

## COMP SCI 371 Advanced Object Design Using C++ -- Spring 2011

Course web site: [<http://www.uwgb.edu/shayw/courses/371>]

Meeting times: MW 8:00-9:20 and F 8:25-9:20 in MAC 122

**Professor:** Bill Shay

**Office:** MAC C308

**Phone:** 465-2316/5038

**email:** shayw@uwgb.edu

**Office Hours:** before and after class; or by appointment.

**Prerequisite:** COMP SCI 257 Software Design II with a C or better

**Required Text:** Big C++ (2<sup>nd</sup> edition), by Cay Horstmann & Timothy Budd

**Course:** There are three main goals of this course: learn C++, learn how to use C++ to implement various designs you learned in previous courses, and investigate additional design issues in preparation for the course in Software Engineering.

You'll also notice the text Big C++ is truth in advertising. This is a BIG book with nearly a 1,000 pages of content. Much of the book covers topics you are already familiar with from your coursework in Java. As such, we will move through those chapters very quickly and cover about two chapters per week for the first few weeks. I will rely heavily on the fact that you are familiar with software design issues covered in COMP SCI 256 and 257 and have a good working knowledge of the Java language. When we reach the more advanced material, we will cover it more slowly and methodically.

Approximately the first half of the semester will be devoted to doing object-oriented programming in C++. We will begin by learning how to work with Microsoft Visual Studio .NET. This is a software developer platform that allows you to design software in several languages including C++. However, this is NOT a course in Visual Studio .NET and after covering some bare essentials of the .NET platform we will move to the fundamental language constructs of C++, including types, functions, loops, decision making, functions, I/O, and object-oriented design issues. Once we have a provided a solid foundation we will investigate some additional design issues and the capabilities of C++. This will include topics such as pointers, dynamic memory allocation, memory leaks, inheritance, polymorphism, operator overloading, memory management, dynamic casting, polymorphic arrays, copy constructors, templates, lists, and iterators.

You will find many similarities between C++ and Java but you will quickly learn they are primarily syntax related. Once you progress beyond syntax you will see that they are very different languages.

The course will be a mix of lecture, demonstration, and hands-on activities. During the semester I will post demo programs on the course web site that will be the basis for the demo and hands-on work. Grading will be based on homework activities and exams as specified later in this document. Questions in class are **always** welcome and I encourage each student to examine material carefully and ask questions frequently.

Your assignments will consist of smaller programs assigned throughout the semester to get you working experience in C++ and a larger project to design and implement that will be assigned approximately mid-semester. You will have to manage your time carefully because there will still be some smaller assignments after the larger project is assigned.

**Approximate syllabus: minor changes are possible**

Week	Topics
1	Review of OO Design and UML: Classes, relationships, Cohesion, Coupling, Association, Aggregation, and Composition. Chapter 22 and 23.
2	Introduction to C++ and Microsoft Visual Studio .NET 2008. Building projects; managed (C++/CLI) and unmanaged applications (standard C++). Comparisons with Java. C++ Data types, constants, assignments, arithmetic, header files, I/O Streams ( <i>cin</i> and <i>cout</i> ), and simple <i>GUI</i> based programs. Chapters 1 and 2.
3	Implementing C++ classes and instantiating objects; interface and implementation files. Review of encapsulation, public and private areas, constructors, default constructors. C++ member and non-member functions, and destructors, method overloading, and exception handling. Chapters 2, 5, and 17.
4	C++ Flow control. <i>if</i> and <i>switch</i> statements; loops, input validation, loop invariants, proving loop correctness, nested loops. C++ Functions, class methods, and arrays. Returning values, <i>pass-by-value</i> , <i>pass-by-reference</i> , constant reference parameters, and <i>l-values</i> . Variable scope: local, global. Assertions. Testing and Debugging. Creating a test harness. Chapter 3 and 4.
5	Vectors vs. arrays, out of bounds conditions. Two dimensional arrays. Container classes. Chapter 6.
6	<b>EXAM:</b> Banking program design.
7	Input and output streams; File I/O. Stream class hierarchy. String streams. Separating application code from the interface code. Chapter 9.
8	Pointer variables, address operator, memory heap vs. memory stack, dangling pointers, memory leaks, relation to arrays and strings, dynamic arrays. Chapter 7.
9	Operator overloading, conversion operators, applications to code reuse. Chapter 14.
10	Memory management: buffer overflow, copy constructors, deep and shallow copies, assignment operator, writing proper destructors. Chapter 15.
11	<b>EXAM:</b> Class inheritance: Base and derived classes, <i>isa</i> relationship in class design, protected access, virtual functions and polymorphism. Chapter 8 and 19.
12	Polymorphic arrays and vectors, static and run-time binding, method overriding, dynamic casting, pointers to functions; multiple inheritance. Chapter 8 and 19.
13	Generic programming: C++ Templates. Application to Lists and Iterators. Chapter 12 and 16.
14	(if time) Multithreading.

**GRADING:** Exam 1: 20%; Exam 2: 20%; Assignments: 40%; Final exam: 20%.  
The final exam is scheduled for Monday May 9, 2011 from 8:00 AM-10:00 AM and must be taken at this time. **PLEASE NOTE:** This date is known many months in advance and any plans such as work schedules, vacations, trips, travel, etc. must be planned around this date.

## Program Grading

All programs are graded using the following criteria.

- Accuracy.  
The program must work correctly in all cases and be consistent with all stated requirements.
- Documentation and Readability.  
Documentation involves placing comments into the code that specify what a class and its various components (methods and attributes) are for. You must use clear and concise statements to articulate what is happening inside the program code. In general, comments must be included with each class definition, its methods, and its variables. However, DO NOT overdocument! For example, there is usually no need to document a single command line. Make use of indentation and visual aids to separate methods. Program demos will contain many examples.
- Interface.  
All output must be clearly marked and well organized. Never display data without some qualifying statement indicating what the data means. The user should never be asked to enter something without clear and explicit instructions as to what should be entered.
- Design. The classes and methods must be defined appropriately. Class variables must represent only attributes necessary for the object's state. Variables used for temporary storage in a method should not be declared as a class variable. Methods must reflect well designed and very specific behaviors. Implementing encapsulation and building flexibility, protection, and robustness into a class and its methods is essential to software engineering. This is where writing a program differs from designing software. You design software without the use of the computer. Find a quiet place and lay out exactly what problem you need to solve and how you are going to design your classes. Make an outline clearly indicating all the necessary components that go into the class and how they work and interact. When you have determined exactly how you are going to solve the problem, then you write the code. Any program designs must be produced neatly using a word processor and submitted as per the instructions on each assignment.

I'll use a form similar to that below for grading all assignments.

Criterion	Score: (0-10)	Weighted Score
Accuracy:		Score $\times$ 5 =
Design (Will not exceed the accuracy score):		Score $\times$ 3 =
Documentation and Readability: (Will not exceed the accuracy score)		Score $\times$ 1 =
Interface: (Will not exceed the accuracy score.)		Score $\times$ 1 =
		Total: (        )/100