CMSC 221 – Data Structures
Syllabus, Fall 2005

**Professor**

Lewis Barnett

Office
Jepson Hall, 212a
(804) 289-8091
email: lbarnett@richmond.edu

Office hours: Monday through Thursday, 1:00 – 2:00pm, or by appointment. (Really, you can drop by my office any time, but I make a special effort to actually *be* in or near the office at these times.)

**Prerequisites**

There are no formal prerequisites for CS221. If you have had an introductory programming course in high school or college that covered built-in data types (integers, real numbers, characters, etc.), control flow constructs like IF-THEN-ELSE, WHILE and FOR loops, arrays (one and two-dimensional) of simple data types, and function and procedure calls using parameters, then CS221 is an appropriate course for you. The course will be taught in the Java programming. If you are not familiar with, see me for some materials that will bring you up to speed on these tools.

**Course Description**

CS221 covers many fundamental ideas, techniques, and issues of the discipline of computer science. The organizing focus for this course is the structuring of the data on which computer programs operate. We will focus on various ways to access data, discuss various ways to organize the data to support the access patterns we are interested in, and trace the impact that data organization choices have on implementation and performance. The concept of *data abstraction* or the *Abstract Data Type* as supported in object oriented programming will be emphasized throughout the semester. We will also consider recursion, dynamic data structures (data structures that grow or shrink on demand) and their use, and the analysis of algorithms.

**Field of Study: Symbolic Reasoning**

The faculty of the School of Arts and Sciences have approved CS221 as satisfying the Symbolic Reasoning Field of Study in the General Education curriculum. The statement of the Field of Study: Symbolic Reasoning requirement from the current Undergraduate Catalog is as follows:
As a field of study, symbolic reasoning is distinguished by its internal attention to logical consistency and by its wide external applicability. This field of study emphasizes symbolic problem solving, a process that includes translating problems into terms that are amenable to treatment within a symbolic system, understanding consistent rules by which the information relevant to the problem may be processed in order to obtain a solution, recognizing important underlying principles that govern the application of these rules, and judging both the appropriateness of known solution methods to a particular problem and the quality or reasonableness of the solution obtained.

Here is the statement approved by the faculty concerning how CS221 satisfies the requirement:

Computer Science 221 continues the development of the problem solving methods begun in Computer Science 150 (Introduction to Computing). Problem solutions are expressed in a high level programming language. The student is expected to master the uses of data abstraction, recursion, and dynamic data organization in appropriate problems. The student is expected to be able to understand how the complexity of an algorithm can be measured. Mastery of these topics enables the student to deal with more sophisticated problems than could be presented in Computer Science 1.

Readings


Appendix A of the textbook provides an introduction to the Java language. You may find yourself wanting more information about the language, or simply an alternative reference. The appropriate book for you will depend on your background. I'd suggest going to Barnes and Noble or Borders and browsing their Java section to find a book which appeals to you. The book that I used to learn Java a few years ago was *Core Java* by Horstmann and Cornell. This book has expanded over the years, and now comes as a two volume set; Volume 1 is all you should need for this course.

In addition, various resources of interest for this course can be found on the class web page:

[http://www.mathcs.richmond.edu/~lbarnett/cs221/](http://www.mathcs.richmond.edu/~lbarnett/cs221/)
Grading Policy

<table>
<thead>
<tr>
<th>Assignment Type</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homework/Lab assignments</td>
<td>15%</td>
</tr>
<tr>
<td>Programming assignments</td>
<td>25%</td>
</tr>
<tr>
<td>Tests</td>
<td>40%</td>
</tr>
<tr>
<td>Final Exam (cumulative)</td>
<td>20%</td>
</tr>
</tbody>
</table>

Makeup tests will not be given; in the event that a test is missed, the final exam grade will be substituted for the missed test in calculating the final grade for the course. Students may complete Laboratory exercises on their own time if they must miss a Lab period for a legitimate reason (illness, family emergency, or participation in a University sanctioned extra-curricular event). Other missed assignments will receive a grade of 0. I expect assignments to be turned in on time.

Assignments

There will be two types of outside-of-class work in this course.

- Homework problem sets - exercises from the textbooks or other sources which will be graded and returned for discussion.
- Programming projects - programs which you will design, implement, test, and document based on a general description of a problem to be solved. As the semester progresses, the size and difficulty of the programs will increase. Some programming assignments may be outgrowths or continuations of laboratory exercises.

The Windows XP computers in Jepson G30 and G25 will be available at all times outside of scheduled lab periods for your use. A schedule for the lab will be posted on the door. Always check the schedule to be sure the lab is not in use by a class before entering. Procedures for turning in programs will be discussed when the first programming assignment is given. Late assignments will not be accepted.

If you have your own computer, there are free or inexpensive Java development environments that you can install on your computer. For more information, see links on the class home page.

Collaboration

As noted above, there will be two types of outside work assigned: problem sets, and programming assignments. It is appropriate and helpful to work on the problem sets together or in small groups. Some of the programming assignments may also be team projects; this will be discussed in the programming assignments. However, unless explicitly stated otherwise in the assignments, the programming assignments must be your own work and will be pledged under the Honor Code. Programming projects may be discussed with others subject to the “empty hands” policy – you may freely discuss ideas and approaches with other students subject to the restriction that each student must leave
the discussion without any written or otherwise recorded material. In your project write-up, you must also document any person or source that you consulted for that project. Failure to comply with this policy will be treated as an Honor Code violation. If you experience problems in completing a programming assignment, please see me for help during office hours, or make an appointment to see me at another time if you are unable to come during office hours. You may also ask me questions by electronic mail. Note that I may not always be available in the final hours before a due date! You should start programming assignments far enough in advance that you will have the opportunity to discuss any kinks you run across with me in a timely fashion.

Collaboration is, of course, not allowed on tests. Note that tests will be given during the lab periods, and that this additional time will allow me to include significant programming components on each test. It is in your best interest to use the programming assignments to hone your own programming skills rather than relying on advice from others to complete them.

**Attendance Policy**

All students are expected to attend each class meeting. Each unexcused absence will result in a reduction by 1 point of your final average. I reserve the right to assign the V grade (Failure due to excessive absences) to any student with more than six unexcused absences.

**Course Outline**

An outline of topics to be covered follows. This list does not necessarily reflect the order in which we will cover the topics -- they will quite likely be interwoven with each other.

Note the date for the final exam: THIS DATE WILL NOT CHANGE. Please make your travel plans accordingly.

- Administrative stuff, course intro
- Refresher on the Java programming language
  - Functions and parameters
  - Objects and references
  - Arrays
  - Simple classes
- Exceptions
- Inheritance and polymorphism
- Algorithm Analysis
- Program construction concepts: data structures and abstraction
- Recursion
  - Recursive algorithms
- Abstract Data Types
  - Data and operations on data
  - Using classes to implement Abstract Data Types
• Stacks
• Queues
• Linked Lists
• Searching and sorting
  o Quicksort algorithm
  o Complexity analysis of quicksort
• Hashing; the table ADT
• Binary trees
  o Implementation
  o Traversals
  o Binary search trees

**Test Schedule**

We will use lab periods for the three tests. While we may not actually be using the computers during the tests, the extra time will allow you to be less hurried as you work.

• Test 1: Thursday, September 22
• Test 2: Thursday, October 27
• Test 3: Thursday, November 17
• **Final Exam:** Monday, December 12, 2:00 pm-5:00 pm