Peters
Office: 209 Sudikoff
traviswp@cs.dartmouth.edu

Office Hours

In 209 Sudikoff, January 5 – March 12:

- Monday, 1:00 – 3:00 pm
- Thursday, 1:00 – 2:00 pm
- Friday, 2:00 – 4:00 pm

In the weeks where we do not hold the x-hour, this time (Thursday, 12:00 – 1:00 pm) will be used as an extra office hour.

If you'd like to meet with me outside of office hours, please contact me and we'll set up a time. Additionally, I hang around my office quite a bit so if my office door is open, feel free to drop in.

Course Staff

Graduate Teaching Assistant:

Yinan Zhang
Office: 220 Sudikoff
Email: yinan.zhang.gr@dartmouth.edu
Office hours: Tuesday, 3 – 4pm | Friday, 2 – 5pm

Undergraduate Section Leaders:

Caleb An ’15
Jessica Fan ’17
Naho Kitade ’16
Brendan Krimsky ’17
Brandon Mader ’17
Liane Makatura ’17
Steve Nugent ’15
Richard Palomino ’15
Charley Ren ’17
Katy Sprout ’17
Xinran Xiao ’15
Joanne Zhao ’16

Prerequisites

In order for you to take this course, you must have fulfilled one of the following prerequisites:

- Taken Computer Science 1
- Taken Engineering Sciences 20
- Taken the Computer Science A AP Exam and gotten a 4 or 5
- Passed the department's placement exam

If you do not have any of the above prerequisites, you should see Travis as soon as possible.

Questionnaire

I need to know a little about you. Please fill out this questionnaire as soon as possible.

NOTE: We are trying out the use of Google Forms this term. Let me know if you have any problems with the form and I'll provide a copy of the form in a different format that you can fill out.

Lectures

- **Location**: LSC 100 (Oopik Auditorium)
- **Times**: MWF 10:00–11:05 am. x-hour is Thursday, 12:00-12:50 pm.

X-Hours

We will occasionally use x-hours, so keep the x-hour times available for this course.

Also, as mentioned above, in the weeks where we do not hold the x-hour, this time will be used as an extra office hour.

Questions

We will all enjoy the experience more if you ask questions in lecture. I actually believe that asking questions and interacting in this way is one of the primary benefits of having/attending lectures. I understand that it can be intimidating to raise your hand for a question in a fairly large lecture, but please do not hesitate to ask questions. Don't worry about what anyone thinks of you for asking questions. Chances are that if you have a question, then at least one other student—and possibly many more—has the same question. You're doing the other students a favor by asking!

As a side note, I used to have a fear of asking questions in large lectures. I often thought my questions were "stupid" and would avoid asking my questions publicly at all costs. I, however, hold the belief that ***there is no such thing as a stupid question*** – so at the very least know that you are in good company and that, in this class, you've got an instructor that enjoys and appreciates your questions.

If you still are reluctant to ask a question in lecture, even after all the encouragement above, feel free to write it down and ask me, the graduate TA, or any section leader after class, by email, in office hours, or in lab hours. We want to help you!

Occasionally, I might elect to answer a question later or after lecture. That will usually be because the answer to the question is tangential to the material we're covering, or simply because I am pressed for time. But please feel free to ask; I'll let you know nicely if I cannot answer the question then and there.

Laptops and Phones

If you bring a mobile phone, please make sure that it does not audibly ring during lecture. I've heard of professors that will answer phones that ring during class in an effort to embarrass students and surprise their caller. I like this idea in theory and I'm not above trying such a tactic...

Also, laptops cause distractions. If you want to distract yourself by Facebooking, shopping, blitzing, etc. during the lecture, that's your business. The problem is that it's not only *your* business. When you use your laptop for non-course-related tasks, you distract those around you as well. And that's not fair.

