CS170: Introduction to Computer Science Fall 2005

Instructor:	Jianmin Ma **jma3@learnlink.emory.edu**
Office:	Seney 115 (770-784-8398), TTh3-4PM and by appointment
Room:	TTh 1:00-2:15PM, Piece Hall 205
Text:	**Computing Concepts with Java Essentials**, 3E by **C. Horstmann**.
Prerequisite:	The desire and enthusiasm to fuel learning to program =)

Course objectives: The primary objective of this course is to provide a gentle yet firm introduction to computers and programming them, which entails many things:

- ✓ Understanding what computers are and learning how to bully them around
- ✓ Learning to think like a programmer
- ✓ Developing analytical and problem-attacking skills
- ✓ Formulating and conveying precise and concise instructions
- ✓ Understanding programming paradigms like object-oriented programming
- ✓ Using Java as a means to our end

Grades: Each assignment will be graded on a scale of 100 points. Your final grade will be determined by a weighted average of the individual grades earned on the assignments. Programming projects will comprise 40% of your final average, exams 30%, Labs 10%, and the final exam 20%, . Programs that do not compile will be penalized heavily (will typically be assigned a grade of 50% without further consideration). You will have two exams each accounting for a total of 30% of your final average. Grades of A-, B+, B-, C+, and C- may be assigned for sums of points near the above cut-off totals. In addition, the assignment of plus and minus is dependent on the overall class distribution of sums of points. In general, letter grades will be determined as follows: A: >= 91%, B 81-89%, C 68-79%, D 55-67%, F <=55%

Class policies: Programming projects and exercises will be handed out regularly. No collaboration is permitted on any assignment unless otherwise instructed. I take academic misconduct very seriously, and such matters, which hopefully will not arise, will be handled according to the Emory Honor Code. In addition assignments will be subject to the Math/CS department's Statement of Policy on Computer Assignments.

Late projects will not be accepted; however, each student is granted **2 penalty-free extensions**. Students must request these extensions **before the project is due**. Additional extensions or makeups will be given only with the appropriate documentation (a doctor's note, a subpoena, etc..) and even with appropriate documentation are subject to the instructor's discretion.

I need help; what do I do? First, make sure you understand the fundamentals. Different perspectives often help, so I recommend you try out some of the notes and tutorials in the Resources section below. Sun's The Java Tutorial seems to be a decent place to start. You can find tutorials on the Java language itself, including essential classes in the API, as well as getting the Java SDK tools installed.

If you need help while working on an assignment, the first thing to keep in mind that each assignment is to be treated as a take-home exam. You should read the Oxford College Students Honor Code. Here are some, hopefully helpful, tips:

- **Check for consistency**: for instance, if you're having problems trying to create a new StringBuffer object using the command `new StringBuffer(String "myString")`, think about how we created new instances of other classes. Did we create new Rectangles in the same way: `new Rectangle(int 5, int 10, int 20, int 30)`? No, we used the constructor call, `new Rectangle(5, 10, 20, 30)`, so then why should we expect the constructor call, `new StringBuffer(String "myString")` to work? In this case another check for consistency shows that commands like `String str`, more generally `<type> <identifier>`, actually declare new variables, which is not what we wanted to do inside the parenthesis above.

- **Play in a sandbox**: make test programs to explore classes and constructs you haven't encountered before. Then once you feel comfortable and are done playing, you can integrate the new stuff with your existing code. This can help isolate errors.
- **Divide and conquer**: the above is really an instance of this tip. Break your problem up into smaller pieces and forget about everything else while concentrating on each piece. For example, when you're writing a method, forget about how it's going to be used later -- just focus on the task at hand, which would be making sure that the method does what its supposed to.
- **Listen to the compiler**: don't despair just because you don't understand an error message. At least you know which line contain errors, so go back and check these lines in excruciating detail. Also tackle each error separately. Often fixing one error will help resolve others.

Resources: Course materials will live online and few will be handed out in class. Class resources

- Reserved and keywords in Java
- A page with some common compiler error messages

Notes and tutorials

- Sun's tutorial on getting started with Java in Windows: A nice tutorial on getting the Java SDK tools running in Windows
- Sun's The Java Tutorial: "A tutorial on just about every aspect of Java from its creators"
- Unix resources
- Emacs Beginner's HOWTO

Software

- Textpad seems to be a decent Windows editor that seems understands vi and emacs commands
- Download Sun's Java Software Development Kit
- BlueJ
- Download Adobe Acrobat Reader
- Download Coffee Cup FTP for windows (you need this to transfer files with the Sun machines)

Reference materials

- Sun's Java 1.4.2 Documentation

Fun stuff

- Sun's Java applet page