CS150: Introduction to Computer Science

Fall 2006
TuTh 11:30– 1025 p.m, Piece Hall 206

Instructor: Jianmin Ma
E-mail: learnlink Jianmin Ma
Office: Piece 122B
Phone Number: 4-8398
Office Hours: W 2-3 p.m. and F 1-2 p.m., and by appointment
Lab Hours: M 2-3 p.m.

Text: Java Applets: Interactive Programming, by Elizabeth Suger Boese

Java Applets: Interactive Programming (COLOR) ($58.83 + shipping) ISBN: 978-1-4116-7994-8

Java Applets: Interactive Programming (BLACK & WHITE) ($25.28 + shipping) ISBN: 978-1-4116-7993-1

Prequisites: The desire and enthusiasm to fuel learning to program =)

Course Contents

The purpose of the CS150 course is to familiarize students not intending to become computer scientists (majors or minors) with the fundamentals of Java programming, program design and problem-solving. The course is oriented towards practical skills including current Java programming technologies for Java applets, graphical user interfaces (GUIs) and Web pages.

The course covers the basic Java syntax and language features, compilation, interpretation, execution, class and object usage, graphical interfaces, program-user interaction, and the Java API. Problem-solving techniques and object-oriented programming are also covered.

The workbook emphasizes the practical hands-on approach taken by the course. This includes examples, practice exercises and step-by-step programming exercises.

Course objectives:

The primary objective of this course is to provide a gentle introduction to practical programming in java (applet). Upon completion of this course, students should be able to

- modify, compile, debug, and execute Java programs,
- create applets and integrate them with webs;

- comprehend the art of programming and, in particular, the structure and meaning of basic Java programs,
- design and build programs using problem-solving techniques such as top-down design,
- understand how to create graphical interfaces and Java applets for a Web page.

Grading and Evaluation

Students are evaluated on the basis of assignments, programming projects, proctored exams, and class attendance/participation.

A student's grade is based on

- 6-8 programming/homework assignments *(20% of Final Grade)*
- lab assignments (80%) and participation(20%) *(15% of Final Grade)*
- two midterm exams *(20% of Final Grade)*
- project (3 phases, worth 10% 25% and 65%) *(30% of Final Grade)*
- final exam *(15% of Final Grade)*

Final Grade: Based on the final weighted percentage computed from the above.
A: 91% and above B: 80-89% C: 70-79% points
D: 60-69% points F: 59% or less

General Topics

1. What is programming? What is Java?
2. The process of compilation and interpretation
3. An introduction to Java applets and graphical programming
4. Graphical User Interfaces; Drawing; Components
5. Image processing
6. Visual design and layout
7. Designing classes and methods
8. Problem solving; simple algorithms
9. Java basics: types, variables, statements, syntax
10. Handling mouse events and actions; control flow
11. Testing and debugging programs
12. Arrays and loops
13. Sound; Animation; Games; Tricks
14. Setting up applets on the Internet

I need help; what do I do?

First, make sure you understand the fundamentals. Read your textbook carefully, run the programs in your text and understand them.

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.

Student Honor Code
The responsibility for maintaining standards of unimpeachable honesty in all academic work and in campus judicial proceedings falls upon every individual who is a part of Oxford College of Emory University. The Honor Code is based on the fundamental expectations that every person in Oxford College will conduct his or her life according to the dictates of the Honor Code and will refuse to tolerate actions in others which would violate the Honor Code.

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 Student Honor Code (http://www.emory.edu/OXFORD/CampusLife/Policies/honor.html). .

Here are some examples of cases which are clearly unacceptable and others that are clearly considered acceptable in the CS Department.

Examples of Behaviors likely to get students in trouble:

- Turning in someone else's work as one's own (with or without the other person's knowledge).
- Turning in a completely duplicated assignment is a flagrant offense.
- Intentionally or unintentionally making it possible for someone else to turn in your work as his or her own. Students who share their work with others are as responsible for academic dishonesty as the student receiving the material. Students are not to show work to other students prior to the assignment due date, in the class or not. Students are responsible for the security of their work and should ensure that printed copies are not left in accessible places, and that file permissions on accounts on shared machines are set to be unreadable by others.
- Several people writing one program and turning in multiple copies, all represented (implicitly or explicitly) as individual work.
- Using any part of someone else's work without the proper acknowledgment.
- Stealing an examination or solution from the instructor. This is an extremely flagrant offense.