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, 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 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 | SAL 310 | 213-740-4510

Lecture 3 hours / week

Lab 2 hours / week

Textbook *Big Java Early Objects, 6th ed.,* Binder Ready + Interactive eText, by Cay Horstmann, Wiley ISBN 978-1-119-26380-7

or

Big Java Early Objects Interactive Edition, 6th ed., by Cay Horstmann, Wiley, ISBN 978-1-119-14159-4

Note: the first ISBN is for an unbound binder-ready version plus a code for the interactive electronic version, available from the campus bookstore, and the second one is the electronic version only (about half the price for a 4-month rental) which you can buy at vitalsource.com.

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 or XQuartz application. 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 for practicing 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 or other assignments through the designated DEN office hours (held through the bluejeans videoconferencing software), or by posting to the d2l discussion boards, 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

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%

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. For programming assignments: if you have questions about your grade, send email to the grader (his/her name should appear at the bottom of your detailed score file). For exams: please come see the instructor in person.

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 your assignment 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), 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.

(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
- Output and computation in Java

Lab: Development environment: basic Unix 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