CSC 330 Object-Oriented Software Design  
Fall 2008  
Course Syllabus

Instructor: Dr. Natacha Gueorguieva, Professor

Lectures:  
Mon 6:30 PM - 8:20 PM - 1N 111; 8:30 PM – 9:10 PM – 3N 107  
Wed 6:30 PM – 9:10 PM – 1N 004

Office Hours: Mon 2:30 pm – 3:15 pm; Wed 12:30 pm – 2:30 pm or by appointment  
1N 205

E-mail: natachag@mail.csi.cuny.edu

Web site for the course:  

Prerequisite: CSC 326  
Number of credits: 4

Pre-Requisite by Topic:  
1. Knowledge of C++ language.  
2. Stacks, queues, linked lists.  
3. Sorting algorithms.  
4. Recursion

Current Catalog Description  
Large-scale software design issues; object-oriented design paradigms; encapsulation;  
polymorphism; inheritance; reusability; specifics of an object-oriented language and associated  
development tools. Students will be required to implement a substantial and well-engineered  
project using an object-oriented language.

Required texts:  

Course Goals:  
The goal of this course is to enhance students' skills in problem analysis and program design and  
implementation through the presentation of complex applications. Additional software  
development experience will be provided by studying and implementing a programming language  
other than the one that the student has already studied as well. Continuous consideration will be  
given on the fundamental logical thinking and problem-solving skills in order to develop  
necessary expert skills independent of the particular language being used. The major attention in  
this course is given to the following parts:

Advanced Object-Oriented Programming: This part of the course will review and further explore  
object-oriented programming topics already introduced in CSC211, CSC326 and will add some  
additional features supported in C++ as inheritance, polymorphism, virtual functions and abstract  
classes, operator overloading, friends, exception handling.

Generic Programming: The second part of the course will look at strategies for creating reusable  
code and for exploiting work already done, including formalizing some common patterns of code  
and introducing C++ standard template library (STL). Topics include templates, generic  
programming and STL.
Computing with C# and .NET framework: Core topics include some C# basics, event-driven programming, user interfaces, inheritance, polymorphism, and databases.


**Course Learning Outcomes:**

Outcomes #1
a) understand and describe the paradigm of object-oriented programming;
b) explain the importance of OOP features including encapsulation, information hiding, inheritance, polymorphism, and abstract classes, exception handling and use them;
c) know the C++ syntax and semantics for inheritance, virtual functions, operating overloading, and friends, when these should be used;
d) ability to utilize C++ OOP features to generate reusable code using inheritance, polymorphism and virtual member functions.

Outcomes #2
a) understand the idea of generic programming and why it is important;
b) ability to utilize C++ OOP features to generate reusable code using templates and making appropriate use of advanced features such as inheritance, polymorphism and exception handling;
c) know the properties and typical operations of the ADTs iterator, container, list, vector, deque, stack, queue, priority queue, set, map and the specifics of the STL implementations of each;
d) use the key algorithms provided in the STL to quickly develop efficient solutions.

Outcomes #3
a) understand and implement event driven design and the development of GUI applications with .NET framework (C#);
b) know how to use C# to query a database;
c) ability to design a GUI for the user to query a database.

Outcome #4
a) understand the design project decomposition and reuse during the design and implementation.

Outcomes #5
a) an ability to function as part of a multi-disciplinary team;
b) an ability for effective verbal and written communication.

**Major Topics Covered in the Course**

Object-oriented and procedural programming. Classes and abstract data types. Information hiding. Encapsulation


Multiple Inheritance. Constructors in Multiply Inherited Objects. Ambiguity Resolution Inheriting from Shared Base Class. Problems with Multiple Inheritance.

Polymorphism – basic definitions. Problems with Single Inheritance.


Basic concepts of Exception Handling. An Exception Mechanism. Exception Handling Examples.


Connecting a database with C#. C# GUI for database queries.


Methods of Assessment

1. Work on individual homeworks and team programming / design projects.
2. Work on team power-point presentations.
3. Two exams during the semester and a final exam.
4. Participation in class through discussions and in-class lab exercises.

Assignments

There will be two kinds of assignments: individual assignments (homeworks) and group assignments (two projects and one presentation). All assignments will be posted on the class web page http://www.cs.csi.cuny.edu/~natacha/TeachFall_2008/CSC330/title.htm.

Assignments turned in late will be penalized 10% for each day or partial day of lateness for up to five days. After five days, no submission will be accepted unless other arrangements have been made in advance or unless unusual circumstances warrant an exception. All individual assignments must be done individually. While you may discuss the assignment in general terms with others, your solutions should be composed, designed, written and tested by you alone. Topics for PPT are posted on the course web site as well as criteria for PPT evaluation. All PPT will be presented in class in early December. Exceptional assignments will receive extra points.

Exams

There will be two exams during the semester and one final exam. The final exam will be comprehensive. The two term exam will take place during the regular class session and will be 70 minutes in length. The final exam will take place during the week specified by the university. The material for all exams will come from either a material covered in class, homework problems, and/or assignment reading.

Discussions and Lab Exercises

You are encouraged to actively participate in the learning process through in-class activities: discussions and in-class lab exercises. You will receive up to 8 additional bonus points which will be added to your final grade if you regularly and successfully take part in these activities.
Attendance Policy
Attendance will be taken each class. Attendance is not a certain percentage of the grade, but might bring a grade down to **WU** if you miss more than two weeks of classes. It is the student's own responsibility to acquire the material covered in classes she / he missed.

