CS 170: Introduction to Computer Science

Fall 2018

Instructor: Dr. Benjamin Purkis

MW 2:30-3:45pm, Lab: W 4:00-5:00pm, Pierce Hall 104

Instructor Information:

Email: benjamin.purkis@emory.edu
Office: Pierce Hall 128
Office Hours: MWF 1:00-2:30pm or by appointment; drop-ins are encouraged!
Drop-in policy: If my office door is open, you are always welcome to come in and ask whatever questions you may have. If my office door is closed, you are welcome to knock; I may answer, but I may also ask that you come back at another time. The best way to see me is to come during office hours or email me to set up an appointment.

Course Information and Policies:

Description: 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.

Course Objectives: At the end of this course, you will 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 and strings for storing and manipulating a large amount of data
- Build classes and objects of your own design
- Effectively use subclasses and interfaces to facilitate object-oriented design
- Begin to become familiar with event-driven programming

Website: This course will use Canvas as its main website. All course materials will be available through this website; you should get in the habit of checking it daily!

Textbook: No textbook is required for the course; if you would like a reference, Daniel Liang's Introduction to Java Programming is a good one.

Other Required Materials: You need access to a laptop computer; a flash drive is recommended to regularly backup your work. You will need to download the Java Development Kit (JDK) and the Eclipse IDE onto this laptop; the first lab section during the first week

will be dedicated to this task.

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 Expectations:

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

Lab Practice:	80 points
Lab Assignments:	200 points
Exams: Three at 130 points	390 points
Final Project:	130 points
Final Exam:	200 points
Total:	1000 points

Grades will be assigned by the following scale:

A	$\geq 92.5\%$	A-	89.5%-92.4%	B+	86.5%-89.4%
B	82.5%-86.4%	B-	79.5%-82.4%	C+	76.5%-79.4%
C	72.5%-76.4%	C-	69.5%-72.4%	D+	66.5%-69.4%
D	59.5%-66.4%			F	$\leq 59.4\%$

Classes: While attendance will not be taken directly, it is essential that you come to class on time every day, *having read the material assigned.* Your ability to get the most out of our class periods is greatly hampered if you are not prepared, especially since we only have 28 classes. You are responsible for all the material covered in class, even if you are absent.

Exercises: Many lectures will have exercises available for them. These exercises are for *your* benefit; they will not be collected. Answers are provided to help you check your answers. Although they are not collected, you should do these! They will help when confronted with a lab assignment or an exam.

Lab Practice: During each week's lab period, you will work with a group on a programming task or tasks which relate to that week's material. These tasks are due at the end of the period and must be submitted by each person through Canvas. They will be graded for completion only; did you make a good faith effort to finish the assignment? These group tasks are meant to give you some basic practice with the concepts before working on that week's lab assignment.

Lab Assignments: The lab assignments are the bulk of the graded programming assignments for the course. You will be given one or more lab assignments they will need to write

and submit on a weekly basis. Generally, you will have about a week to complete your submissions, although there may be exceptions to this. You are allowed to work with one and only one other person on these lab assignments, and your partner must be clearly noted in all of your code files (see the Honor Code section).

In the occasions where you are allowed to work in pairs (i.e. generally only labs and the final project), students should avoid the temptation to "divide and conquer" their assignments. Students giving in to this temptation will end up learning only half the material, which invariably shows up on test day. Allowing you 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.

Make sure to *regularly* backup your workspace! Lab assignments will be submitted on Canvas for this course, and are due at the date and time listed on Canvas. Catastrophic computer failure is not an excuse for a late project, and late lab assignments are not accepted.

Exams: You will have three midterm exams and a cumulative final exam this semester. All exams are done in class; the exam dates are:

- Exam 1: **Wednesday, October 3rd in class**
- Exam 2: **Monday, October 29th in class**
- Exam 3: **Wednesday, November 28th in class**
- Final Exam: **Friday, December 14th from 9:00-12:00 in our usual classroom**

The midterm exams will cover all material from that section of the course, although computer science is somewhat unavoidably cumulative; the final exam will cover material from the entire course. All exams are closed book and notes, and *calculators are banned.*

About Programming Assignments:

Writing a Program: Writing a computer program is inherently an act of inquiry. There is no "recipe" or formula that can be given to students so charged. On the contrary, with every program you write, you create their own recipe to accomplish the task in question.

