

# Advanced Application Development

ITS 369

## COURSE SYLLABUS

*Term* Fall 2007

*Time and Place* Wednesday nights 6:30PM – 9:00PM in LT 132

*Instructor* Alan G. Labouseur *Office* LT101

*Office Hours* Tuesdays 1:30PM – 3:30 PM / Wednesdays 10:30 AM – 12:30 PM and 1:30 – 3:30 PM / and by appointment

*Voice* 845-575-3000 x2831 *Marist* 845-440-1102 *alternate*

*Fax* 845-575-3506 *Marist*

*E-Mail* alan.labouseur@marist.edu alan@3NFconsulting.com *alternate*

*Web* www.3NFconsulting.com – follow links for students and then class name.

*Text* Code Complete, 2<sup>nd</sup> Edition by Steve McConnell  
ISBN 0-7356-1967-0, Published by Microsoft Press, 2004

<i>Grading Criteria</i>	A	93-100 %	C+	77-79 %
	A–	90-92 %	C	73-76 %
	B+	87-89 %	C–	70-72 %
	B	83-86 %	F	0-69 %
	B–	80-82 %		

<i>Grading Opportunities</i>	Homework	25%	5 worth 50 points each = 250 points
	Chapter Presentation	5%	1 presentation (25 minutes) worth 50 points
	Test One	20%	1 worth 200 points
	Test Two	20%	1 worth 200 points
	Major Project	25%	1 worth 250 points
	Class Participation	5%	aggregate worth 50 points

- Course Objectives and Assessment Methods*
1. Develop the skills to design an advanced GUI application with a simple interface capable of solving complex problems.
    - Assessment methods include assignments, exams, and projects.
  2. Be able to construct programs using the C# programming language and Microsoft .Net environment
    - Assessment methods include exercises, assignments, and projects.
  3. Master the core concepts of Object-oriented programming.
    - Assessment methods include assignments, exams, and a project.
  4. Master the core concepts of event-driven programming.
    - Assessment methods include assignments, exams, and projects.
  5. Provide the students an opportunity to develop .Net-based systems over the course of the semester, where they have to live with their past mistakes and shortcuts, or fix them. Either will teach a valuable lesson.
  6. Troubleshooting: Developing the program is only half the battle. Debugging is a critical skill for a talented professional, and one that will be stressed in this course.
    - Assessment methods include exercises, assignments, projects, and pain.
  7. Continuing Education: Capable problem solvers never stop learning. Students will get practice in finding some answers for themselves.
    - Preparation and presentation of the final project, as well as participation in class discussions.

# Advanced Application Development

ITS 369

## SEMESTER SCHEDULE

C#	Week	Ch	Topic	Due
1	9/5	–	<i>Theory:</i> Introduction and “The Plan” / Overview of .Net vs. Java <i>Practice:</i> Intro to Visual Studio / Forms and Events / Our First App	<i>Presence</i>
2	9/12	22, 23	<i>Theory:</i> Testing and Debugging <i>Practice:</i> Controls and Menus / Event programming / Debugging	Hw 1
3	9/19	24, 25, 26	<i>Theory:</i> Refactoring and Code Tuning <i>Practice:</i> C# Operators, Flow Control, and Program Structure	<i>Progress</i>
4	9/26	–	<i>Class</i> Develop a report on unit-testing and the Nunit framework. <i>Exercise:</i> Detail best practices with examples in C# for Visual Studio.	Hw 2
5	10/3	–	<i>Theory:</i> Present Nunit Report to Alan / Integrate into Hw2 <i>Practice:</i> C# Data types / Collections / Creating and Reading XML	<i>Gumption</i>
6	10/10	5, 6	<i>Theory:</i> Design and Classes <i>Practice:</i> Classes, Constructors, objects / Properties & protection level	Hw 3
7	10/17	–	Test One in Class	<i>Expertise</i>
8	10/24	7, 8	<i>Theory:</i> Routines and Defensive Programming <i>Practice:</i> Encapsulation, Inheritance, and Polymorphism / Exceptions	<i>Diligence</i>
9	10/31	10, 11	<i>Theory:</i> Variables and their names <i>Practice:</i> Handling global data, Static classes, Enumerations	Hw 4
10	11/7	14, 15	<i>Theory:</i> Data Type Selection / Code Organization <i>Practice:</i> Rudimentary Graphics in C#	<i>Talent</i>
11	11/14	16, 19	<i>Theory:</i> (Obsessive Compulsive) Control Issues <i>Practice:</i> Working with databases – <i>Hand out Test Two</i>	Hw 5
12	11/21	–	No class meeting – Thanksgiving Break	<i>Appetite</i>
13	11/28	20, 21	<i>Theory:</i> Software Quality and Construction <i>Practice:</i> File system and Registry Operations	Test 2
14	12/5	31, 32	<i>Theory:</i> Style <i>Practice:</i> Multi-threaded processes and applications	<i>Headway</i>
15	12/12	–	Major Project presentations	Major Project
16	12/19	–	Comprehensive Final Exam	<i>Expertise</i>