Oral and Written Communications
Each homework assignment should be accompanied by a short report (1-2 pages) and the two projects require 2-3 pages written report as well. Each team member will participate in presentations of up to 20 minutes each.

Academic honesty
You are encouraged to discuss assigned problems with other people but you must individually design and write your own solutions / code for all assignments. Furthermore, you should explicitly acknowledge any sources of ideas used that are not your own; this includes other people, books, web pages, etc. Submitting modified versions of other people's work as your own is considered cheating.

Grading
- Exams: ~20%
- Projects: ~30%
- Homeworks: ~20%
- Presentation: ~10%
- Final: ~20%

Course Outline
2. Introduction to Object Oriented Design.
4. Inheritance.
5. Software Reuse.
6. Polymorphism.
7. Exception Handling
8. Standard Template Library (STL)
9. C++ vs. C#.
10. Software design and implementation.

ABET Computer Science Program Outcomes

<table>
<thead>
<tr>
<th>a) An ability to apply knowledge of computing and mathematics appropriate to the discipline</th>
</tr>
</thead>
<tbody>
<tr>
<td>b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution</td>
</tr>
<tr>
<td>c) An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs</td>
</tr>
<tr>
<td>d) An ability to function effectively on teams to accomplish a common goal</td>
</tr>
<tr>
<td>e) An understanding of professional, ethical, legal, security and social issues and responsibilities</td>
</tr>
<tr>
<td>f) An ability to communicate effectively with a range of audiences</td>
</tr>
<tr>
<td>g) An ability to analyze the local and global impact of computing on individuals, organizations, and society</td>
</tr>
<tr>
<td>h) Recognition of the need for and an ability to engage in continuing professional development</td>
</tr>
<tr>
<td>i) An ability to use current techniques, skills, and tools necessary for computing practice.</td>
</tr>
</tbody>
</table>
(j) An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.

(k) An ability to apply design and development principles in the construction of software systems of varying complexity.

### Assessment of Program Outcomes

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Level</th>
<th>Course Outcomes</th>
<th>Method of Proficiency Assessment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a) Understand and describe the paradigm of object-oriented programming</td>
<td>S</td>
<td>b</td>
<td>discussions, PPT</td>
</tr>
<tr>
<td>1b) Explain the importance of OOP features including encapsulation, information hiding, inheritance, polymorphism, and abstract classes, exception handling and use them</td>
<td>H</td>
<td>b</td>
<td>PPT, homeworks (1, 2), projects, exams</td>
</tr>
<tr>
<td>1c) Know the C++ syntax and semantics for inheritance, virtual functions, operating overloading, and friends, when these should be used</td>
<td>H</td>
<td>b, i</td>
<td>homeworks (1, 2), lab exercises, project1, exam1 and final</td>
</tr>
<tr>
<td>1d) Ability to utilize C++ OOP features to generate reusable code using inheritance, polymorphism and virtual member functions.</td>
<td>H</td>
<td>b, i</td>
<td>homeworks, project1, exam1 and final</td>
</tr>
<tr>
<td>2a) Understand the idea of generic programming and why it is important</td>
<td>S</td>
<td>c</td>
<td>discussions</td>
</tr>
<tr>
<td>2b) Ability to utilize C++ OOP features to generate reusable code using templates and making appropriate use of advanced features such as inheritance, polymorphism and exception handling</td>
<td>H</td>
<td>b, c, i</td>
<td>homeworks (1, 2), lab exercises, project1, exam1 and final</td>
</tr>
<tr>
<td>2c) Know the properties and typical operations of the ADTs iterator, container, list, vector, deque, stack, queue, priority queue, set, map and the specifics of the STL implementations of each</td>
<td>S</td>
<td>b, c</td>
<td>discussions, homework3</td>
</tr>
<tr>
<td>2d) Use the key algorithms provided in the STL to quickly develop efficient solutions.</td>
<td>S</td>
<td>b, c</td>
<td>discussions, homework3</td>
</tr>
<tr>
<td>3a) Understand and implement event driven design and the development of GUI applications C#</td>
<td>H</td>
<td>b, c, i</td>
<td>exam2, homeworks (4, 5), project2, PPT, lab exercises, exam2, final</td>
</tr>
<tr>
<td>3b) Know how to use C# to query a database</td>
<td>S</td>
<td>b, c, i</td>
<td>discussions, homeworks (4, 5), project2</td>
</tr>
<tr>
<td>3c) Ability to design a GUI for the user to query a database</td>
<td>S</td>
<td>b, c, i</td>
<td>discussions, homeworks (4, 5), project2</td>
</tr>
<tr>
<td>4a) Understand the design project decomposition and reuse during the design and implementation</td>
<td>S</td>
<td>k</td>
<td>discussions</td>
</tr>
<tr>
<td>5a) An ability to function as part of a multi-disciplinary team</td>
<td>H</td>
<td>d</td>
<td>projects, PPT</td>
</tr>
<tr>
<td>5b) An ability for effective verbal and written communication</td>
<td>H</td>
<td>d, f</td>
<td>projects, PPT</td>
</tr>
</tbody>
</table>

**Level:**  
S = Somewhat supported  
H = Highly supported