CMSI 186-03

PROGRAMMING LABORATORY

http://myweb.lmu.edu/dondi/spring2017/cmsi186

Spring 2017—Pereira 142
TR 2:40–3:55pm, 3 semester hours
Office Hours: TR 4–6pm, W 2–4pm, or by appointment

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

Objectives and Outcomes

This course seeks to advance students' programming knowledge and experience in multiple directions: a new language and paradigm; increased programming problem size and complexity; and more rigorous testing and validation. This focus on programming advancement intrinsically involves a laboratory environment, apprenticeship-style training, and increased adherence to process and form. Long after the course concludes, my hope is that you will be able to:

1. **Design, compile, and test programs within a command-line environment.**
2. **Correctly solve medium-sized programming problems.**
3. **Follow academic and technical best practices throughout the course.**

Prerequisites/Prior Background

The prerequisite course is CMSI 185 or its equivalent. Students who have not taken (specifically) CMSI 185 require prior approval of the instructor.

Materials and Texts

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

- Benjamin J. Evans and David Flanagan, *Java in a Nutshell*, Sixth Edition, O'Reilly Media, 2014
- David Flanagan, *Java Examples in a Nutshell*, Third Edition, O'Reilly Media, 2004
- Assorted handouts, articles, and sample code to be distributed throughout the semester

In addition, do not hesitate to look for further information regarding the concepts, techniques, tools, and paradigms that we will discuss.

Course Work and Grading

Your final grade will be based on the percentage of the points you get for the following deliverables against the total number of possible points:

Java, GitHub, Piazza, and YouTube account setup	40 points
Java + GitHub commit practice	60
Java transition exercises	100
Classes and objects; board/parlor game logic	100
Discrete event simulation	100
Arithmetic first principles	100
Randomized estimation	100
Dynamic programming	100
Backtracking search	100
Total	800 points

Percentages ≥ 90% get an A– or better; ≥ 80% get a B– or better; ≥ 70% get a C– or better. I may nudge grades upward based on qualitative considerations such as degree of difficulty, effort, class participation, time constraints, and overall attitude throughout the course.

Programming Assignments

A new programming problem will be assigned approximately every few weeks. For each one, we will discuss several possible algorithms, finally singling out one of them for implementation. Sketches of key program structures will then be provided, and your mission will be to complete the program.

An assignment's number is its due date in *mmdd* format, and it is always due by 11:59:59pm of that date. Point values are based on the state of your assignments at that moment.

You will do a lot of programming during the workshop itself; the remainder is to be done as homework. Because the emphasis of this course is the development of *individual* programming skill, col-

laboration with other classmates should be kept to a minimum.

- You must be in front of a workstation during class, whether your own or the lab's.
- You may ask a classmate for help in diagnosing problems or discuss the assignment in general terms, but unless the assignment explicitly calls for pair programming, *you may not share any code*.

As programs increase in difficulty, their proficiencies will weigh more heavily. For simplicity, this weighting is reflected by counting that proficiency twice or more on the record.

Version Control

Version control is an indispensable part of today's computer science landscape in industry, the academe, and the open source community. We use version control heavily in this course: make sure that you get the hang of it.

None of the assignments can be completed (well) overnight; they should be the result of steady progress from the moment they are assigned to the date they are due. "One and done" submissions will negatively affect the final score.

As an additional incentive for making steady progress early and often, an automated QA (quality assurance) system will be connected to your assignment repository, described in the next section.

Automated Quality Assurance System

The use of version control enables *continuous integration* (CI). CI monitors your assignment repository and can trigger assorted automated activities when new work arrives. For this course, CI performs automated QA on your source code.

Submit your work-in-progress to have the system identify areas of improvement. These areas will appear in a QA report that will be linked to your commits. Eliminate these errors and warnings and make sure your programs work correctly "out of the box" by the due date to maximize your points.

I reserve the right to discontinue further evaluation of your work if the severity or amount of QA flags makes the process too difficult or disruptive.

Workload Expectations

In line with LMU's *Credit Hour Policy*, the workload expectation for this course is that for every one (1) hour of classroom instruction (50 scheduled minutes), you will complete at least two (2) hours of out-of-class work each week. This is a 3-unit course with 3 hours of instruction per week, so you are expected to complete $3 \times 2 = 6$ hours of weekly work outside of class.

