CS 170Q: Introduction to Computer Science, Fall 2019

Instructor: Ting Li

Email: tli41@emory.edu

Office: Pierce Hall 127

Office Hours: Th 8:00am – 10:00am, F 2:30pm – 3:20pm

Location: Pierce Hall 104

Office Hour: Th 8:00am – 10:00am, Th 3:30pm – 5:30pm, or by appointment. Drops-in are encouraged if the door is open. You can always knock the door but I may ask you come back at another time. The optimal solution is always to come during the office hours or email me to schedule an appointment.

Textbook: The required textbook for this class is: zyBook's Programming In Java. This book is only available as an e-book. To access it please follow the following steps:

1. Sign in or create an account at learn.zybooks.com
2. Enter zyBook code EMORYCS170liFall2019
3. Click Subscribe

You will need a credit card or Paypal account to pay the subscription price of $58. Contact support@zyBooks.com for help or with any questions regarding obtaining a subscription.

If you drop the class in the first two weeks, you can contact support@zyBooks.com to get a full refund for the text.

If students wish to have a text for reference, Daniel Liang's Introduction to Java Programming is a great resource – but this text will not be explicitly addressed in the course.

Other required materials: You will, of course, need access to a computer during class time. Having a flash drive to regularly backup your work is also highly recommended.

Overview:

This course is an introduction to computer science for the student who expects to make serious use of the computer in course work or research. Topics include: fundamental computing concepts, general programming principles, and the Java programming language. Emphasis will be on algorithm development with examples highlighting topics in data structures.

Prerequisites: There are no official prerequisites although some familiarity with email and web browsers will be helpful. Knowledge of high school algebra and basic problem solving skills are assumed. This course is the first of a two-semester sequence for computer science majors and is followed by CS 171.

Course objectives: Students at the conclusion of this course should be able to...

- Effectively use primitive data types and common pre-made objects in the Java language
• Effectively use program-flow-control concepts (i.e., "for"-loops, "while"-loops, "if"-statements, etc.)
• Effectively use arrays for storing and manipulating a large amount of data
• Build classes and objects of their own design
• Effectively use subclasses and interfaces to facilitate Object-oriented design

Course expectations:

Classes: Class time will be used for short lectures, design examples, in class assignments, and exams. There is no requirement for attendance, it is essential that you come to the class on time every day and have the reading assignment finished. Students are responsible for all material covered in class. Missed work, quizzes, or exams will receive a grade of zero.

Exams: You will have three exams and a final exam this semester. The tests will emphasize reading, understanding, and debugging code -- more than writing code (students’ capability in doing the latter is primarily measured through their performance on the labs). That said, some questions on the tests might also require students to write code. Doing well on these exams will strongly correlate to having read and understood the readings online and other reference material provided, and having worked in earnest -- and successfully -- on the programs assigned up to that point in the class. All the exams including final are closed book and notes, and calculators are not allowed.

In general, makeups are not allowed for exams and assignments. However, you have a valid reason for a makeup exam, inform me as soon as possible. Valid reasons include medical emergency, a death in the family, or religious observations. Extensions will only be granted for emergency situations.

Lab Assignment: We have weekly lab assignment and the due dates for programming assignments will be on Tuesday unless there are exceptions. The lab periods are designed to give students valuable time to work on the lab assignments, whether that be accomplished by individually programming; collaborating with their partners, as appropriate; or seeking the advice of the instructor, as necessary. Unless a student has already submitted the final versions of the lab assignment assigned for that week, he or she is expected to be present in class for the entire lab period and working to these ends.

Students are expected to perform their work individually unless otherwise specified by the instructor. Plagiarism will be not tolerated. Students may discuss assignments in general terms with other students and may receive assistance from the instructor or classmates. Allowing students to work in pairs is intended to create conversations between students as they both develop each program in their own way -- conversations that will reveal when one approach is better than another, and should be adopted by both. Assistance does not mean obtaining working designs or solutions and modifying them; this is considered copying.

Make sure to regularly backup your workspace! Lab assignments will be submitted on Canvas for this course. Missed assignments or late submission will receive a grade of zero. Programming assignments must be done individually. Source file must be submitted in the class on the due day regardless its status (complete or incomplete).

Grading: Your grade for this course will be calculated as follows.

Reading 5%
Assignment 30%
Writing a program:

Writing a computer program is an act of inquiry. There is no "recipe" that can be given to students so charged. On the contrary (and in a quite literal sense) – with every program they write, students "create their own recipe" to accomplish the task in question.

