Course Description:
It covers the concepts of object-oriented programming (OOP) languages and systems including an introduction to fundamental abstraction, modularity and encapsulation mechanisms in OOP from a software engineering and representational perspective. Basic OOP concepts covered include: polymorphism and operator overloading; message passing via generic functions; late versus early binding times; and inheritance mechanisms and their relationship to the type systems of programming languages. C++ and Java are the two languages used to convey the fundamental concepts and principles. Other topics include: OOP languages versus imperative programming languages; and contrasting pure OOP environments versus mixed paradigm languages.

Prerequisite: COSC 1557, COSC 1567

Recommended Textbook:

Course Objectives:
- This course will introduce the concepts of Object Oriented Programming (OOP) Languages and Systems. The course will start with an introduction of fundamental abstraction, modularity and encapsulation mechanisms in Object Oriented Programming Languages such as C++ and Java.
- Abstract data types will be examined from both a software engineering and representational perspective. Contrasts with imperative programming languages will be made. Advanced C++ and Java constructs such as abstract base classes, static members and other constructs will be presented. Comparisons with Java between C++ will be made.
- Other concepts to be covered include polymorphism, operator overloading, message passing viz. generic functions, late versus early binding times, as well as inheritance mechanisms and their relationship to programming language type systems.
- The emphasis of this course this semester will be the comparison of central features of several OOP. Special topics will be included as time permits. One such topic will be the discussion of Object libraries including the Java class libraries. Pure OOP environments such as Smalltalk will be contrasted with mixed paradigm languages such as C++ and Java.
- Object Oriented Design techniques as well as OOP idiom will be presented. We'll look at classic OOP patterns and frameworks.

Grading:
- Projects: 30%  
- Tests: 34%  
- Final Exam: 36%

Grading Table:
<table>
<thead>
<tr>
<th>No.</th>
<th>Date</th>
<th>Lecture</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01/07</td>
<td>Lect1</td>
<td>Think with objects</td>
</tr>
<tr>
<td>2</td>
<td>01/14</td>
<td>Lect2</td>
<td>Object-Oriented Design, Classes and Methods, Project 1</td>
</tr>
<tr>
<td>3</td>
<td>01/21</td>
<td>Lect3</td>
<td>Messages, Instances, and Inheritances</td>
</tr>
<tr>
<td>4</td>
<td>01/28</td>
<td>Lect4</td>
<td>Subclasses, subtype and Multiple Inheritance, Test 1</td>
</tr>
<tr>
<td>5</td>
<td>02/04</td>
<td>Lect5</td>
<td>Static, Dynamic Behavior and Substitution, Project 1 Due, Project 2</td>
</tr>
<tr>
<td>6</td>
<td>02/11</td>
<td>Lect6</td>
<td>Polymorphism and overloading,</td>
</tr>
<tr>
<td>02/18</td>
<td>No class</td>
<td>Study week</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>02/25</td>
<td>Lect7</td>
<td>Overriding, Test 2</td>
</tr>
<tr>
<td>8</td>
<td>03/04</td>
<td>Lect8</td>
<td>The Polymorphic Variable and Generics, Project 2 due, Project 3</td>
</tr>
<tr>
<td>9</td>
<td>03/11</td>
<td>Lect9</td>
<td>Container classes &amp; STL</td>
</tr>
<tr>
<td>10</td>
<td>03/18</td>
<td>Lect10</td>
<td>Frameworks and Object Interconnections</td>
</tr>
<tr>
<td>11</td>
<td>03/25</td>
<td>Lect11</td>
<td>Patterns, Class Objects and Distributed Objects, Project 3 Due</td>
</tr>
<tr>
<td>12</td>
<td>04/01</td>
<td>Seminar</td>
<td>E-CARGO and Review</td>
</tr>
<tr>
<td>04/??</td>
<td></td>
<td>Final Exam</td>
<td></td>
</tr>
</tbody>
</table>

Note: Tests are in-class, closed books and close notes. **Everything may be subject to change, please pay attention to the class declarations.**