Cape Cod Community College

Course Syllabus

Prepared by the Department of Mathematics
Date of Departmental Approval: April 10, 2017
Date approved by Curriculum and Programs: April 12, 2017 **Effective: Fall 2017**

1. **Course Number: CSC130**
   Course Title: Computer Programming II: JAVA

2. **Description:** Students use projects and teamwork to design, implement, and test large computer programs in Java, with emphasis on programming style, expression, and documentation. Object-oriented programming methodology, abstract data types, data structures, internal searching and sorting methods, exceptions, generics, multithreading, and simple recursion are covered. Students analyze the efficiency and compare times of recursive and non-recursive sorts and searches, as well as searches of graphs using stacks and queues.

3. **Student Learning Outcomes (instructional objectives, intellectual skills):** Upon successful completion of the course, the student are able to do the following:
   - Implement and customize a recursive binary search and a recursive sort.
   - implement, compare, and create appealing visual displays for sorting algorithms
   - Determine time and space requirements of common sorting and searching algorithms.
   - Trace complex recursion and write effective simple recursive solutions.
   - Determine whether a recursive or iterative solution is most appropriate for a problem.
   - Process single and multi-dimensional arrays in a relatively large program.
   - Implement and analyze the time and efficiency of a heuristic algorithm to improve the success rate, efficiency, and time of a brute force algorithm.
   - demonstrate two-way communication between two classes,
   - Effectively design and program with generics, class hierarchies, abstract classes, interfaces, and packages.
   - Implement and compare various flavors of java.
   - Extend classes, override superclass methods, use superclass constructors with arguments, and use information hiding.
   - Design and implement a program with abstract classes and/or interfaces which uses an array of subclass objects with dynamic method binding to demonstrate inheritance and polymorphism.
   - Write try/catch blocks with predefined exceptions and multiple exceptions; throw, try, and catch a programmer-defined exception.
   - Implement file input/output and multithreading.
   - choose appropriate abstract data types to solve a given problem
   - compare array and linked implementations of lists, stacks, queues and graphs
   - implement and compare depth and breadth first searches using stacks and queues

4. **Credits(s):** 4

5. **Satisfies General Education Requirement:** No

6. **Prerequisite(s):** CSC110 (Computer Programming I: JAVA) or CSC120 (Computer Programming I: C++) or CSC105 (Computer Programming I: Python)

7. **Semester(s) Offered:** Fall and Spring

8. Suggested General Guidelines for Evaluation: Comprehensive final examination, midterm, programs, quiz papers, and homework papers

9. General Topical Outline:

1) Java Concepts
   a. Classes and Methods
   b. Exception Handling
   c. Graphics
   d. Event Model

2) Non-recursive Sorts
   a. Implementation
   b. Graphical Display
   c. Timing Analysis

3) Searching
   a. Sequential Search
   b. Non-recursive binary search

4) Multi-Dimensional Arrays
   a. Brute force solutions to intractable problems
   b. Heuristics to improve success rate and time
   c. Graphical displays of solution

5) File Input/Output

6) Program Design and Methodology
   a. Data Abstraction
   b. Testing and debugging
   c. Verification
   d. Portability
   e. Efficiency and clarity

7) Basic Recursion
   a. Simple Recursion
   b. General Recursion
   c. Recursive Sorts
   d. Recursive Binary Search
   e. Timing Analysis of Recursive Sorts

8) Introduction to Algorithmic Analysis
   a. Big-O Notation
   b. Analysis of Sorts
   c. Analysis of Searches

9) Inheritance
   a. Abstract Classes
   b. Interfaces
   c. Packages

10) Abstract Data Structures (Lists, Stacks, Queues, and Graphs)
   a. Array Implementation
   b. Linked Implementation
   c. Compare Efficiency of Array and Linked Implementations

11) Graph Searches
   a. DFS using Stack
   b. BFS using Queue
   c. Implementation to Solve “Real World” Problem
   d. Comparison of Timing and Algorithm Analysis

12) Multithreading