Introduction to Programming System Design

CSCI 455x (4 Units)

Description This course covers programming in Java and C++. Topics include review of basic programming concepts such as control structures, functions, and arrays; coverage of more advanced programming topics such as classes, recursion, inheritance, and linked lists; use of a container class library to program with tools such as a map class and a sort function; and an introduction to algorithm analysis. There will also be an emphasis on good development techniques such as good code style and documentation, unit testing and use of debugging tools. A second goal of the course is to introduce the Linux programming environment, including tools such as the shell, simple shell scripts, and makefiles.

Prerequisite: minimal programming experience in some high-level language (can program with loops, if statements, and call and write functions with parameters)

Instructor Claire Bono

Contact Info bono@usc.edu

Lecture 3 hours / week (on zoom)

Lab 2 hours / week (on zoom)

Textbook *Big Java Early Objects Enhanced eText, 7th ed.*, by Cay Horstmann, Wiley, ISBN 978-1-119-49909-1

Available on vitalsource.com (only ~$39 for a 4-month rental).

First lecture and lab Link for online lecture will be available on d2l under Access to Online Lecture. A link to the first lecture meeting will also be sent to enrolled students by sometime Monday, Jan. 18.

No lab will be held on Friday, January 15: labs will start the following week (1/21 – 1/22).

Assignments Programming assignments are graded on thorough testing, documentation, and style, as well as correctness. All work to be submitted for the class is to be done individually unless an assignment specifies otherwise.

Late policy for programming assignments. You may turn in a program up to two days late for a penalty of 10% of the available points. So, for example, if you would have gotten a 70/100, you will get 60/100 instead (not 63). After this two-day grace period, a late program receives no credit.

Computing environment You will be using the Vocareum cloud-based environment for program development, to be introduced in the first lab. In addition, you may use an environment local to your machine (e.g., IntelliJ, Eclipse, Visual C++, or command line plus text editor) for developing

programs. Note: Because licensing for your own copy of Java has recently changed, you will need a version of OpenJDK 1.8, rather than Oracle's JDK. For more information, see documents linked in the Java section of the Documentation course web page. **For programs you develop locally, you are responsible for making sure your code compiles and runs on the Vocareum environment before submitting, because we will be grading your programs there.** Vocareum uses Java 8 (aka, 1.8), and g++ 5.5.0.

Labs The lab is intended for practicing some of the techniques learned in class on the computer in an environment where you can get immediate help from a lab assistant.

Labs meet online once a week for two hours. This semester, courses start on Friday 1/15, but we will not hold the first labs until the following week (1/21 – 1/22). Every week you will be given the lab exercises a few days before the lab: some require some advance preparation. You may work on or complete the lab exercises before the lab period if you wish, but they are due **during** your lab section. If you finish early, you are free to leave (once you get the lab checked off) or spend the rest of the time working on your other CS 455 assignments.

The lab section offerings should make it possible for synchronous participation by remote students participating from US and Asian time zones and some other locations. However, any student (local or remote) with an ongoing hardship attending labs can be granted special accommodations: for those students please email the instructor with information about your specific situation once the semester starts.

Each set of lab exercises can earn you up to 3 - 5 points. There be will up to roughly 50 lab points total. To take some of the pressure off the lab score only 80% of the available points are applicable towards your final score in the class (but scaled to be worth 10% of the total course score). This gives you some leeway if you have to miss a lab, or if you don't have time to solve all of the problems in the two-hour session. Accordingly, if you have to miss a lab, it's not necessary to contact the instructor.

Den students. Den students will complete their labs remotely, and submit them electronically. Den students do not have to be available during the lab session. They can get help on the lab or other assignments through zoom online office hours, or by posting to the piazza discussion board, or email to course staff.

Exams All exams are closed book, closed note. Makeup exams will *not* be given. Absence due to a *serious* illness will be an acceptable reason for missing an exam, and the final grade will be scaled accordingly. The exam dates will be announced the first day of class.

Website *bytes.usc.edu/cs455*
This is where you will get most of your course materials, and has links to the following other platforms we are using: d2l (for videos of lectures and grades), Vocareum (for program development and grading), piazza (for electronic discussions), and zoom (for lectures, labs, and office hours).

Grading The following is the relative weight of each part of the course work. At the end of the semester, you will have a score out of 100 percent. This score will be used in a class curve to arrive at a letter grade. I guarantee that >=90 will be some kind of A, >=80 will *at least* be some kind of B, >=70 will *at least* be some kind of C, and that >=60 will *at least* be some kind of D.

Participation	5%
Programming assignments	30%
Labs	10%
Midterm Exam 1	10%
Midterm Exam 2	15%
Final Exam	30%
Total	100%

Policy on regrades (e.g., if you think there was a scoring mistake on your work): **you have until one week from when you get the graded work back** to initiate a regrade. We'll discuss the exact procedure for requesting a regrade once the course starts.

Academic Integrity

The USC Student Conduct Code prohibits plagiarism. All USC students are responsible for reading and following the Student Conduct Code, which appears in the sections on University Student Conduct Code (sections 10.00-16.00) in the current version of *SCampus*. The relevant part of *SCampus* is available on the web at https://policy.usc.edu/scampus-part-b/.

In this course we encourage students to study together. This includes discussing high-level general strategies to be used on individual assignments. But it would not, for example, include jointly developing pseudo-code for an assignment solution with another student. All work submitted for the class is to be done individually, unless an assignment specifies otherwise. Also, all exams are closed book, closed note.