I have heard of various ways of dealing with the distractions that laptops cause during lecture: laptop zones, laptop permission forms, and the honor system. None worked. I also have a hard time saying "no laptops" since I know that some people do use them to take notes and follow along with live coding examples, etc. Here's what I propose for this term:

> ***If you are someone that absolutely needs their laptop during lecture and you can't part with it for 3 hours per week, then by***

> ***all means, use your laptop. If you can manage to go through lecture without your laptop, I believe you will learn more and ultimately take more away from this class. There is recent research that attests to the negative impacts of learning and retention when multitasking. It has also been shown that writing notes by hand rather than on a laptop engages different cognitive processes and has direct (positive) consequences for learning.***

Recitation Section

You are required to attend a one-hour recitation section meeting each week. Attendance is mandatory – section leaders will take attendance.

Short Assignment 0 has the forms you'll fill out so that we can put the recitation sections together. Completing this form will be part of your grade for Short Assignment 0. The recitation section schedule will be determined sometime in the later part of the first week, and your section assignment will then be posted on the Canvas site. The first recitation sections will meet during week 2 (January 12-16).

If you say you are available during a one-hour slot, I expect you to honor this commitment. Please take this expectation into account as you make other commitments on and off campus this term.

We recognize that occasionally (which means "not on a regular basis"), you might not be able to attend the recitation section to which you have been assigned. In that event, please find another section to attend that week. Make sure that your regular section leader knows that you'll be attending another section, and make sure you get permission *in advance* from the section leader whose section you'll be visiting.

Grading

- Programming Drills: 10%
- Short Assignments: 10%
- Lab Assignments: 35%
- Exams: 45%

There will be a midterm exam during week 6. If a scheduling conflict prevents you from taking the midterm exam at the scheduled time, you must let me know one week before the exam.

The final exam is scheduled by the Registrar and will be offered at one time, and one time only: **Friday, March 13, 2015 at 8:00 am**. If you have plans that prevent you from taking the final exam on March 13 at 8:00 am, you have three choices: change your plans, do not take this course, or get a 0 on the final exam. *The only way that you may take the final exam at another time is if you have a documented illness that prevented you from taking the exam at the scheduled time.*

Before each exam, I will post some resources to help you review. There will also be review sessions before each exam either during regular class or the x-hour.

The midterm and final exam count toward 20% and 25% of your final grade, respectively.

There is no fixed scale for assigning letter grades. I expect the median grade for the course to be B or B+, *but I make no guarantees in advance*. The median grades depend on *you*. If your grade is borderline, I may take your attendance at recitation sections into account when deciding which way to go.

Programming Drills

New to CS 10. During your weekly recitation sections you will solve small programming problems. You must, therefore, bring your laptop to recitiation. There will be a total of 8 such problem sets. We will drop your lowest grade. You must attend your own recitiation to receive a grade from your section leader. If you miss a recitation, then there will be no way to submit a solution and you will receive a 0 for that week.

Your section leader will grade your work on a scale of 0 to 2.

- **2** means that you got it right or mostly right.
- **1** means that there were problems with your solution.
- **0** means that either you did not submit a solution, or that what you submitted was a very poor effort.

Short Assignments

Short assignments are relatively brief exercises that you will turn in at the next Monday, Wednesday, or Friday class. In some cases, you might have two classes to work on a short assignment. Short assignments will usually consist of one or two short programs to help you understand some concept being covered.

You will submit short assignments through the Canvas site. They will have a timestamp, and so we will know when you submitted them. They are due at the start of class on the day they are due. The start of class is 10:00 am *sharp*. If your short assignment comes in at 10:01 am, it is late. **Short assignments handed in late will receive no credit.**

Your section leader will grade your short assignments on a scale of 0 to 5.

- **5** means that you got it right, though your section leader might suggest some ways that you could improve your work.
- **3-4** means that you got it mostly right, but there was a few problems with your solution.
- **1-2** means that there were significant problems with your solution.
- **0** means that either you did not submit the short assignment, or that what you submitted was a very poor effort.

You may resubmit up to five short assignments on which you get a 1-4. If you resubmit a short assignment, you may resubmit it one time, and you have to resubmit by the start of the next Monday/Wednesday/Friday class after you got back your original assignment. When you resubmit a short assignment, we ask that you do a couple of things as a courtesy to your section leader.

- First, please make it clear that it is a resubmission.
- Second, please include your original submission along with the resubmission, so that your section leader can see your progress.

