Master Syllabus for TCSS 1412
Version: September 2009

Prerequisites

one quarter of pre-calculus

Preconditions

- basic computer literacy
  - able to use a text editor / word processor
  - exposure to basic computer organization concepts
    - operating system and processes
    - files and folders
- basic problem-solving skills
  - ability to extract the essence of a word problem involving a calculation
  - ability to formulate an algorithm to solve such a problem, and solve it by hand

Topics Covered for Comprehension and Application

- basic computer operation: how writing and executing Java applications (and applets) fits into the larger computing environment
- basic programming concepts
  - source code versus byte code versus object code
  - compilation, interpretation, and execution
  - the concept of a function / procedure /method
  - coding and documentation standards
- basic class structure
  - components of a class definition
  - constructors and instances
  - instance variables and methods
  - static variables and methods
- variables and parameters
  - variable definition: name, type, and scope
  - reference versus value distinction
- operators and expressions
  - primitive data types and their operators (arithmetic, comparison, logical)
  - operator precedence
- control structures and iteration
  - conditionals
  - loops
  - method / procedure calls and parameter passing
  - control flow via procedure / method calls (procedural decomposition)
- collections
  - single-dimensional arrays
  - abstract collections implementing arrays
- input/output

- input and output from the console
- opening, reading, writing, and closing files, given a simple file IO class and its public interface

Topics Covered for Knowledge and Comprehension

- detailed class structure
  - design of multiple interacting classes, including decision about public versus private methods and static versus member methods
- inheritance
- interfaces
- Javadoc documentation
- throwing exceptions for non-local control (e.g. on bad input)
- multi-dimensional arrays

Postcondition Skills

- basic computer and programming concepts
  - write, edit, document and debug a simple Java application
  - implement a basic IO/processing loop
    - example: take multiple data values as input from console, filter invalid values, perform calculation, write result to console
- facility using fundamental Java data types: numbers, characters, strings, Booleans
- ability to implement basic algorithms. (e.g., sequential search, find the maximum item in an array, compute the average of elements in an array)
- facility using single-dimension arrays
  - example: load values into a single-dimension array, with capacity checking and error handling, then subsequently being able to find a value in that array
  - example: shuffle a deck of cards stored in an array
  - example: reverse the elements in a one-dimensional array
  - above examples for single-dimension arrays, using an `ArrayList` implementation
- use a simple class, given only its API (e.g. for file IO)
- class definition: define a class, including static member functions and variables, overriding methods from the `Object` class, and using that class in an application
  - example: an *Employee* class with unique IDs and getters and setters for name and job title, getters only for ID and salary, a *giveRaise* method, and overriding the *equals* and *toString* methods. Read a list of employees from information in a file, and print the list sorted in ascending order by ID number, then print the list sorted in descending order by salary
- class implementation: construct a class that implements a given interface
  - example: write an ordered pair class with constructors and arithmetic operators specified in interface only

- problem statement to implementation: given a problem statement, able to cast the problem in OO/Java terms and write a short (100 lines) solution from a blank screen or skeletal code
- multiple-class solutions: can write a multi-class program if classes are prescribed
    - example: a *Student Registration* program that accepts from the user several pieces of information about students, creates *Student, Course,* and *Registration* objects and places them into an array. Array is then scanned to produce class lists
- program comprehension and modification
    - able to determine what a relatively conceptually simple program does
    - able to make minor modifications to a program with the purpose of fixing a bug or to adapt the program to a different but similar purpose