CSC 305 – Individual Software Design and Development

1. CSC 305 – Individual Software Design and Development

2. credit units  4    contact hours  6

3. Course Coordinator: Clint Staley

4. Textbook (or other required material):
   A C++ Primer, Prata; Core Java V1, Horstmann; Design Patterns, Gamma et al

5. a. Course Description:
   Practical software development skills needed for construction of mid-sized production-
   quality software modules, using the CSC upper division programming language. Topics
   include inheritance, exceptions, and memory and disk-based dynamic data structures.
   Students must complete an individual programming project of mid-level complexity. 3
   lectures, 1 laboratory.

   b. Prerequisite: CSC/CPE 357.

   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></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Elective</td>
<td>X</td>
<td>X</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:
   - Discuss the features of a standard OO language
   - Use a standard debugger
   - Use makefiles or a standard IDE
   - Develop and properly organize multi-sourcefile projects
   - Use modularity in building a project
   - Use module-level testing
   - Use a language-based assertion facility
   - Use and design generic classes (e.g. templates in C++, or genericity through inheritance in
     Java)
   - Use, at an intermediate level, a language-standard class library
   - Explain and use exceptions and exception handling
   - Explain the use of constructors and destructors, including copy constructors and
     assignment operators where applicable
   - Describe single inheritance, including abstract base classes
   - Describe class members and methods
• Use polymorphic methods and interface types
• Evaluate for use a selection of object-oriented design patterns
• Use Application Frameworks
• Individually write and test mid-sized object-oriented software modules of a professional quality, using a standard OO programming language, and using the concepts listed above.

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>I</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td>I</td>
<td>N/A</td>
</tr>
<tr>
<td>SE/ CPE</td>
<td>I</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>I</td>
</tr>
</tbody>
</table>

7. Major Topics Covered: (number of lecture hours per)
• Software Construction (15 hours)
  o Style rules
  o Testing practices, assertions
  o Automation of testing and regression testing.
  o Basic OO design process
  o Use of pointers or references (depending on language)
  o Design by Contract
• Class Libraries (5 hours)
  o Generic classes
  o Standard class library
• Object-Oriented Principles (6 hours)
  o Constructors/destructors
  o Inheritance
  o Virtual functions and polymorphism
  o Exceptions and exception handling
  o Design Patterns
  o Dynamic extension and plug-in architecture