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 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 Unix programming environment, including tools such as the shell, simple shell scripts, and makefiles.

Prerequisite: minimal programming experience in a high-level language

Instructor Claire Bono

Contact Info bono@usc.edu | SAL 310 | 213-740-4510

Lecture 3 hours / week

Lab 2 hours/ week

Textbook *Big Java*, 4th Ed., by Cay Horstmann, Wiley 2010,
ISBN 978-0-470-50948-7

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 the two day grace period, a late program receives no credit.

Computing environment All submitted programs must compile and run on aludra Sun java compiler for java programs and the g++ compiler for C++ programs. Aludra is a time-share Unix computer on the SCF file system. You can access it remotely from PC's on or off campus using the x-win32 software, or from Macs using the X11 application. You can also access the SCF file system and the compilers using Sun Unix Workstations in Sal 125. The first lab will be focused on introducing the programming environment.

If you choose to develop your programs on your own computer using another environment (e.g., Eclipse or Visual C++) you are responsible for making sure your code compiles and runs on the SCF environment before submitting.

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

Labs meet once a week for two hours. They will start the first week of classes. You will be given the lab exercises a few days before the lab: some require some advance preparation. You may 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.

Each set of lab exercises usually can earn you up to 4 points. There be will up to roughly 40 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 one day.

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 the same way they do for other assignments: generally via email (whenever) or by phone (during office hours) with someone on the 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 *https://www-scf.usc.edu/~csci455*

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.

Programming assignments	30%
Labs	10%
Midterm Exam 1	10%
Midterm Exam 2	20%
Final Exam	30%
Total	100%

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 Governance (sections 10.00-16.00) in the current version of *SCampus* and on the web starting from: http://web-app.usc.edu/scampus/university-student-conduct-code/ (links to subsequent sections are at the left side of the page).

In this course we encourage students to study together. This includes discussing general strategies to be used on individual assignments. However, 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 your assignment solution, consulting with another student during an exam. 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), and appropriate sanctions will be given.

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.

(continued next page)

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

Course Outline

Computing environment basics (1 lecture)
- Basic Unix commands
- Compiling and running Java programs on Unix
- Simple I/O
- Expressions, precedence
- Simple types, variables

Lab 1: Development environment: basic Unix commands, compiling and running java programs

Using objects (2 lectures)
- Examples: PrintStream, String, and simple graphics classes
- Objects and object references
- Constructing objects
- Method calls, implicit parameter and explicit parameters, return value
- Reading Java API documentation

Lab 2: Simple computation with console I/O and dialog box I/O

Implementing classes (1 lecture)
- Instance variables
- Methods
- Constructors
- Public interface
- Javadoc

Control structures (1 lecture)
- If, while, for
- Boolean expressions
- Short-circuit evaluation
- Error-checking input
- Common loop algorithms

Lab 3: Implement a simple class to a specification

Arrays and Array Lists (2 lectures)
- Random access in arrays
- Partially filled arrays
- ArrayList class
- Arrays of objects
- Common array algorithms

Lab 4: Enhance a small program with loops and ArrayList (e.g., manage scores for a student)

More on designing and defining classes (2 lectures)
- Desirable qualities of classes
    - Independence among classes

    - Data encapsulation
- methods
    - accessors, mutators, and constructors
    - Preconditions and postconditions
- Instance variables vs. locals
- Class invariants
- Static methods and variables

Lab 5: Implement a class to a specification (e.g., re-implement Lab 3 as a Scores class)

Testing (1 lecture)
- Unit testing
- Stubs
- Shell-scripts for testing
- Redirecting input and output in Unix
- Regression testing

Lab 6: Use debugger on supplied buggy program

Lab 7: Implement test driver for a class; write a shell-script for testing

Inheritance and Interfaces for GUI programming (2 lectures)
- Inheritance in Java Swing and AWT libraries
- Dynamic binding
- callbacks
- Inner classes
- Events and Listeners

Lab 8: Modify program with buttons and listeners

Reading and Writing Text files; Exception handling (2 lectures)
- Scanners and PrintWriters
- Command-line arguments
- Checked and unchecked exceptions
- Throw and catch exceptions

Lab 9: read and write a file, use exceptions for error-conditions.

Linear Container classes (2 lectures)
- Java.utils LinkedList
- Lists vs. arrays
- Stack
- Queue

Lab 10: Modify program using java LinkedList class

Algorithm analysis and big-O notation (1 lecture)
- Constant, linear and quadratic time
- Big-O of earlier examples
- Log n time example: binary search

Lab 11: Empirical comparisons of (1) list vs. array implementation of a sequence, and (2) linear vs. binary search on an array

Maps, Sets, and Sorting (3 lectures)

- Hash tables
  - Hash functions
  - Collision resolution
  - Applications
- HashMap vs. TreeMap
- Java sort methods, Comparable interface

Lab 12: Implement concordance using a map; sort results by number of occurrences

(C++) Differences between C++ and Java (3 lectures)

- Running g++ compiler
- I/O
- Stand-alone functions
- Parameter passing
- Object model

Lab 13: Re-implement an earlier lab in C++

(C++) Dynamic data, pointers, and linked lists (3 lectures)

- Pointers and memory
- Delete
- Array syntax
- Dynamic arrays
- Linked lists

Lab 14: Implement various linked list functions

Separate compilation and make (1 lecture)

- Compilation units
- Header files
- Forward declarations
- Makefiles

Lab 15: write a makefile; convert a single file program into a multi-file program