COMP 333: Concepts of Programming Languages
Fall 2019

Instructor: Kyle Dewey (kyle.dewey@csun.edu)
Course Web Page: https://kyledewey.github.io/comp333-fall19/
Office: JD 4419, Extension 4316 (not yet connected)

Course Description (From the Catalog)
Discussion of issues in the design, implementation and use of high-level programming languages through a historical framework, including how languages reflect different design philosophies and use requirements and the technical issues in the design of main abstraction constructs of programming languages. Other approaches to imperative or object-oriented programming, functional programming, logical programming and parallel programming.

Learning Objectives:
Successful students will be able to:
- Understand when to use, and write programs using:
  - Execution approaches: compilation, interpretation, and just-in-time (JIT) compilation
  - Memory management: garbage collection, reference counting, ownership
  - Types: dynamic typing, static typing
  - Parameter passing: call by value, call by reference, call by name
- Read context-free grammars, construct abstract syntax trees, and parse programs
- Write object-oriented programs using:
  - Object-oriented classes
  - Virtual dispatch
  - Inheritance
- Write functional programs using:
  - Higher-order functions
  - Algebraic data types and pattern matching
  - Typeclasses (NOT object-oriented classes)
  - Generics and parametric polymorphism
- Write logic programs using:
  - Unification
  - Nondeterminism
- Write concurrent, low-level, memory-safe programs using:
  - Affine types
  - Threads

Course Motivation and Goal
Programming languages, like the tools in a typical toolbox, are built to solve problems. A toolbox may have different sizes and shapes of both hammers and screwdrivers. Similarly, different programming languages may be closely related to each other, or

potentially very different from each other. The more different a language or tool, the more different the kind(s) of problem(s) it is designed to solve.

The danger of getting to close to one language or programming paradigm is that your thinking adapts to fit that language/paradigm. In keeping with the toolbox analogy, you have a hammer, and all problems become nails. I can insist on fixing a leaky pipe with a sledgehammer, but I won't get back my security deposit.

My primary goal with this course is to expand your toolbox, and expose you to languages which behave very differently from each other. My intention is to warp your brain a bit, and force you to think in ways you're not used to. This will improve your problem-solving skills, specifically by allowing you to look at the same problem from different angles.

A secondary goal is to give you a sense of how different languages are built, and how they work. We will focus primarily on modern language design and implementation, though many basic concepts haven't changed much in the 60+ year history of programming languages.

Textbook
No textbooks are required. You may wish to look at Programming Language Pragmatics (Michael Scott) as a reference, though the course does not follow that book.

Grading
Your grade is based on the following components:

Normal Assignments	40%
Midterm Exam 1	15%
Midterm Exam 2	15%
Presentation	5%
Final Assignment	10%
Final Exam	15%

There will be several normal assignments issued throughout the semester, which cover core parts of the different languages and paradigms you'll use. Not all of these will be weighted evenly, nor will you always be given the same amount of time for assignments. Exactly which assignments are assigned depends on how the class progresses. In general, assignments will be submitted through Canvas (https://canvas.csun.edu/). In the event that there is a problem with Canvas, you may email your assignment to me (kyle.dewey@csun.edu), though this should be considered a last resort.

As part of the course, you will need to learn at least some parts of a language not used in the course, and do a short presentation covering the design, implementation, use cases, pros, and cons of the language. You will also reimplement a prior assignment in that language (the final assignment).

Plus/minus grading is used, according to the scale below:

If your score is >=...	...you will receive...
92.5	A
89.5	A-
86.5	B+
82.5	B
79.5	B-
76.5	C+
72.5	C
69.5	C-
66.5	D+
62.5	D
59.5	D-
0	F

If you are not present for the final exam and you have not previously made alternative arrangements with me for the final exam, a grade of WU (unauthorized withdrawal) will be assigned.

