CMSI 186
PROGRAMMING LABORATORY

http://myweb.lmu.edu/dondi/spring2008/cmsi186

Spring 2008 — Doolan 112 (Keck Lab)
TR 9:25–10:40am, 3 semester hours
Office Hours: TR 3–6pm or by appointment

John David N. Dionisio, PhD
e-mail: *dondi@lmu.edu,* AIM: *dondi2LMU*
Doolan 106; (310) 338-5782

Course Objectives

To attempt to perfect students' programming style and methodology by working closely, in a laboratory environment, on about seven medium-sized programs. The problem areas usually include discrete simulation, algorithms for arithmetic, dynamic programming, backtrack search, Riemann integration, randomized estimation, plotting numerical functions, and constructing simple graphical user interfaces (GUIs).

- We will begin to use diagrams to help clarify our designs, especially the various relationships among classes and objects.
- We will concern ourselves with the build environment; to this end, we will construct simple build scripts.
- We will pay close attention to unit testing, and we will design countless such tests as a routine part of program development.
- We will cover more of the language Java, including but not limited to multi-dimensional arrays, exception handling, event handling, and programming with graphical components.
- We will further explore issues of object-oriented programming that were only touched upon in CMSI 185 (e.g., constructing packages and classes, subclassing existing classes, overriding methods, etc.).

Course Requirements

Only computer science majors and minors may take this course, unless explicitly permitted by the instructor. CMSI 185, or an equivalent course using Java, must also have been completed with a grade of C or better.

Materials and Texts

There are no *required* texts; however, the following are recommended (latest editions are not necessary; sufficiently recent editions will work):

- David Flanagan, *Java in a Nutshell*, Fifth Edition, O'Reilly Media, 2005.
- David Flanagan, *Java Examples in a Nutshell*, Third Edition, O'Reilly Media, 2004.
- David Flanagan, *Java Foundation Classes in a Nutshell*, O'Reilly Media, 1999.

Additional information is also available on the Web; do not hesitate to look for further sources of information regarding the concepts, techniques, tools, and paradigms that we will discuss.

Course Work and Grading

Graded coursework consists solely of the programming problems assigned throughout the semester; the scores for each assignment will be weighted equally. Letter grades are determined as follows: $\geq$ 90% gets an A– or better; $\geq$ 80% gets a B– or better; $\geq$ 70% gets a C– or better. The instructor may curve grades upward based on qualitative considerations such as degree of difficulty, effort, class participation, time constraints, and overall attitude throughout the course. Grades are never curved downward.

Programming Assignments

A new programming problem will be assigned approximately every other week. For each one, we will discuss several possible algorithms, finally singling out one of them for implementation. I will then provide sketches of the important program structures, and your mission will be to complete the program. In addition to the code you write, you will submit a formal, plain-text lab report at the conclusion of each problem.

You will do a lot of programming during the workshop itself; the remainder is to be done as homework. Your grade will be determined according to two factors:

1. The overall quality of the programs and lab reports that you submit (see *Grading Criteria*).

2. Your class participation, including your attendance at the workshop and your preparation for them (see *Attendance*).

Consult the course schedule for projected submission deadlines. Submissions after the deadline receive *half* the grade that they would have gotten if submitted on time.

Collaboration with other classmates should be kept to a minimum. You may ask a classmate for help in diagnosing a syntax problem, or discuss the assignment in general terms; but, unless the assignment explicitly calls for pair programming, you may not share any code.

Grading Criteria: Source Code

1. *Design (30%):* How good is the overall structure of the code? Is it clear, flexible, and easy to maintain? Is it elegant or innovative? How well does it apply the principles of "separation of concerns" and "one change, one place?"
2. *Functionality (30%):* How well does the code work? Does it fulfill requirements? Are its results accurate or correct? Does it perform its tasks in a reasonable amount of time? How well do unit tests validate the code?
3. *Naming (20%):* Are program entities — classes, subroutines, variables, etc. — clearly and consistently named? Do their names correspond to their functions and roles?
4. *Comments (20%):* Are comments provided where appropriate? Are they clear and well-written? Does the code take advantage of any special support for comments provided by the project language or platform (e.g., JavaDoc)?

Grading Criteria: Lab Reports

1. *Content (40%):* What is the quality of the work? Does it contain the requested information for the assignment?
2. *Organization (30%):* Is the text structured well? Are its ideas and flow easy to follow? Are distinct sections or topics clearly identified?
3. *Writing (20%):* Are statements clear and easy to follow? Is the language precise and grammatically correct? Is the report's tone appropriate?
4. *Polish (10%):* Is the content properly proofread? Are there any misspellings, typos, or other formatting faux pas?

Attendance

Hands-on interaction with the instructor while programming is one of the major aspects of this course. Thus, regular attendance is expected; absences should be formally excused prior to the scheduled class time.

Remember that the university add/drop with 100% refund deadline is January 18. The deadline for withdrawal or credit/no-credit status is March 14.

University Policy on Academic Honesty

Loyola Marymount University expects high standards of honesty and integrity from all members of its community. Applied to the arena of academic performance, these standards preclude all acts of cheating on assignments or examinations, plagiarism, forgery of signatures or falsification of data, unauthorized access to University computer accounts or files, and removal, mutilation, or deliberate concealment of materials belonging to the University Library.

Course Schedule

This schedule, except for university dates (italicized), may change based on the actual ebb and flow of the class.

January 18	*Add/drop deadline for full refund*
January 31	Program 1 due
February 14	Program 2 due
February 28	Program 3 due
March 3–7	*Spring break; no class*
March 14	*Withdraw/credit/no-credit deadline*
March 20	Program 4 due
April 3	Program 5 due
April 17	Program 6 due
May 8	Program 7 due

You can view the class calendar on the Web at *http://ical.mac.com/dondi/LMU*. If you have an iCalendar-savvy client (i.e., Mozilla Calendar, Ximian Evolution, KOrganizer, Apple iCal, etc.), you can subscribe to the class calendar at *webcal://ical.mac.com/dondi/LMU.ics*. On-the-fly updates and adjustments to the class schedule will be reflected in this calendar.