TCSS 143 Master Syllabus
(Approved: June 2, 2017)
(Effective: Autumn 2017)

Course Title

Fundamentals of Object-Oriented Programming Theory and Application

Catalog Description

Develops fundamental concepts and techniques for analysis, design, and implementation of computer programs using an object-oriented language. Includes recursive techniques and use of abstract data types. Prerequisite: a minimum grade of 2.0 in either TCSS 142, CSE 142, or TCES 201.

Preconditions

- Develop and implement programs involving the fundamental programming constructs (variables, types, expressions, assignment, simple I/O, conditional and iterative control structures, functions and parameter passing, structured decomposition).
- Develop and implement programs that use each of the following: arrays, strings, and objects.

Student Learning Goals (to be added to syllabus handed out to students)

- Implement a low-complexity program (3 or more interacting classes) which includes the use of interfaces and/or abstract classes, and polymorphism given some design guidance.
- Apply object-oriented design concepts and techniques such as inheritance, composition, encapsulation, abstraction, method overloading, method overriding, exception handling, and scope appropriately to the implementation of a program.
- Provide formal documentation for a program, e.g., Javadoc comments
- Use single and multidimensional arrays and basic abstract data types (lists, queues, sets) in the implementation of a program.
- Use a provided class given only its API
- Apply recursive techniques to solve problems

CSS Degree Student Learning Outcomes that this course contributes to (to be added to syllabus handed out to students)

a. an ability to analyze a problem, identify and define the computing requirements appropriate to its solution;

b. an ability to design, implement and evaluate a computer-based system, process, component, or program to meet desired needs;
c. an ability to use current techniques, skills, and tools necessary for computing practice.
d. Explain what is meant by “best”, “expected”, and “worst” case behavior of an algorithm. (from AL)
e. Determine informally the time and space complexity of simple algorithms. (from AL)
f. Use recursive backtracking to solve a problem such as navigating a maze. (from AL)

UWT Student Learning Goals that this course contributes to (to be added to syllabus handed out to students)

Inquiry and Critical Thinking
Students will acquire skills and familiarity with modes of inquiry and examination from diverse disciplinary perspectives, enabling them to access, interpret, analyze, quantitatively reason, and synthesize information critically.

Topics covered

- review of basic programming concepts with emphasis on memory models, method declaration (with reference to pass by value vs. pass by reference)
- review of single dimensional arrays
- introduction to 2D arrays
- basic matrix operations
- basic object-oriented programming concepts (classes, objects, encapsulation, abstraction, cohesion)
- inheritance, interfaces, abstract classes, polymorphism
- use of ADT lists, stacks, queues, sets, and maps
- recursion, recursive backtracking
- searching (sequential and binary)
- sorting (including identification of various quadratic (such as selection, insertion, bubble) and efficient sorts (such as mergesort, quicksort, etc))

Additional Information

This course has an associated lab section that meets once per week in addition to the 2 lectures per week.

There is an optional 2 credit seminar workshop (TCSS 390A) associated with this course.

The textbook used in recent years:

Building Java Programs Second Edition, Stuart Reges and Marty Stepp, Addison Wesley, ISBN-10: 0-13-609181-4