# CS 2510: Course Charter & Syllabus

Jay Aslam, Will Clinger, David Van Horn
College of Computer and Information Science
Northeastern University

February 21, 2012

## 1 CS 2510 Course Charter

**Course Title:** CS 2510 *Fundamentals of Computer Science 2* (4 SH)

**Course Description:** Introduces the fundamentals of the design of class hierarchies in object-oriented programming with emphasis on representation-dependent program design with objects. The course also covers basic reasoning techniques concerning the correctness and algorithmic behavior of computations.

**Prerequisites:** CS 2500. In addition, CS 1800 Discrete Structures for Computer Science should be taken prior to CS 2510 or concurrently.

**Textbooks:** *How to Design Programs*, Felleisen, Flatt, Findler, and Krishnamurthi, MIT Press, 2000. *How to Design Classes*, Felleisen, Flatt, Findler, Krishnamurthi, Gray, and Proulx, in preparation.

**Topics Covered:** Objects, classes, class hierarchies. Data definitions and methods. Class diagrams. Interfaces, abstract classes, subclasses. Basics of static types vs. dynamic types. Primitive types vs. reference types. Polymorphism. Accumulators, iterators, loops. Function objects, comparators, visitors. Sorting: quick sort, merge sort, heap sort. Notions of equality such as structural equality and intensional equality. Java collections library. Overriding `equals` and `hashCode`. Testing: unit testing, stress testing, and property-based random testing. Binary trees, binary search, heaps, hash tables. Invariants of functional data structures. Invariants of stateful data structures and iterators. Graphs, graph traversal, generative recursion. Depth-first, breadth-first, and priority-queue based search. Design patterns. Mutation and stateful programming. Indexed data structures.

**Course Outcomes:** Upon completion of this course, a student should be able to: (1) Design, implement, and test class hierarchies with appropriate methods. (2) Read the documentation of class libraries and write programs that use them appropriately. (3) Understand the algorithmic behavior of individual methods.

**Measurement of Course Outcomes:** Weekly programming homeworks. Final project. Weekly short paper and pencil quizzes. Two to four major exams. Inclusion of student's work in an electronic portfolio.

## 2 CS 2510 Sample Syllabus

The course studies the class-based program design and the design of abstractions that support the design of reusable software and libraries. It covers the principles of object oriented program design, the basic rules of program evaluation, the use of abstractions in support of the design of reusable code, and examines the relationship between algorithms and data structures, as well as basic techniques for analyzing algorithm complexity.

The course is suitable for both CS majors and non-majors. It assumes that student has been introduced to the basic principles of program design and computation.

**Course Objectives:** The goal is to help the student understand the principles of class based program design using an object-oriented programming language. Java is used to illustrate how the program design principles are used in practical applications. Using a concrete language also presents an opportunity to discuss the strengths and weaknesses of languages and language paradigms.

**MODULE 1:** Review of Fundies 1 in the context of a class-based statically typed language (2 weeks)

- Review of the program design and abstractions in functional style; design recipe, abstractions, loops with accumulators
- Defining classes and class hierarchies to represent complex information: simple classes, classes with fields that are instances of another class, unions of classes, self-referential and mutually referential class hierarchies: lists, trees, graphs
- Designing methods for classes: static type system, dynamic dispatch, delegation (chain of responsibility)
- Design Recipe for data definitions, for method definitions

**MODULE 2:** Designing Abstractions (3 weeks)

- Review of the Design Recipe for Abstractions
- Abstracting over the data: abstract classes; subclasses
- Abstracting over the functional behavior: function objects, functional iterator
- Understanding equality of complex data

**MODULE 3:** Stateful Programming (3 weeks)

- Designing methods with effects; Designing tests for effects
- Understanding equality in the presence of mutation
- Direct access data structures: (e.g. Java ArrayList)
- Loops

**MODULE 4:** Abstracting over the Data Type; Designing and using libraries (6 weeks)

- Polymorphism: parametrized classes, methods
- Abstract Data Type — stack, queue, list, set
- Abstract Data Type — map, hashmap
- HashCode method and overriding equals method
- Algorithm complexity — estimates; Stress tests
- Java Collection Framework

**Notes:**

- A number of algorithms (filter, andmap, ormap, linear search, binary search, binary search trees, insertion sort, selection sort, quicksort, heap-based priority queue, heapsort, table lookup, use of hashmap, and some graph traversal algorithms (BFS, DFS, and possibly shortest path) are used to illustrate the program design, the need for appropriate data representation, and the need for understanding the complexity of algorithms.
- Stress tests are used to reinforce the need for understanding the algorithm complexity.
- Data structures (stacks, queues, lists, trees, arraylist, hashmap, lookup tables) are presented in the context of problems that motivate their use.
- The goal is to make sure the students begin to understand the design of class based libraries, know how to look up the needed information, and can begin using libraries in their programs.
- A comprehensive final project challenges the students to apply the design principles and the knowledge of the libraries to a creative larger program with up to 1000 LOC.