This course has a web site at URL http://www.cs.uic.edu/~sloan/CLASSES/202-course-info
All the information in this handout, and maybe more, is there.
This course also has a [Blackboard page]. It contains a link to this information and a links section pointing to useful tools, and is one way to find the first homework. We will definitely use it for its grade book, and that is where I’ll post any solutions etc., that I don’t want world readable. I’ll also put useful links there to Java documentation, etc. You can reach Blackboard directly at http://blackboard.uic.edu or get to Blackboard from a quick link from the university home page.

1 Course Topics

This course will be a combination of the theoretical and the applied. Basic combinatorics will be the highlight of the discrete math material.

The course will be taught in Java!

Since about half the class is coming from the old C++ sequence, I will give a crash course on Java for C++ folks at the start, but those folks will need to be prepared to work like demons.
Here is an overview—first two lectures in great detail, rest not so detailed, and not necessarily in the order we’ll cover it.

— Subject to change without notice —

• Fast introduction to Java for C++ programmers. 1 week regular semester time; Tuesday June 1, summer time. (Main text, Chapters 1 and 11.1.1.)

• Exceptions, Interfaces, Abstract Classes, and Casting in Java. 2/3 week regular semester time; part of Thursday June 3, summer time. (Main text, Chapter 2.3–2.4.)

• Iterators. 1/2 week regular semester time; rest of Thursday June 3, summer time. (Main text, Chapter 5.5.)

• Priority Queues and the Heap Data Structure. (Chapter 7.)

• Throughout: Various intermediate level Java programming tidbits, including inheritance.

• Basic combinatorics, or how to count. (Epp text.)

• Maps, Dictionaries, and Hash Tables. (Chapter 8.)

• Binary Search trees, and one balanced binary tree scheme (Either Splay Trees or (2,4) together with Red-Black trees). (Parts of Chapter 9.)

• Sorting.

• Graph Theory. (Epp)

• Graph Data structures.

• If Time allows:
  
  – An additional balanced tree scheme
  – B trees
  – Text compression
  – Huffman Codes
2 Textbook

The two required textbooks for the course are:

   Note the book’s web site at http://java.datastructures.net.

2. Susan Epp’s Discrete Mathematics Book

There are several optional Java books, depending on one’s background:

1. *Beginning Java 2*, by Ivar Horton, is a good comprehensive tutorial introduction to the language for somebody who can already program some in some language. It should be priced more like a “trade/professional” book, i.e., at most fifty bucks, than like a text book.

   If you have come through the Java sequence, you probably do not want to buy this book; if you have come through the C++ sequence, you may.

2. Pick your choice of either, or none, of two Java reference books that some of us older folks like to keep next to us on the desk when we’re hacking, instead of being 100% reliant on the on-line help:

   (a) *Java in a Nutshell*, by David Flanagan, O’Reilly.

   (b) *The Essential Java Class Reference for Programmers*, by Brian Durney, Pearson Prentice Hall. (A much shorter reference to major classes.)

3 Grading

This policy is subject to change at any time for any reason.

There will be three to seven major programming assignments, which will count for 25–30% of the grade.

There will be one midterm counting about the same or a little less than the major programs, and a final counting for about 40% of the grade.

There will be some discrete math problem sets. They will be worth perhaps 5–15% of the grade.

Late homework will not be accepted, because solutions will often be given on the day the problem sets are due. They will receive a grade of 0.

This policy is subject to change at any time for any reason.

3.1 Two easy ways to fail

You must pass the final to pass this course.

If you have grades under 35/100 on more than two of the major programs, you will automatically fail this course.
4 Rules and regulations

Incompletes

The *UIC Undergraduate catalog* states that in addition to needing excellent justification for an incomplete, a student must *also* have been “making satisfactory progress in the course.”

Therefore, no matter how good your excuse, I will not grant you an incomplete if you have less than a C average at the time you ask for an incomplete.