During this class, you will be given many opportunities to write programs - especially through the lab assignments. You will have a goal (a task their program must perform); you will have the tools you need (the language specification and API); and how you get to that goal is up to you. You will attack these problems 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 task can be accomplished with what you know.

The Instructor's Role: 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:

- Helping students see how certain fundamental questions can guide their efforts
- Driving students away from inefficient solutions
- Revealing to students cases they haven't considered that might cause their program to behave in a manner contrary to what they intended - and thus altering them to the need to debug as necessary.

How to Approach a Program: Be prepared to spend a considerable amount of time in front of a computer working on your assignments, outside the standard class time! You 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.

For EVERY line or section of code written, you should be able to describe what you are doing and why you are doing it. You should force yourself to confront code that you don't really understand and figure out how it works, as opposed to semi-randomly making modifications to your code until it produces the correct output and then moving on. When dealing with code that doesn't make sense (or gives unexpected output), you should first take some action to reveal the state of the program at the point of confusion. One could do this with a debugger, or just use a well-place print statement. Then, experiment with the existing code or write small programs on the side to test hypotheses about what is happening, until the functioning of the program is clear.

Much of programming involves being able to ask the right questions. Students should consult the "Questions to Ask When Programming" handout on the Canvas Website.

Readability, Maintainability, and Good Style: The importance of good style and readable, maintainable code in programming cannot be overstated.

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, you 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 greatly assist with debugging programs later, so

students should do their very best to adopt these habits as early as possible.

Other Information and Policies:

Makeups: In general, makeups are not allowed for exams or assignments. However, if 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.

A Word on Technology: Please leave all iPods, MP3 players, etc. stowed and off for the duration of the class. Cell phones should be silenced. Return all seats and tray tables to the upright and locked position.

Honor Code: The Honor Code of Oxford College applies to all work submitted for credit in this course. In order to receive credit for your work, you must place your name on it. By placing your name on submitted work, you pledge that the work has been done in accordance with the given instructions and that you have witnessed no Honor Code violations in the conduct of the assignment.

For tests and exams, 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.

For the labs and final project, however, students may work in pairs if they wish, unless otherwise directed. If two students work as a pair on any given lab, a comment similar to the following (with the two names changed appropriately, but everything else absolutely identical) must be included as the first two lines of every .java file submitted:

```
//SUBMITTED BY: Mike Beck
//HELPED BY: Jon Fowler
```

If only a single student works on a given lab, a comment similar to the following (with the name changed appropriately, but everything else absolutely identical) must be included as the first line of every .java file submitted:

```
//SUBMITTED BY: Mike Beck
//HELPED BY: nobody
```

If a .java file is submitted without one of these two types of comments present and in the correct position, points may be deducted!

Lab partners may not turn in a single lab between them - each person should turn in his or her own assignment - even if it's substantially similar to their partner's (which in most cases would be expected).

Collaboration on lab assignments between students that are not lab partners is not allowed, painfully obvious to spot by the instructor, and does a serious disservice to all involved on test day.* Really.

** This means that if the instructor gets the impression that inappropriate collaboration on labs is occurring, which nullifies the capability of the labs to provide a genuine measure of students' ability to write code, the instructor will be forced to ask more questions on the tests that will require students to write code "on the spot". As such code-writing is done with pencil or pen, and without the benefits of code-completion or alerts to syntax errors like the Eclipse IDE provides, students will undoubtedly find such tests more difficult.*

Academic Accommodations: The Office of Accessibility Services (OAS) works with students who have disabilities to provide reasonable accommodations. In order to receive consideration for reasonable accommodations, students must contact OAS 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 OAS 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 accommodation as needed throughout the semester. This meeting should occur as early in the semester as possible. Contact the OAS for more information at (770) 784-4690 or `adsroxford@emory.edu`. Additional information is available at the OAS website at `http://equityandinclusion.emory.edu/access/students/index.html`.

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`.

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.