Computer Science 062 – Spring 2007

Instructor & Text

Instructor: **Kim Bruce**
222 Edmunds, x7-1866
kim@cs.pomona.edu
Office Hours: MWF 1:30-3:00 p.m., & by appt.
TA: **Adi Kovashka**

Lectures: MWF 10:00 - 10:50 a.m., Lincoln 1135
Lab: T 1:15 p.m. - 2:30 p.m. or 2:45 p.m. - 4:00 p.m., Edmunds 229
Texts: **Required:** *Data Structures & Algorithms in Java* by Michael T. Goodrich & Roberto Tamassia, Wiley, 2006.
Strongly Recommended: C++ for Java Programmers by Mark Allen Weiss, Prentice Hall, 2004.
Course web page: www.cs.pomona.edu/classes/cs062/
Instructor's web page: www.cs.pomona.edu/~kim/

Overview

This course couples work on program design, analysis, and verification with an introduction to the study of data structures that are important in the construction of sophisticated computer programs. Because we will be interested in studying more modern techniques for designing and implementing efficient computer programs, we will primarily be using the object-oriented programming language, Java, though later in the course you will learn to develop programs in C++ as well. We will see that the object-oriented style of programming is extremely useful in designing large, complex programs and supporting reusable software.

One goal of this course is to make you comfortable with using UNIX. To that end we will be using UNIX tools like emacs under Mac OS X on Macintoshes available in the Computer Science Macintosh Laboratory in Edmunds 2189, and compiling and executing Java programs using the javac and java commands. Students should use these machines when possible. It will also be possible to install these tools on your own computer. Later in the semester we will use the free Eclipse IDE in order to take advantage of some tools they provide and their debugger.

Students will be expected to write several programs, ranging from very short programs to more elaborate systems. Since one of our goals in this course is to teach you how to write large, reliable programs composed from reusable pieces, we will be emphasizing the development of clear, modular programs that are easy to read, debug, verify, analyze, and modify.

I assume that all students enrolled are comfortable writing small to medium-sized programs (< 10 pages of code with several interacting classes) in Java. The knowledge assumed is generally equivalent to that of CSCI 051 as offered at either Pomona or CMC. If you have any doubts as to whether your

programming experience is sufficient for this course or if you have never programmed in Java, please see me as soon as possible.

By the end of this course you should have a good understanding of the object-oriented design, coding, and debugging of programs in Java and C++, and have a good understanding of how one might analyze programs for correctness and efficiency. In particular, you will understand the trade-offs involved in selections of different data structures and algorithms to solve computational problems.

This course is a prerequisite for most upper level Computer Science courses. It may be taken either before or after CSCI 052.

Lectures and Readings

The schedule on the following two pages shows the topics to be covered at each class meeting during the semester. Students should consult the on-line version of the course syllabus (see URL above) regularly to see the most current version of the schedule of topics and readings.

All reading assignments are from the text. Students should come to class having completed the indicated readings for the day. You should attempt to work all the problems at the end of each section as you are reading. Chapter review problems will be assigned during each lecture.

Lecture	Date	Topic	Reading
1.	Jan. 17	Introduction & Overview	Ch 1
2.	Jan. 19	Object-Oriented programming & Java	Ch 2
3.	Jan. 22	Object-Oriented Design	Ch 2
4.	Jan. 24	Linked Lists	Ch 3
5.	Jan. 26	More Linked Lists	Ch 3
6.	Jan. 29	Analysis of Algorithms	Ch 4
7.	Jan. 31	Sorting	Ch 11.1-11.2
8.	Feb. 2	Sorting	Ch 11.2-11.3, 11.5
9.	Feb. 5	Stacks	Ch 5.1, 14.1
10.	Feb. 7	Queues	Ch 5.2
11.	Feb. 9	Deques	Ch 5.3
12.	Feb. 12	Array Lists	Ch 6.1
13.	Feb. 14	Node Lists & Iterators	Ch 6.2-6.3
14.	Feb. 16	More Lists	Ch 6.4-6.5
15.	Feb. 19	Trees	Ch 7.1-7.2
16.	Feb. 21	More Trees	Ch 7.3
17.	Feb. 23	Trees & Visitors	
18.	Feb. 26	GUI Programming: Events & Swing	
19.	Feb. 28	Concurrency	
14.	Mar. 2	Concurrency & JUnit	
21.	Mar. 5	*Midterm*	