Students will be given many opportunities to write programs – especially through the lab assignments. They will have a goal (a task their program must perform); they will have the tools they need (the language specification and API); and how they get to that goal is up to them. Students will attack these tasks in a variety of ways, some elegant, some brutish, but all will have to pull up their sleeves and get down in the trenches of figuring out how this new thing can be done with what they know.

The instructor’s job in this course is to demonstrate the requirements and capabilities of the Java language, give students some good guiding principles, and then largely get out of their way – letting them discover how to use this language to do what they want to do.

The instructor will also play a supporting role: 1) helping students see how certain "fundamental" questions can guide their efforts in accomplishing the goals given to them; 2) driving students away from inefficient solutions; and 3) revealing to students (primarily through questioning) cases they haven’t considered, that might cause their program to behave in a manner contrary to what they intended – and thus alerting them to the need to debug as necessary.

How to Approach Programming Assignments in this Course

*Students should plan on spending a considerable amount of time in front of a computer working on their assignments outside of class this semester.* Students should make every effort to work all of the programs assigned, as programming is a skill best learned by “doing”.

Assignments will involve designing, coding, testing and debugging programs based on a written assignment specification. These programs will involve a conceptual understanding of language features and require skill with various software tools. With programming it is important to “work smarter, not harder.” Brute-force approaches often lead to long, tedious, unsuccessful hours of work.

As is the case with most disciplines, however, the more one knows the right questions to ask when approaching a given task, the better off one will be.

Focusing on the fundamental questions that should always be at the forefront of any programmer’s mind can help you write correct, easy-to-understand, and efficient code with minimal effort.

For EVERY line or section of code written, students should be able to describe "what you are doing and why you are doing it". Students should force themselves to confront code they don’t really understand and figure out how it works, as opposed to semi-randomly making modifications to their code until it produces the correct output and then moving on -- without really knowing what they did, or why it worked.
When dealing with code that doesn’t make sense (or with unexpected output), students should first take some action to reveal the state of the program at the point of confusion. One could do this with a debugger, although often a well-placed print statement works just as well. Then, students should experiment with the existing code or write small programs "on the side" to test hypotheses they make about what is happening, until the functioning of the program is completely clear.

**Readability, Maintainability, and Good Style**

The importance of good style and readable, maintainable code in programming cannot be understated.

Through appropriate naming of identifiers and structuring of their code, students should seek to convey what the program is doing at each point in its execution, and -- even more importantly -- *why* it is being done. Single line and block comments can be added to further clarify the intent of the code written.

To aid in this end, students will be given some "Java Programming Style Guidelines" to follow when programming. These will include best practices with regard to naming conventions, indentation, constants, etc.. Such practices will greatly assist with debugging programs later, so students should do their very best to adopt these habits as early as possible.

**Special Accommodations:**

Access, Disability Services and Resources (ADSR) works with students who have disabilities to provide reasonable accommodations. In order to receive consideration for reasonable accommodations, students must contact ADSR and complete the registration process. Faculty may not provide disability accommodations until an accommodation letter has been processed; accommodations are not retroactive. Students registered with ADSR who receive a letter outlining specific academic accommodations are strongly encouraged to coordinate a meeting time with their professor to discuss a protocol to implement the accommodations as needed throughout the semester. This meeting should occur as early in the semester as possible. Contact Access, Disability Services and Resources for more information at (770) 784-4690 or adsroxford@emory.edu. Additional information is available at the ADSR website at [http://equityandinclusion.emory.edu/access/students/index.html](http://equityandinclusion.emory.edu/access/students/index.html).

**Honor code policy:**

All work done in this class is governed by the Oxford College Honor Code.

Students may not give, access, or receive any information not expressly permitted by the instructor on tests or exams. Collaboration between students on tests and exams is strictly prohibited.

**Religious Holidays:** Instructors are encouraged, not required, to accommodate students’ academic needs related to religious holidays. Please make every effort to negotiate your religious holiday needs within the first two weeks of the semester; waiting longer may compromise your instructor’s ability to extend satisfactory arrangements. If you need guidance negotiating your needs related to a religious holiday, the College Chaplain, Rev. Lyn Pace, ppace@emory.edu, Candler Hall 202, is willing and available to help.

**Please be aware that Rev. Pace is not tasked with excusing students from classes or writing excuses for students to take to their professors.**

Emory's official list of religious holidays may be found at: [http://www.religiouslife.emory.edu/faith traditions/holidays.html](http://www.religiouslife.emory.edu/faith traditions/holidays.html).
This syllabus is a guide for effective learning in this class; it is not a legal contract. The instructor reserves the right to modify the syllabus as needed.