CSC/CPE 102 Fundamentals of Computer Science II

1. CSC/CPE 102 Fundamentals of Computer Science II

2. **credit units** 4  
   **contact hours** 6

3. **Course Coordinator**: John Clements

4. **Textbook**: (and/or other required material) Big Java: Compatible with Java 5, 6 and 7, 4th edition, Cay Horstmann, Wiley. or similar text

5. a. **Course Description**: Basic design, implementation, testing, and documentation of object-oriented software. Introduction to classes, interfaces, inheritance, algorithms (sort, search, recursion), abstract data types, data structures (lists, stacks, queues), file I/O, and exceptions. Credit not available for students who have taken CSC/CPE 108. 3 lectures, 1 laboratory. Crosslisted as CPE/CSC 102.

   b. **Prerequisite**: CSC/CPE 101 with a grade of C- or better and either MATH 141 or MATH 221 with a grade of C- or better, or consent of instructor.

   c. **Required/Elective/Selective Elective for CPE, CSC, EE, SE**

<table>
<thead>
<tr>
<th></th>
<th>CSC</th>
<th>CPE</th>
<th>SE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Elective</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Selective Elective</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6. a. **Course Goals/Outcomes**

   The student will be able to:
   - Understand, explain, and apply the principles of object oriented design and programming: classes and objects, class types and primitive types, instantiation of objects and invocation of object and class methods, interaction between objects, composition, encapsulation, use of client and server classes and libraries of classes.
   - Understand, explain, and apply the principles of interfaces: abstraction and specification, independence of specification and implementation, contractual requirements in interface implementation, subtypes and type casting, polymorphism, interface hierarchies, multiple inheritance of interfaces.
   - Understand, explain, and apply the principles of inheritance: generalization, specialization via extension, difference between extension and composition, subtypes and type casting, polymorphism, inheritance hierarchies.
   - Understand, explain, and apply the principles of Abstract Data Type (ADT): specification and implementation, examples of common ADTs (list, other collections), ADT hierarchies, possibility of alternate ADT implementations, independence of application and ADT implementation.
• Understand, explain, implement, and apply basic data structures: static structures - single and multi-dimensional arrays; dynamic structures - array list (vector), linked list; other collections.
• Understand, explain, implement, and evaluate the operation of basic search and sort algorithms (O(N^2) exchange sorts), use of generic types in search and sort algorithms.
• Understand, explain, implement, and apply the use of exceptions: concept of method failure and exceptions, exception types and class hierarchies, exception objects, checked and unchecked exceptions, generating and handling error conditions via exceptions, control flow in exception handling.
• Understand, explain, implement, and apply the use of i/o streams: concept of an i/o stream, stream sources and destinations, stream data types, stream class types and hierarchies, stream objects, constructing and using streams, handling stream errors via exceptions, reading and writing text files.
• Understand, explain, and apply basic software engineering concepts such as programming by contract, pre- and postconditions, class invariants, unit and system testing, documentation; and graphical modeling notation for classes.
• Understand and explain the concept and principles of generic data types: know how to use generic types.

b. **How Student Outcomes addressed**
   (“B” = Basic level, “I” = Intermediate level, “A” = Advanced level)

<table>
<thead>
<tr>
<th></th>
<th>3a</th>
<th>3b</th>
<th>3c</th>
<th>3d</th>
<th>3e</th>
<th>3f</th>
<th>3g</th>
<th>3h</th>
<th>3i</th>
<th>3j</th>
<th>3k</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSC</td>
<td></td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B</td>
</tr>
<tr>
<td>SE/CPE</td>
<td>B</td>
<td></td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B</td>
</tr>
</tbody>
</table>

7. **Major Topics Covered: (number of lecture hours each)**

• Object oriented design fundamentals (5): classes, objects, state, behavior, class as a data type, primitive and reference types, object construction, composition, encapsulation, specification and implementation, interaction between objects, client (application) and server classes, class versus instance state and behavior.
• Object oriented programming fundamentals (4): fields, constructors, methods, constructor and method overloading, access control (private, public), static (class) versus instance fields and methods, instantiation of objects, invocation of instance methods and class, development and use of class libraries (packages), mutable versus immutable classes, memory model for objects.
• Interfaces and polymorphism (4): interface as a specification, interface as a type, interface implementation, polymorphism, type casting, strategy pattern and delegate objects, interface hierarchies, multiple inheritance of interfaces.
• Class inheritance (2): basic principles of inheritance and class extension, subclasses, overriding versus overloading methods, polymorphism, type casting, accessibility and scoping, class hierarchies.
• Abstract Data Types (2): specification versus implementation, list ADT, alternate implementations of list ADT, collection types and collection hierarchies, applying polymorphism to use alternate ADT implementations in an application.
• Data structures (4): single and multi-dimensional arrays, array list (vector), linked list, collection classes and hierarchies.
• Algorithms (2): linear and binary search; selection, insertion, and other $O(N^2)$ exchange sorts; counting number of operations; comparison of efficiency; using interfaces in sort and search algorithms; using generic types in sort and search algorithms.

• Exceptions and exception handling (2): exception types and class hierarchies, exception objects, checked and unchecked exceptions, generating and handling method failure via try and catch blocks, multiple catch blocks, nested try/catch blocks

• I/o streams (1): stream class types and hierarchies, constructing and using stream objects, handling stream errors via exceptions, using Java standard library stream classes, reading and writing text files.

• Software engineering (3): programming by contract, preconditions, postconditions, class invariants; unit testing and system testing; documentation; graphical modeling notation for classes.