CSci 1933 - Introduction to Algorithms and Data Structures
(4.0 cr; Prereq 1133 or #; fall, spring, summer, every year)

Advanced object oriented programming to implement abstract data types (stacks, queues, linked lists, hash tables, binary trees) using the Java language. Inheritance. Searching/sorting algorithms. Basic algorithmic analysis. Use of software development tools. Weekly lab.

5 hours per week; Every academic year; Fall, Spring, Summer

Component 1: Lec (with final exam)
Component 2: Lab (with no final exam)

Prerequisites: 1133; Replaces CSci 1902

SLO
CSci 1933 is focused on the use of algorithms and data structures to solve problems. Specifically, the course labs, homework assignments, and exams all ask the students to solve various problems using appropriate software design methods and software tools. For example, students need to decide which problem solving strategies (such as divide and conquer) might be useful for a specific problem, construct a solution, design appropriate data types and algorithms, and verify the correctness of the solution.

This SLO will be assessed through labs, homework, exams, and in-class exercises. Each of these types of students' work is problem-based; most involve open-ended problems where students need first to identify, define, and/or clarify what the problem is before solving it, and explain the solution they propose.

Sample Syllabus
Class Format

4 credits, 3 hour long lectures + 2 lab hours per week. The lab is used for programming and more in-depth discussion of topics and examples given in class. Since one of the emphases of the class is on developing students' problem solving skills, the practice given in the lab will be highly
valuable for many students.

CSci 1933 focuses on learning the key abstract data types (list, stack, queue, tree) and object oriented concepts (class, object, method, inheritance) in the Java programming language. In addition the course
introduces students to the use of modern debugging and software development tools.

CSci 1933 is a required undergraduate course which computer science and computer engineering students should take in their freshman or sophomore year. It is one of the courses required for admission to the CSci major, and is a prerequisite for many of the higher level CSci classes. Transfer students who have completed 2 years at another school should have taken an equivalent class.

Why This Class is Important and its Role in the Curriculum

Much of computer science assumes familiarity with the common abstract data types (e.g. stacks) and basic concepts of an object oriented programming language. Moreover, many employers require programming skills in a modern object oriented programming language such as Java.

Prerequisites and Rationale

CSci 1133 builds fundamental problem solving and programming skills, and prepares students for learning the more complex Java language.

Classes Having 1933 as a Prerequisite and Rationale

These include CSci 2021 and 2041 which in turn are prerequisites for many other classes. 1933 teaches the notion of abstract data types as well as the fundamental abstract data types (stacks, queues, and trees), which are used extensively in the rest of Computer Science. To give a few quick examples, (i) the queue abstract data type is used for CPU and disk scheduling; (ii) stacks are used extensively in the compilers, architecture and programming languages courses to describe parsing algorithms for arithmetic expression, computer programs etc.;
(iii) abstract data types and object oriented programming are used as basic tools for software engineering.

Probable Text
Carrano, Frank. Data Structures and Abstractions with Java, 3rd Ed.

Outcomes

Upon successful completion of the course students should have the following skills and proficiencies:

1. For each of the abstract data types discussed in class, the student should be able to:
    - define the basic terminology and use it correctly,
    - give an explanation of why it is important,
    - provide and discuss specific CSci examples of its use,
    - be able to identify its important characteristics as well as any important variants or special cases,
    - perform the basic operations associated with it,
    - use it, when applicable, to represent and solve problems.
2. Given a problem, students should be able to:
    - identify which abstract data type and/or Java construct could be useful in representing or solving the problem, and why;
    - modify or specialize abstract data types or techniques to make them applicable to problems that are not amenable to straightforward use of the abstract data types;
    - develop a functioning object oriented program in Java, and document its functionality in terms of the given problem.
3. Student should be familiar with the process of developing and testing
programs, using modern proper software development tools such as debugger and version control software.

Weekly outline of material

WEEK 1: Overview. Introduction to Java and to IDE.
WEEK 2: Basic introduction to classes. Class and method specification. Object oriented design. Control structures, standard I/O, scalar data types.
WEEK 3: Additional material on classes. Method definition and invocation, parameter passing, scoping rules.
WEEK 4: Composite data types: arrays, strings.
WEEK 5; Use of software development tools (debugger, version control).
WEEK 6: Linked lists.
WEEK 7: Stacks.
WEEK 8: Queues. Priority queues.
WEEK 9: Introduction to trees: basic terminology and operations.
WEEK 10: Binary search trees. Heaps.
WEEK 11: Hash tables.
WEEK 12: Inheritance and polymorphism.
WEEK 13-14: Searching and sorting algorithms.
WEEK 15: Wrap up