Attendance

Attendance at all sessions is expected, but not absolutely required. If you must miss class, it is your responsibility to keep up with the course. The last day to add or drop a class without a grade of W is January 13. The withdrawal or credit/no-credit deadline is March 17.

Academic Honesty

Academic dishonesty will be treated as an extremely serious matter, with serious consequences that can range from receiving no credit to expulsion. It is never permissible to turn in work that has been copied from another student or copied from a source (including the Internet) without properly acknowledging the source. It is your responsibility to make sure that your work meets the standard of academic honesty set forth in:

http://academics.lmu.edu/honesty

Special Accommodations

Students with special needs who require reasonable modifications or special assistance in this course should promptly direct their request to the Disability Support Services (DSS) Office. Any student who currently has a documented disability (ADHD, autism spectrum, learning, physical, or psychiatric) needing academic accommodations should contact DSS (Daum 224, x84216) as early in the semester as possible. All requests and discussions will remain confidential. Please visit *http://www.lmu.edu/dss* for additional information.

Topics and Important Dates

Correlated outcomes are shown for each topic. Specifics may change as the course progresses. University dates (italicized) are less likely to change.

January	Introduction to the Java programming language *(1a–1c, 3a–3f)*
January 13	*Last day to add or drop a class without a grade of W*
February	Java transition exercises *(1a–1c, 3a–3f)*; board/parlor game logic programs *(all outcomes, 1a–3f, from this point forward)*
March	Discrete event simulation programs; implementation of arithmetic and geometry routines
March 6–10	*Spring break; no class*
March 17	*Last day to withdraw from classes or apply for Credit/No Credit grading*
March 31	*Cesar Chavez Day*
April	Dynamic programming; backtracking search
April 12–14	*Easter break; no class*

You can view my class calendar and office hour schedule in any iCalendar-savvy client. Its subscription link can be found on the course web site (it's too long to provide in writing).

If necessary, this syllabus and its contents are subject to revision. Students are responsible for any changes or modifications announced in class.

Tentative Nature of the Syllabus

If necessary, this syllabus and its contents are subject to revision; students are responsible for any changes or modifications distributed in class or posted to the course web site.

Course Outcomes

1 Design, compile, and test programs within a command-line environment.

1a	*Set up and maintain an operational, productive programming environment.*	These outcomes represent first-principle programming *fundamentals*. Developer tools abound for improving the efficiency and productivity of experiences programmers…emphasis on *experienced*. Such tools presume pre-existing knowledge and understanding—*these very outcomes*.
1b	*Edit and manage source code effectively.*	
1c	*Issue correct commands for common programming tasks.*	

2 Correctly solve medium-sized programming problems.

2a	*Adhere to problem specifications.*	As the types of programs that you write increase in complexity and size, *discipline* and *good habits* factor more greatly into your success. These outcomes seek to reflect this expansion.
2b	*Gracefully handle errors and edge cases.*	
2c	*Pass automated unit and functional tests.*	

3 Follow academic and technical best practices throughout the course.

3a	*Write syntactically correct, functional code.*	Code has to compile. Code has to work. No errors, no bugs. Use unit tests as much as possible.
3b	*Use coding best practices, demonstrating principles such as DRY, proper separation of concerns, correct scoping of variables and functions, etc.*	This is the basis of good software design. It makes software easier to maintain, improve, and extend. In this course, these practices will be introduced as the class progresses. Heed feedback well.
3c	*Write code that is easily understood by programmers other than yourself.*	This outcome involves all aspects of code readability and clarity for human beings, including but not limited to spacing & indentation, proper naming, presenting code in a manner that is consistent with its structure, documentation & comments when appropriate, and adherence to conventions or standards.
3d	*Use available resources and documentation to find required information.*	The need to look things up never goes away. Remember also that the course instructor counts as an "available resource," so this outcome includes asking questions and using office hours.
3e	*Show steady progress by using version control as described by the protocol given in the course.*	This course introduces version control in a specific way, to acquaint you with its principles and uses without information overload. Industry-standard use comes in later courses.
3f	*Meet all designated deadlines.*	