Lab Assignments

In-depth lab assignments will be due on certain **Wednesdays** through the course; you will have one week to work on each lab. You are not required to complete the lab assignment in the lab. You may use your own computer wherever you like, just as you do for short assignments.

Lab assignments require a significant time commitment. **Start early**, and if you need to, get advice from me, the graduate TA, or a section leader. Remember that you can get help at office/lab hours.

Like short assignments, you will submit lab assignments via Canvas. They are due at the start of class, 10:00 am, on the day they are due. If your lab assignment comes in at 10:01 am, it is late.

Late Penalties:

- < 8 hours: 10%
- < 24 hours: 20%
- < 48 hours: 40%
- ≥ 48 hours: no credit.

You are allowed at most one late submission (up to 48 hours) with no penalty – no excuse required. Indicate in your submission that you are electing to use your free pass; you cannot change this decision later.

Regarding late lab assignment submissions when working with partners: if you are working in a team of two and you want to use your free pass to submit a lab late (within 48 hours of the original due date/time), this will "cost" 2 free passes – yours and your partner's. If only one member (or neither member) has their free pass left, your group is subject to the regular late penalties.

Extra Credit

Some of the assignments may suggest extra credit work. Extra credit in this course will be tallied separately from regular scores. If you end up on a borderline between two grades at the end of the course (or are being considered for a citation), extra credit will count in your favor. Failure to do extra credit will never be counted against you, however. You should do extra credit work if you find it interesting and think that it might teach you something. It never pays to skimp on the regular assignments in order to do extra credit problems. Or, more simply put, **don't even attempt extra credit unless you are done with the regular assignment.**

One corollary of the way in which we count extra credit is that if you get a 30 on a lab assignment and also 5 points of extra credit, that is *not* the same as getting a 35 with no extra credit. The latter—35 points with no extra credit—is far better.

Handing in Assignments

Everything will be submitted through Canvas (with the exception of programming drills which will be submitted in person to your section leader at the end of each section). Even when an assignment has some written exercises, you are required to either type in a file or scan your written work and submit it electronically. To submit output from your program, submit a copy-pasted file in plain text format and/or a screenshot, as appropriate. For plain text, you can use a program like Sublime (that is what I use) TextEdit, NotePad, Emacs, or even Word, but be sure to save as plain text. For a screen shot, you can use the Grab utility on a Mac or the PrntScrn button on Windows.

If an assignment requires you to submit multiple files, zip all the files into a single zip file and submit that. If you have to revise your submission, submit your new zip file once more, with V2 appended (V2 for Version 2, V3 for Version 3, etc.); only the last version will be graded.

Sample Solutions

Sample Solutions for all of the lab and short assignments will be posted under the "File" tab in Canvas.

Partners

Much of the learning in this course comes from doing the programming exercises. On some lab assignments, you may work jointly with one other person, if so stated. No more than two people may work together on a given problem set. If you choose to work with someone else, you and your partner must submit a single joint assignment with both names on it, and you must work with the same person for the entire assignment. You cannot work with one person for some parts of an assignment and a different person for other parts, although you may work with different partners on different assignments.

If you work with a partner, you are still responsible for understanding the entire assignment. That means that splitting the coding into two halves, doing your half, and never looking at your partner's half is not a good idea. You can learn a lot by reading your partner's code and figuring out how it works, whether it is correct, and how it might be improved. You can also catch things like poor or missing comments that could cost you style points when the assignment is graded.

When working with a partner, I suggest that you borrow a practice from Extreme Programming, a method of writing code that many businesses find quite effective. It is called Pair Programming. One person (the driver) sits at the keyboard. The other person (the navigator) looks at the screen as the driver types, asking questions, making suggestions, and catching errors. Both of you will understand the code better if you discuss it as it is written than if you just write it (or read it) by yourself. Regularly trade off who is driver and who is navigator.

The usual reaction to this idea is, "that will take twice as long!" In practice it is usually faster than each person programming alone. The reason is that errors are caught earlier, and the amount of time are saved when debugging more than makes up for the lack of parallelism in code writing. Also, the code tends to be better written. These are the reasons that this idea has been adopted in industry.