Academic Integrity

All work you turn in must be entirely your own. You may discuss general strategies for programs with other students in the class, but you must do 100% of the coding by yourself.

Similarly, you may discuss the math problems with other students—in fact, I encourage you to do so—but you are expected to write up your solutions by yourself. *If you do work on the math problems with other students, please put the names of your group at the top of your problem set.*

However, the minimum penalty for any cheating will be an E for the course (not just the assignment in question!), and the maximum penalty is expulsion from the University.

“You can do absolutely anything here, except for murder and plagiarism. And even in that case, they’re often willing to punish first-time offenders lightly—if it’s only murder.”

—Direct quote of speaker at Prof. Sloan’s freshman orientation in fall of 19-mumble-mumble.

I will be checking for copied programming assignments using a quality automated tool.

*You are responsible for protecting your own code. This includes making sure not to throw out print outs in the computer lab as well as keeping your files protected. Failure to protect your own code will also be treated as a cheating/copying violation.*

In industry, if you are the person whose carelessness causes your company to lose vital intellectual property that has been protected through trade secret, I guarantee you that you will be harshly treated.

5 Initial Reading Assignment

*For Week 1 of semester time, or Tuesday June 1 Summer time*: Java Basics. If you already know Java, then you probably know just about all of this. Note that
if you know C++ you know most of Java’s syntax; I’m trying here, especially from the Horton book, to pick out the differences.

From the text, Chapter 1 and 11.1.1. For more detail, either see the portions of the Sun Java Tutorial pointed to by the first problem set, and/or the following selections from the Ivor Horton *Beginning Java 2 Book*:

- Chapter 1, Java Introduction, including a bit about the Java library and getting your first program to compile.

- From Chapter 2, Data and Variables: Pages 38–40, 47–50 (simple output), 58–61 (Math library functions), 72–74 (comments!).

- Chapter 3, Loops and Logic (control flow): 103–104 (using a `double` (or `float`) as an index to a `for` loop, 108–113 (`labeled` break and continue statements).

- Chapter 4, Arrays and Strings: 121–129 (basics of 1-D arrays), 134–141 (basics of Strings, 149–145 (accessing char or substring of a string).

- Chapter 5, Defining Classes: Pages 172–183 (basics of classes), 186–187 (constructors); 196–202 (using objects, examples).

For week two of term time, Thursday June 3 summer time. Text, 2.3, 2.4, and 5.5.

For much more detail on exceptions (more than you probably need), you could read Chapter 7 of Horton instead of or in addition to 2.2 of the text.

### 6 Programming and such

Your programs will need to be turned in using turnin on the CS department system. Your programs must compile without error using javac.

You may do your programming however you wish, but I have suggest that you use one of the following methods:

1. A smart editor that knows about Java and then compile at the command line with javac.

   Good editors include emacs (either GNU emacs or XEmacs) and JEdit. Stupid editors that are not helping you as much as you should be helped include Notepad and vi. You will save lots of time over just this one course, much less over the CS major, if you switch to a good editor.

2. If you want a helpful, CS-student-friendly Integrated Development Environment (IDE), I suggest JGrasp. It includes a nice debugger and a nice visualization of control flow, and it is really easy to get it working immediately.
3. If you want a very powerful IDE and are willing to study how best to use it for a while, I suggest Eclipse.

Warning: if you use other tools, you must be sure that they don’t do strange things with your code (e.g., add package statements, or rely on custom libraries that are not part of the standard Java API) that will cause it not to compile at the command line on a Unix system.

7 First Problem Set: Due in 44 hours!

Problem Set 1 is a Java warm-up. It is intended to review the amount of Java that I would expect CS 102 students to know by two-thirds of the way through the semester. During the semester, I would give it out during the first week, and it would be due in a week; i.e., during Week 2. In the weird accelerated summer time, that means it’s due right before our second lecture—it’s due Thursday by noon.

You can and should access it from the course web page (reached any of directly, through my homepage, or through Blackboard).