Some examples of what is **not** allowed by the conduct code: copying all or part of someone else's work and submitting it as your own, giving another student in the class a copy of all or part of your assignment solution, or pseudo-code for a solution, making an assignment solution available to other students (for instance, putting it in a public github repository), consulting with another student during an exam, using a solution or adapted solution to an assignment that you found on the web. The outside code resources students will be allowed to use in assignments for this class are limited to code written by the course staff for the purposes of helping students in the course, or code from the textbook for this course. If you do use any such code not written by you, you are required to acknowledge your sources in your README file. If you have questions about what is allowed, please discuss it with the instructor.

Because of past problems with plagiarism in this and other computer science courses, we may be running all submitted programming assignments through sophisticated plagiarism-detection software.

Violations of the Student Conduct Code will be filed with the Office of Student Judicial Affairs and Community Standards (SJACS) (for undergraduates) or the Viterbi Academic Integrity Coordinator (for graduate students), and appropriate sanctions will be given. *The sanctions are usually a lot more severe than not submitting the assignment.*

Students with Disabilities

Any student requesting academic accommodations based on a disability is required to register with Disability Services and Programs (DSP) each semester. A letter of verification for approved accommodations can be obtained from DSP. Please be sure the letter is delivered to the instructor as early in the semester as possible. DSP is located in STU 301

and is open 8:30 a.m. - 5:00 p.m., Monday through Friday. The phone number for DSP is (213)740-0776.

Introduction to Programming System Design
CSCI 455x (4 Units)

Course Outline

Note: the exact lab assignments and order of topics may vary a little from this example.

Computing environment basics (1 lecture)
- Basic Linux commands
- Compiling and running Java programs on Linux
- Output and computation in Java

Lab: Development environment: basic Linux commands, compiling and running java programs

Using objects (2 lectures)
- Objects and object references
- Constructing objects
- Methods and method calls: accessors and mutators
- Primitive values; Strings
- Reading Java API documentation
- Examples: PrintStream, String, Rectangle, and Scanner classes

Lab: Write a program using a Java class; use Java documentation

Implementing classes (1 lecture)
- Instance variables
- Method definitions
- Scope and lifetime of variables
- Public interface vs. private
- Constructors
- Test programs
- Example: Student class

Lab: Implement a simple class to a specification

Control structures (1 lecture)
- If, while, for
- Boolean expressions
- Short-circuit evaluation, DeMorgan's law
- Error-checking input
- Multi-way tests
- Dangling else

Arrays and Array Lists (2.5 lectures)
- Random access in arrays; ex: counting scores
- Partially filled arrays
- ArrayList class
- Arrays of objects
- Ex: array operations in Names class

- Incremental development
- Test-driven design
- Code refactoring

Lab: Enhance a small program with loops and ArrayList

More on designing and defining classes (2 lectures)

- A class represents a single concept
- When static methods are used
- Methods: preconditions and postconditions
  - Assert statements
- Instance variables vs. locals: minimize scope
- Class invariants
  - Testing implementation invariants
- Parameter passing
- Methods with side-effects
  - Defining immutable classes
  - Returning references from inside objects
  - Copying objects

Lab: Test assert statement; write invariants; line-oriented input

Algorithm analysis and big-O notation (1 lecture)

- Constant, linear and quadratic time
- Big-O of earlier examples
- Merge algorithm
- Finding big-O of Java library methods

Lab: Use debugger on supplied buggy program

Recursion (2 lectures)

- Thinking recursively
- helper functions
- computational complexity of recursive functions
- tree recursion
- backtracking

Lab: Write some recursive routines

Linear Container classes (2 lectures)

- java.utils LinkedList
- Lists vs. arrays
- Iterators
- Stacks
- Queues

Lab: Modify program using java LinkedList class
Lab: Empirical comparisons of (1) list vs. array implementation of a sequence, and (2) linear vs. binary search on an array

Inheritance and Interfaces (1 lecture)

- Examples of inheritance
- Inheritance in Java graphics programs

- Overriding Object methods: clone, toString, equals
- Interfaces: ex: sorting and Comparable interface

Lab: Implement sort Comparator

Reading and Writing Text files; Exception handling (1 lecture)
- Scanners and PrintWriters
- Checked and unchecked exceptions
- Throw and catch exceptions

Lab: read command-line arguments, use exceptions for error-conditions.

Maps, Sets, and Sorting (4 lectures)
- Java Map and Set interfaces
    - Iterating over a Map or Set
    - Ex: concordance
- Binary search and log n time
- Overview of binary search trees
- Hash tables
    - Hash functions
    - Collision resolution
    - Applications
    - Big-O
- Sorting: insertion sort and mergesort
- Comparison of Map implementations
- Java sort methods, Comparable interface

Lab: Implement concordance using a Map; sort results by number of occurrences

(C++) Differences between C++ and Java (2 lectures)
- Running g++ compiler
- I/O
- Stand-alone functions
- Parameter passing
- Fixed-size arrays
- C++ object model
- Defining classes

Lab: Use C++ vectors

(C++) Dynamic data, pointers, and linked lists (3 lectures)
- Pointers and memory
- Delete
- Pointers to objects
- Linked lists
- Dynamic arrays
- C strings
- Pointer arithmetic

Lab: C++ debugger; implement various linked list functions

Separate compilation and make (2 lectures)
- Compilation units

- Header files
- Forward declarations
- Makefiles

Lab: write a makefile; convert a single file program into a multi-file program