Lecture	Date	Topic	Reading
22.	Mar. 7	Priority Queues	Ch 8.1-8.2
23.	Mar. 9	*No class*	
	Mar. 12-16	**Spring Break**	
24.	Mar. 19	Heaps & Heapsort	Ch 8.3
25.	Mar. 21	Intro to C++	
26.	Mar. 23	More C++	
27.	Mar. 26	More C++	
28.	Mar. 28	More C++	
	Mar. 30	**College Holiday - Chavez Day**	
29.	April 2	More C++	
30.	April 4	More C++	
31.	April 6	More C++	
32.	April 9	More C++	
33.	April 11	More C++	
34.	April 13	Maps & Dictionaries	Ch 9.1-9.2
35.	April 16	More Dictionaries & Hashing	Ch 9.3-9.5
36.	April 18	Binary Search Trees	Ch 10.1
37.	April 20	Balanced Trees	Ch 10.2-10.3
38.	April 23	Balanced Trees	Ch 10.4-10.5
39.	April 25	Catch up	
40.	April 27	Catch up	
41.	April 30	Catch up	
42.	May 2	Summary	

Programming Assignments and Laboratories

There will be two types of programming assignments: *laboratory programs* and *teacm programs*. All programs assigned during the semester should be completed following the guidelines in the *Academic Honesty Policy* described below.

There will be approximately 10 programs due. All programs will be graded on design, documentation and style (40%), correctness (40%), and efficiency (20%). Programs should be turned in electronically by 11:59 p.m. on the date due (though we typically don't impose the grade penalty for a program turned in within an hour of the deadline). There will be a penalty assessed of 3^n % for a program that is n days late. Programs will not be accepted more than four days late. It is usually better to turn in a correct and well-documented program one or two days late than a non-functioning or non-documented program on time.

Labs for this course will be held on Tuesday afternoons from 1:15 p.m to 2:30 p.m. and from 2:45 p.m. to 4 p.m. in 2189 Edmund. Those who are able are encouraged to come to both sessions. The room is equipped with iMac computers.

The purpose of the lab sessions is to provide you with hands on instruction in using some of the software tools and to provide a time during which your instructor can actively assist you in the development of *laboratory programs*. Attendance at these lab sessions is **mandatory**. You will submit your laboratory programs electronically. The procedure will be explained in laboratory. *Laboratory programs will generally be due by 11:59 P.M. on Thurday night after your lab session.*

Wednesday	Laboratory Title
Jan. 23	Java review
Jan. 30	Linked lists
Feb. 6	Analysis of Algorithms
Feb. 13	Stacks & Queues
Feb. 20	Eclipse debugging
Feb. 27	Trees
March 6	GUI Practice
Wednesday, March 7	**Midterm Exam in class**
March 20	Darwin I
March 27	Darwin II & C++
April 3	C++
April 10	C++
April 17	C++
April 24	Simulation?
May 1	Simulation continued?

Exams

There will be one in-lab midterm exam plus a scheduled final exam.

- *Midterm examination*: Monday, March 5, in class, or Tuesday, March 6, in lab. A sample exam will be available before the midterm.
- *Final examination*: Friday, May 11, at 9 a.m.

Grading Summary

Programs:			35–40%
Exams:			55%
	Midterm Exam:	25%	
	Final Exam:	30%	
Homework & quizzes:			5–10%
Total:			100%

Collaboration & Academic Honesty Policy

We highly encourage students to get together in small groups to go over material from the lectures and text, work practice problems from the text, study for exams, and to discuss the general ideas and approaches to laboratory assignments. However, work to be turned in, including programming assignments, must be done according to the rules in effect for that assignment.

When a program is assigned, we will identify it as either a "laboratory" or a "team" program. The academic honesty policy applies differently to each with respect to collaboration or assistance from anyone other than the mentors or instructors:

This course has two kinds of programming assignments, laboratory programs and team programs.

Laboratory Programs. Laboratory programs are expected to be the work of the individual student, designed and coded by him or her alone. Help locating errors is allowed, but a student may only receive help in correcting errors of syntax; help in correcting errors of logic is strictly forbidden. Guideline: Assistance from anyone other than the TAs or instructors in the design or coding of program logic will be considered a violation of the honor code.

Team Programs. Team programs are programs to be worked on in teams of two or more students. You are allowed to discuss team programs with your partners, but work with others is otherwise restricted by the appropriate rules above. Guideline: Any work that is not the work of your team is considered a violation of the honor code.

If you do not understand how the academic honesty policy applies to a particular assignment, consult with us. When in doubt, credit the people or sources from whom you got help. This also goes for any help obtained via the Internet. If you are ever unsure about what constitutes acceptable collaboration, just ask.

Failure to abide by these rules is considered plagiarism, and will result in severe penalties. Violations are easy to identify and will be dealt with promptly. The first offense typically results in failure in the course. A second offense is automatically referred to the College's Board of Academic Discipline. See the Academic Honesty Policy in the Student Handbook for further information. Please do not put us, yourself, or anyone else in this unpleasant situation.