CPE/CSC 203 – Project-Based Object-Oriented Programming and Design

1. CPE/CSC 203 – Project-Based Object-Oriented Programming and Design

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

3. **Course Coordinator**: Zoë Wood

4. **Textbook (or other required material)**: Core Java Volume 1, Horstmann

5. a. **Course Description**: Object-oriented programming and design with applications to project construction. Introduction to class design, interfaces, inheritance, generics, exceptions, streams, and testing. 3 lectures, 1 laboratory. Crosslisted as CPE/CSC 203.

   b. **Prerequisite**: CPE/CSC 202 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 Learning Objectives**

   The student will be able to:
   - Explain key object-oriented concepts including: classes, objects, methods, instantiation, invocation, interaction between objects, composition, encapsulation, and use of class libraries.
   - Use object-oriented concepts and design to implement moderately sophisticated “large” programs.
   - Describe the philosophy and mechanics of interfaces including abstraction and specification, independence of specification and implementation, contractual requirements in interface implementation, subtypes and type casting, polymorphism, interface hierarchies, and implementation of multiple interfaces.
   - Define interfaces in programs to support abstractions according to the principles of interface segregation and dependency inversion where design weaknesses are identified.
   - Create a unit test plan for a set of methods in a class.
   - Implement program pieces (classes and methods) that use generic types.
   - Discuss the differences among generics, subtyping, and overloading.
   - Describe the philosophy and mechanics of inheritance including generalization, specialization via extension, difference between extension and composition, subtypes and type casting, polymorphism, and inheritance hierarchies.
   - Use inheritance in the implementation of program components such as in the application of a refactoring process to an existing software implementation to improve some aspect of its design.
   - Explain the relationship between object-oriented inheritance (code-sharing and
overriding) and subtyping (the idea of a subtype being usable in a context that expects a supertype).
• Articulate design principles including separation of concerns, information hiding, coupling and cohesion, and encapsulation.
• Apply class design principles to the development of a design document for a moderately sophisticated “large” program.

b. Level at which Student Outcomes are addressed
(“B” = Basic level, “I” = Intermediate level, “A” = Advanced level)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSC</td>
<td>B</td>
<td>B</td>
<td></td>
<td></td>
<td>I</td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>SE/ CPE</td>
<td>B</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td>B</td>
<td></td>
</tr>
</tbody>
</table>

7. Major Topics Covered: (number of lecture hours per)
• Language support for object-oriented programming (6 hours)
• Class design (6 hours)
• Inheritance (3 hours)
• Interfaces (3 hours)
• Object-Oriented Design principles (3 hours)
• Generic programming (3 hours)
• Exceptions (2 hours)
• Introduction to Design Patterns (2 hours)