How to Get Help

There are many ways for you to get help. You can visit me during office hours, you can go to 003 Sudikoff when course staff is there, or you can make an appointment with me, your section leader, or our graduate TA.

You can also ask for help through email, however, due to the size of the class, it may be difficult to provide timely help through email to all. Therefore, I encourage you to attend office hours, lab hours, and your recitation section, and to make use of **Piazza**.

Piazza can be used for class discussions. Piazza enables anyone enrolled in the course to post questions so that you can get help fast and efficiently from classmates, section leaders, the TA, and myself. Rather than emailing questions to the teaching staff, I encourage you to post your questions on Piazza. **Never include your own code in a post to Piazza.**

Please go to the Help page to find links where you can sign-up and use Piazza.

The Tutor Clearinghouse is another source of help. They will have private, one-on-one tutors available for this class. The tutors are recruited on the basis that they have done well in the subject, and they are trained by the Academic Skills Center. If you are on financial aid, the College will pay for three hours a week of tutoring. To get a tutor, go to 301 Collis and fill out an application.

Other Notes:

- If you visit me or any of the course staff for help with a program that you are writing, make sure that you can get to an electronic copy of your program. That way, we can try to compile and run it. It's nice to have a printed version of your program, but we cannot compile and run a printed version.
- When you email a program to one of us, you just need to add your program as an enclosure (*do not copy and paste your program into the message.*) to your email (I prefer it in a zipped folder), along with additional information as to what problems you have observed.
- When you email a question, please be as specific as possible, and tell us what you've done to try to figure out a solution for yourself. We reserve the right to not answer emails that say nothing more than "My program doesn't work; where is the problem?"
- If you have a question that does not require you to attach your code, then you might find **Piazza** to be a better source of help. Other students in the course can give you answers to your more conceptual questions, and you might find the answer already posted there. I'll repeat: **Never include your own code in a post to Piazza.**

Lab Help Hours

Several days per week, there will be course staff in **003 Sudikoff** (we will also use **002 Sudikoff** as an overflow space for lab help hours) to lend a helping hand with assignments or anything else you would like help with.

- Sundays, 6:00 – 9:00 pm
- Mondays, 7:00 – 10:00 pm
- Tuesdays, 7:00 – 10:00 pm
- Thursdays, 7:00 – 10:00 pm

Other Notes:

- **Please prepare before going to lab; course staff can be much more helpful if you've already made a solid effort at solving the problem you are working on.**
- The 002 and 003 Sudikoff labs are locked at all times, but if you're registered for CS 10, you should be able to get in with your Dartmouth ID.
- The entrances to Sudikoff are locked on weekends and after 6:00 pm, but if you're registered for CS 10, you should be able to get into the building with your Dartmouth ID.
- The second floor of Sudikoff also requires card access after 6:00 pm, and again, your Dartmouth ID should get you in, if you're registered for CS 10.

Textbook

The textbook is *Data Structures and Algorithms in Java*, **6th edition**, by Michael T. Goodrich, Roberto Tamassia, and Michael H. Goldwasser. The bad news: the textbook is required, and the paperback version lists for $145.95, though Amazon sells it for less, and they even rent it. The good news: you can purchase an e-book version directly from the publisher, Wiley, for "only" $59.50. You'll need to download the VitalSource Bookshelf software to read it. And in order to download VitalSource Bookshelf, you'll need to create an account on their site. The Wiley site links to the VitalSource site.

Software

We will be using a tool known as "Eclipse" to write programs this term. This software is a free Integrated Development Environment (IDE) that runs on both Windows and Mac OS X. You will install this software on your own computer as part of the first short assignment.

For more information about the software we will use, please see the **Course Software** page (link located at the top of this page).

Unfortunately for Windows users (and, yes, we know that many Dartmouth students have opted for Windows), the Mac will be the "primary" machine supported. The CS undergraduate lab for this course contains only Macs. I use a MacBook Pro. Some of the course staff have experience with Windows, however, so you may find it a good idea to ask them your Windows-specific questions. Because I don't run Windows, I will be unable to help you with Windows-specific problems. Indeed, one great asset of our course staff is that many of them can help you with Windows issues.