# Advanced Application Development

ITS 369

## PROJECT AND ASSIGNMENTS

- Chapter Readings** You are expected to keep up with the chapters outlined in this syllabus.
- Tests** Tests cover material presented up to the class in which the test is administered. No makeup tests will be given. If you anticipate missing a test deadline, make arrangements with me to take and hand-in the exam prior to its due date.
- Homework** Homework assignments are essays or programs or problems that I assign. All assignments must be handed in at the beginning of class on the day they are due. Since all homework assignments are outlined in this syllabus or on the web site, arrange to submit homework on schedule, even when a class will be missed.
- Late Submissions** Late assignment submissions will be taken only at the end of the semester. If at that time the assignment appears to be correct you will be given half credit for it. If it does not clearly and obviously appear to be correct, you will receive no credit at all for it. Since the appearance of correctness relies on my ability to remember what the assignment was, if you miss a deadline and submit it late you should take extra care to make it as obviously correct as possible.
- Program and Project Evaluation Guidelines** All programs and applications must be free of syntax errors to receive any credit. Programs that compile or execute cleanly but contain logic errors will be graded based on the severity of the errors and your ability to demonstrate your approach to solving the problem. When evaluating your programming assignments, I will ask myself the following questions about your program:
- Is it correct; i.e., free from faults in its specification, design, and implementation?
  - Is it usable? About the interface...
    - Is it “clean” and well-organized? Would Mr. Monk be proud?
    - Does it obey the laws of least astonishment?
    - How accurate is it?
    - Is it easy to use?
  - Is it reliable and robust; i.e., able to perform its functions without breaking down, even given unexpected data or other circumstances?
  - About the readability and maintainability of the source code...
    - Are the comments plentiful, clear, meaningful, and helpful?
    - Are the identifier names accurate, clear, and meaningful?
    - How is its object-oriented architecture?
    - Does it compile or interpret cleanly?
  - About the Design...
    - Is the solution well-designed?
    - Is it re-usable?
    - Does it illustrate the points made or principles discussed in class?
    - Have any “lazy shortcuts” been taken?
    - Can the program be unit and system tested?
  - About the results, are they accurate; i.e., are the qualitative outputs free of error?
    - Does it perform as assigned?
    - Is the output well-formatted and meaningful?

# Advanced Application Development

ITS 369

Remember, neatness and style count. If you hand in a program that works, but that does not adhere to reasonable style standards, is inadequately commented, or is poorly designed, you will be penalized. Good habits are important and I want you to develop some.

## **Chapter Presentation**

Each student will volunteer for (or be assigned to) one chapter from the text book from which to prepare a 25 minute presentation to the class. This will count for 5% of the final grade. The presentation should include slides and source code examples illustrative of the points being made. It will be graded on quality, completeness, accuracy, and creativity. Have some fun with it.

