Note: Student work submitted as part of this course may be reviewed by Oxford College and Emory College faculty and staff for the purposes of improving instruction and enhancing Emory education.

Course Syllabus for
CS 170Q - *Introduction to Computer Science I*
Fall 2012: Section 12A, (Course# 4709 & 4710)

Lecture and Integrated Lab: MWF 12:00 – 1:05 PM and Tu 12:00 – 12:50 PM in Pierce 206

Instructor: Paul Oser
Email: poser3@emory.edu
Office: Pierce 122A
Hours: (CS 170 only) 1:05 to 2:00 PM on M,Tu,W,F and "Open-door" policy;
(all Math and CS courses) 3:00 – 6:00 PM on M,T,W,Th

Web: http://www.oxfordmathcenter.com/drupal7/node/10
Textbook: Java Concepts, 6th Edition by Cay Horstmann (reference only)
Other Materials: A 1GB (or bigger) USB Flash Drive is required.

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.

Goals for Student Learning: Students at the conclusion of this course should be able to…

- ☐ Effectively use primitive data types and 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 their own design
- ☐ Effectively use subclasses and interfaces to facilitate Object-oriented design
- ☐ Begin to become familiar with event-driven programming

Prerequisites:

There are no 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.

The "Ways of Inquiry" at Oxford:

"Ways of Inquiry" courses are designed to introduce students to the specific ways knowledge is pursued in each discipline through active engagement in the discipline's methods of analysis. INQ courses start with questions, are student-centered and often collaborative, and they place increasing responsibility on students for their own learning. Students not only experience each discipline's distinctiveness but also move beyond its boundaries to understand connections with other disciplines and fields.

The "Ways of Inquiry" Used in this Course:

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 some given task.

Students will be given many opportunities to write programs. 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 (through questioning) cases they haven't considered, that might cause their program to behave in a manner contrary to what they intended – and alerting them to the need to debug when necessary.

Students will have the opportunity to pair up with other students as they engage in many of their programming tasks – and the tasks themselves will often connect in some way to interesting problems or subfields of mathematics, cryptography, and/or other disciplines.

Lab Work / Assignments:

Students should plan on spending a considerable amount of time in front of a computer outside of class this semester.

Students should make every effort to work all of the labs assigned, as programming is a skill best learned by "doing". The Java software in Pierce 206 should also be available to you on the specialty computers in the Library and in the Kaleidoscope Lab, should you not have access to a computer of your own.

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 as well as requiring 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. The right approach can help you write correct, easy-to-understand and efficient code with minimal effort. Programs should be completed individually or as otherwise directed, although you are welcome to discuss general principles and concepts about the Java language with other students (and the instructor).

Students will be asked to periodically turn in via email their Eclipse project folder (which will ultimately contain all of the programs written for this course). Selected programs from this project folder will be graded. The programs graded may include example programs that were substantially fleshed out in class or programs left to the student to complete on their own.

Students are required to keep a copy of their Eclipse workspace on their USB drive, and will be expected to regularly backup their workspace, to protect against the loss of their drive and/or catastrophic failure of their computer.

Exams:
There will be three closed-book exams and a final that will test your understanding of the material. The three exams will each be given in two parts: one that emphasizes being able to read and debug code, and one that requires you to write code. Doing well on the first part of each exam will strongly correlate to having read and understood the notes online and other reference material provided. Doing well on the second part of each exam will strongly correlate to your having worked in earnest and successfully on the programs assigned up to that point in the class.

Final Project:
Each student will code (possibly with a partner) a final project of their own design (in consultation with the instructor) to demonstrate their mastery of the language and methods covered in this course.

Grading:

Assignments:	20%
Exams:	45%
Final Project:	15%
Final Exam:	20%

How the programs written for exams will be graded will differ from how other programs submitted in this class will be graded.

In the former, the grade will be based entirely on appropriate syntax, style (good variable names, no magic numbers, appropriate indentation, good program flow-control choices, avoidance of brute-force approaches, etc…), and correct output – with very minimal "partial credit".

In the latter, roughly 60% of the grade will be determined by these aforementioned items, while the remaining percentage of the grade will be based on the comments laced throughout the code and how they contextualize and explain the code to which they are attached in terms of the "good questions to ask" and other good habits of inquiry in computer science.

Late Policy:

Students are expected to be present for all scheduled tests. Any conflicts should be brought to the instructor's attention as soon as possible. If a legitimate reason exists for missing a test – as determined by the instructor – then the test must be taken prior to the regularly scheduled date. In the unusual circumstance where taking the test early is not possible, ***students should be aware that any make-up tests given will be designed to be more difficult to offset the additional time given for study***. Students must provide written documentation in advance of any special accommodations required for testing. This includes additional time or other needs. The final exam cannot be rescheduled.

In general, late programming assignments will not be accepted; this policy will be waived only in an "emergency" situation with appropriate documentation and at the instructor's discretion.

Honor Code Policy:

All class work is governed by the Oxford College Honor Code. No collaboration is allowed on programming assignments except that which is explicitly authorized by the instructor. (At the discretion of the instructor and for non-exam based programs, this may involve being allowed to discuss the programs in question with one other student that is identified by name at the time of submission.)

Students should be aware that it is actually fairly easy to detect inappropriate collaboration or copying by running programs that analyze submitted programs.

Every program submitted must have the following comment included at the top of the file containing the respective "main()"or "run()" method. Full credit for the program will not be given if this is not present.

```
/*
THIS CODE WAS WRITTEN IN ACCORDANCE WITH THE HONOR CODE POLICY FOR THIS COURSE. I
HAVE NOT BEEN INVOLVED IN ANY INAPPROPRIATE COLLABORATION WITH RESPECT TO THE
DEVELOPMENT OF THIS PROJECT. _Your_Name_Here_
*/
```