Note that you can always email your program to yourself. So even if you normally run Windows but you want to work on a Mac in the lab, you'll be able to work on your own code.

The Web

All documents, class examples, lab assignments, short assignments, and sample solutions related to the course will be on the web: http://www.cs.dartmouth.edu/~traviswp/cs10/.

Honor Principle

1. On exams, all work must be your own.

2. You may work on short assignments individually or in groups. Programs that you turn in, however, should be created, typed, documented, and output generated, yourself.

3. For the lab assignments, you may consult freely with instructors and classmates during the phase of designing solutions, but you should then work individually when creating your programs—typing, documenting, and generating output. During the debugging stage you may discuss your problems with others in the class, but you should not copy code to "fix" bugs. To do otherwise is a violation of the Academic Honor Principle. If you work with a classmate on any assignment, you should tell us who you worked with in a comment at the beginning of your program.

4. You should attribute the proper source in any code that you submit that you did not write yourself. This includes code that you take from outside references—for example a book other than the course text. And it even includes code that you take from class examples, a book, or the assignments. (I agree that may be tedious to attribute the source in code that we have given you, but we want you to be in the habit of attributing your sources.)

5. If you resubmit a short assignment and use code from the published solution, you should attribute that. Note also that proper respect for copyright laws as it applies to printed and software products is part of the Computing Code of Ethics.

6. Whenever we ask you to turn in sample runs of your program, the runs you turn in must be the result of actually running your program. It is a violation of the Academic Honor Principle to falsely represent output as coming from your program if it did not. If you change your program, make sure to generate output from the version of the program that you hand in. It's amazing how a seemingly minor change to the code can cause a big change to the output of a program. Also, make sure that when you are running a program, that it is your program; it is easier than you might think on a public Mac to run a program that someone else had left on the machine.

7. In the past, we have had a few incidents in which students turned in output that did not come from the program handed in. In each case, it turned out that the student had made a foolish mistake (in not rerunning the program or handing in an old version of the program or the output) and had not intended to misrepresent the work. Yet it caused many an uncomfortable moment for the student and also for the student's section leader and for the instructor as well. So please—pretty please with sugar on top—endeavor to verify that you're handing in output that comes from the very program you're handing in.

8. It is not easy to come up with good homework problems that help you learn a concept, are interesting, and require an appropriate amount of work. Over the years we have developed and refined a number of homework problems, and I plan to reuse some of them for this class. You should not look at any solutions to homeworks assigned in previous terms, including sample solutions, or solutions written by other students.

9. We have had some uncomfortable situations occur in the past, and I want to make it clear what the policy is. Two students, Alice and Ralph, discuss an assignment and design their code together. That is fine. But then they decide that since their programs would be so similar, they might as well have Alice type in the code, have Ralph make his own copy of the file containing the code, and then have Ralph make his own minor changes. This is a violation of the Academic Honor Principle. Although you may discuss and design with others, the code you hand in must be entirely your own.

10. Here's another situation that occurred. Trixie and Ed start working independently on a program. Trixie finishes and has a working version. Ed has trouble with his. Trixie helps Ed debug. That is fine. But then Trixie realizes that Ed has a section of code that is all wrong and the program she wrote has just the right code for that section. She shows Ed her program. Or worse, she gives Ed an electronic copy of her program so that he can just paste in the correct code. Either action is a violation of the Academic Honor Principle.

11. I realize that it can be hard to decide when you might be violating the Academic Honor Principle when we let you collaborate to a limited extent. Here is a good rule of thumb. If you are talking in normal English (or Chinese or German or some other natural language) you are probably OK. If you find yourself talking in Java code, you have crossed the line. So saying, "Your program runs forever because you have the wrong condition in the while loop" is OK. But saying, "Change while (x == 0) to while (x >= 0)" is not.

12. If you have any questions about whether what you're doing is within the Academic Honor Principle, do not hesitate to check with me. If it's late and you can't find me, you're better off erring on the side of caution.

