CS 142 : Computer Science II with Java

Credits 5
Quarter Offered Spring
This course continues CS& 141, delving more deeply into computer science principles and professional software development principles and practices. We cover and use object-oriented and functional programming paradigms, basic top-down context-derived software processes and architectures, abstract data types, generics, data structures, recursion, complexity analysis of algorithms and O-notation, computer ethics, handling and querying data, unit tests, developing to standards, modeling physical processes, graphical user interfaces. We use a modern, intelligent professional development environment to implement concepts concretely. This course will help you become more competent and comfortable on the paths to both computer science and professional software development. This class may include students from multiple sections. (Elective)
Prerequisites
CS& 141 and currently enrolled in MATH& 141 OR 2.0 or higher in MATH& 141 with instructor permission
Course Outcomes

Estimate time and space complexities for a given algorithm using Big-Onotation.
Contrast standard complexity classes.
Implement common search algorithms, including linear and binary searches.
Compare various data structures for a given problem, such as array, list,set, map, stack, queue, hash table, tree, and graph.
Create and execute different traversal methods for trees and graphs.
Calculate probabilities of events and expectations of random variables for elementary problems.
Implement in code OOP constructs, including encapsulation, abstraction,inheritance, and polymorphism.
Contrast functional and object-oriented programming paradigms.
Use a professional-level integrated development environment (IDE) to create, execute, test, and debug secure programs.
Apply consistent documentation and program style standards.
Implement in code different types of testing, including security, unit testing, system testing, integration testing, and interface usability.
Compare professional codes of conduct from the ACM, IEEE Computer Society, and other organizations.
Understand and apply appropriate software architecture and software development strategies in creating solutions to a range of problems.
Apply directions, requirements, and specifications in solving problems.
Search and manipulate data using functional stream techniques.
Evaluate computer ethics issues.