## **Major Project**

You will write a substantial application or two of the course of this class. Details will follow. This is an individual project, and all work must be your own. You will make a presentation to the class to demonstrate your project and speak briefly about your experience developing it. This will be an informal presentation. You will also write-up documentation for your project to be handed in.

<i>Grading</i>	Analysis of problem	20%
	Completeness	20%
	Correctness as per the "Program and Project Evaluation Guidelines"	40%
	Quality of documentation	10%
	Presentation of project	10%

## **Class Participation**

Questions and class discussion are encouraged as we learn as much from each other as we do from the text and assignments. Besides, I occasionally get sick of hearing myself talk (really!), so your participation is very important and greatly appreciated. (And required.)

## **About homework**

Learning is an iterative process, which requires time and effort. It cannot be sped up. Homework plays a significant role in this respect. Spending the time to put your best efforts into the homework assignments over the course of the semester will guarantee that you get the most out of this class. I cannot make you do that, only you can.

## **ATTENDANCE**

The attendance policy for this class is simple: attend. I reserve the right to give a failing grade if you miss too many classes. Any planned or anticipated absences should be approved by me in advance. While "class participation" accounts for only 5% of your final grade, this presumes your full attendance. In other words, if you attend all class meetings but never partake in any class discussions, you will receive nothing for class participation. That 5% must be earned.

# Advanced Application Development

ITS 369

## ACADEMIC HONESTY

As a part of this class, I will uphold and enforce the general policies of this institution on academic honesty and plagiarism. All examinations, papers, projects, and homework assignments are subject to the usual standards of academic honesty as described in the Student Handbook and/or other related publications. Furthermore, I expect my students to behave in a manner appropriate to Computer Science and Information Technology professionals. Professional ethics demand that students embrace traditional “thou shall not cheat” behaviors, and also that they reject additional forms of dishonesty and abuse which are uniquely possible working with computers.

### Collaboration vs. Cheating

Each of you is expected to submit your own original work for assignments. On many occasions when working on assignments (but never exams!) it is useful to ask others – the instructors, your fellow students, strangers – for hints or to talk generally about aspects of the assignment. Collaboration in solving the problems is encouraged; you have a lot to learn from your fellow students, this is an important part of learning, and this is generally a positive and acceptable activity. However, in order to make grading the assignments a meaningful way to measure your effort and your understanding of the material, I must place some restrictions:

- You may work together in small groups on finding solutions, but each of you must then write up your favorite solution independently. You are responsible for understanding, presenting, and being able to explain on your own, all the work that you submit.
- You must indicate on all submitted work any assistance (human or otherwise) that you received. This means the names of your collaborators, the URLs of resources you used, etc. Any assistance that is not given proper citation will be considered a violation of this Academic Honesty policy.
- Any and all essay-type answers must be completely and entirely in your own words. You may use references (obviously) so long as they are cited. You may not, under any circumstances, copy and paste another’s material and hand it in as your own. Any violation of this will be considered a breach of this Academic Honesty policy.

There are always gray areas. Within the ground rules, the honesty of a student's behavior can usually be explored with the help of the following two guidelines:

- Plagiarism is suspected if an assignment calling for independent design and implementation results in two or more solutions that differ only by simple mechanical transformations.
- Cheating is suspected if an assignment calling for independent design and implementation results in a solution that can not be explained to the instructor, in terms of either general method or specific techniques. If you are suspected of cheating, you will be asked to explain the work. If you cannot you will be considered in violation of this Academic Honesty policy.

Any violation of this Academic Honesty policy will result in one or more of the following in addition to any other forms of recourse available to the instructor as specified by the Student Handbook:

- you will be ejected from the course with a failing grade
- a letter will be sent to your department chair, your Dean, and the president of the college
- and more (and worse)

The bottom line is that you are expected to conduct yourself as a person of integrity—you are expected to adhere to the highest standards of academic honesty. This means that plagiarism in any form is completely unacceptable. You are a (soon-to-be) computing professional; I encourage you to consult the ACM code of ethics. See [www.acm.org/constitution/code.html](http://www.acm.org/constitution/code.html).