13. Most violations of the Academic Honor Principle come down to failure to cite work

that is not yours. If you copy any portion of your program from your friend Elvira and represent it as your work, then you either intended to deceive or were careless about citing. Either case is a violation of the Academic Honor Principle. If you copy your entire program from Elvira but include the comment, "This code was copied in its entirety from Elvira," then you cited properly, though you didn't actually do the work. In this latter case, I would not report a violation of the Academic Honor Principle, though your grade on the assignment would be 0. But that would be far preferable to a COS hearing.

14. The same goes for code that you find in some other book or on the Internet. You are in violation of the Academic Honor Principle if you fail to attribute your sources.
15. You don't need to cite if you wrote the code yourself. You don't need to cite just because you're using a construct you saw elsewhere. For example, you need not cite for using `System.out.println("Hello")`, even though it was in the class examples. That would be like citing "printing press" in an essay! Nor do you have to cite just because you use a while-loop, even though you saw a while-loop in a class example. It's when you lift several lines of code from elsewhere that you need to cite.

To cite, include in a comment—near the top of your file is fine—stating where you got the code from:

```
// Based on code from page 66 of the textbook.
```

16. Please do not cheat. Cheaters—whether or not they are caught—bring dishonor upon themselves and upon everyone else at Dartmouth. To do that, for just a few lousy points in a course, is [insert your favorite strong adjective meaning "stupid" here]. Furthermore, if you cannot solve the short assignments and lab assignments on your own, then you will do poorly on the exams, which are worth almost half of your grade!

A Special Note for Working on Shared Computers

If you are working on a computer that is not yours—especially a Mac in 003 Sudikoff—or that someone else in the course might use, you should be very careful to remove your code from the computer when you are all done. You should probably email your code to yourself before you remove the code. Why do we tell you to do this? Because if you leave your code on a computer, and someone else can see it, then they can copy it and hand it in. If that happens, then we have a bad situation involving you (the copy-ee) and the other person (the copy-er), and it's difficult—if not impossible—to tell who was the copy-ee and who was the copy-er. By removing your code from the computer when you're done, you can avoid getting yourself into that situation.

To remove your code, you'll want to delete it from the Eclipse workspace. And you'll also want to move any other copies on the computer to the Trash (or the Recycling Bin) **and empty it**.

Religious Observances

Some students may wish to take part in religious observances that occur during this academic term. If you have a religious observance that conflicts with your participation in the course, please meet with me before the end of the second week of the term to discuss appropriate accommodations.

Disabilities

I encourage students with disabilities, including "invisible" disabilities such as chronic diseases and learning disabilities, to discuss with me after class or during my office hours appropriate accommodations that might be helpful to you.

Students with disabilities enrolled in this course and who may need disability-related classroom accommodations are encouraged to make an appointment to see me before the end of the second week of the term. All discussions will remain confidential, although the Student Accessibility Services office may be consulted to discuss appropriate implementation of any accommodation requested.

Advice

- Read the material I ask you to read, and follow the instructions. Many students have wasted a lot of their valuable time, and their grades have suffered, simply because they did not follow directions.
- **Start all assignments early**. With few exceptions, at the time you receive an assignment, you'll know everything you need to do it. There is no reason to procrastinate.
- Try to read through lab assignments and short assignments before going to your recitation section each week. I know this can be tricky because recitation sections are often scheduled pretty close to the time that assignments are posted, but reading over assignments before going to your recitation section is *very* helpful as your section leader will often start/end your section with time for specific questions about assignments that are out.
- Do not be afraid to get help. The purpose of this course is not to waste your time. If you are spinning and not making progress on a problem, please see me, the graduate TA, or a section leader. We can point you in the right direction without giving away the store.

Acknowledgments

A special thanks is owed to Scot Drysdale, Tom Cormen, and Chris Bailey-Kellogg. They are the masterminds behind much of the course content and have been actively involved in making CS 10 what it is today!

Three final pieces of advice

1. Don't fall behind in this course.
2. *Don't fall behind in this course.*
3. **DON'T FALL BEHIND IN THIS COURSE.**

The material in CS 10 builds on itself, and the pace is fast. As a result, it's easy to fall behind in this course, and if you do it's very difficult to recover.