Collaboration for Assignments
All students are required to submit their own individual work. For assignments (and **only** assignments), students may discuss among each other, as long as they don't digitally share code. That is, you **cannot** simply email your code to someone else. However, you **may** discuss your actual code with someone else, including viewing the code on a monitor. The only stipulation is that **if you do discuss with someone else, say so in your submission.** This is not for punitive reasons; this is only so I get a sense of who is working with who. My intention with this policy is to enable collaborative learning, as opposed to simply sharing a solution.

Plagiarism and Academic Honesty
While collaboration is allowed on assignments, you are responsible for all of your own work. You may **not** take code from online sources and submit it as your own. No discussion whatsoever is allowed during exams, except with the instructor. Any

violations can result in a failing grade for the assignment, or potentially failing the course for egregious cases. A report will also be made to the Dean of Academic Affairs. Students who repeatedly violate this policy across multiple courses may be suspended or even expelled.

Attendance
In the first week of class, I will take attendance. If you miss both sessions in the first week and have not made alternative arrangements with me, you must drop the class, as per University policy (http://catalog.csun.edu/policies/attendance-class-attendance/). After the first week I will not take attendance, and attendance is not mandatory, though you are strongly encouraged to attend.

Communication
You're encouraged to use Canvas discussions to ask questions, as long as the questions don't involve your specific solution. This way, anyone in the class can answer the question, and everyone can benefit from the answers. For anything else, email me directly at kyle.dewey@csun.edu.

Late Policy / Exam Scheduling
Late assignments will be accepted without penalty if prior arrangements have been made or there is some sort of legitimate emergency (at my discretion). If you must be absent from an exam, contact me ASAP to see if alternative accommodations can be made.

If an assignment is otherwise submitted late, it will be penalized according to the following scale:

If your assignment is late by <= this many days...	...it will be deducted by...
1	10%
2	30%
3	60%
4+	100%

To be clear, assignments which are submitted four or more days beyond the deadline will not receive credit. The reason for such a harsh late policy is that we will generally discuss solutions in class shortly after the deadline, and this late policy discourages people from simply pulling a solution from an in-class discussion.

Class Feedback
I am open to any questions / comments / concerns / complaints you have about the class. If there is something relevant you want covered, I can push to make this happen. I operate off of your feedback, and no feedback tells me “everything is ok”. This is the

first time I'm teaching this course, and it is the first time the course has had this particular structure, so I'm anticipating that it won't all be smooth sailing.

Class Schedule (Subject to Change):

Week	Monday	Wednesday
1	8/26: Introduction, motivation	8/28: OOP introduction, inheritance, virtual dispatch
2	~~9/2~~: Labor Day (no class)	9/4: Dynamic typing, reference counting, garbage collection
3	9/9: FP introduction, higher-order functions	9/11: Higher-order functions, generics, parametric polymorphism
4	9/16: List operations	9/18: BNF grammars, abstract syntax trees
5	9/23: Tokenization, parsing	9/25: Algebraic data types, pattern matching
6	9/30: Combinators (advanced higher-order functions)	10/2: Call by value vs. call by name
7	10/7: Typeclasses	10/9: Typeclasses
8	10/14: **Midterm 1**	10/16: Midterm 1 retrospective, interpreters, compilers, JIT compilers
9	10/2: Introduction to LP	10/23: Nondeterminism, backtracking
10	10/28: Structures, unification	10/30: FP vs. LP, Lists
11	11/4: Parsing in LP	11/6: Introduction to concurrency
12	11/11: Introduction to Rust	11/13: Ownership/affine types, lifetimes, borrowing
13	11/18: Generics in Rust	11/20: Typeclasses in Rust
14	11/25: Concurrency in Rust	11/27: Extra time buffer
15	12/2: **Midterm 2**	12/4: Midterm 2 retrospective, class presentations

Week	Monday	Wednesday
16	12/9: Class presentations	~~12/11~~: No class; lectures over

Final exam times:

- Section 1: Monday, 12/16/19, 12:45 PM - 2:45 PM, JD1600
- Section 2: Monday, 12/16/19, 3:00 PM - 5:00 PM, JD 2221