CP 469 iPhone Application Programming
Winter 2018
Course outline

Lecturer: Chính T. Hoàng

Lecture: T R 4:00—5:20 in BA209
Prerequisite: CP213, CP217 (or CP264), CP317

It is the student's responsibility to read the course's policies described in this document.
Office: N 2076G, phone: X-2613, email: choang@wlu.ca
Office Hour: Monday 1:30 to 2:20, or by appointment
Text: *Beginning iPhone Development with Swift. By David Mark, Jack Nutting, Kim Topley, Frederik Olson and Jeff Lamarche. Publisher: Apress*

Course summary. How to write applications for the iPhone and iPod Touch, using the Cocoa Touch framework on Mac OSX. Introduction to the programming language Swift. interface development for mobile devices and dealing with different input modalities, web services.

What to expect: It is recommended that you own a Mac computer. The lab N2090 has about 30 Mac's, that number is less than the number of enrolments in the class. The assignments are inter-dependent, a later assignment will require code from a previous assignment. If you could not do the former, you would not be able to do the later.

Course evaluation:

• Assignments (6)	45%
• Midterm test	10%
• Final exam	20%
• Project	25%

You must achieve at least 50% of each the three components (Assignments, Project, Midterm test and Final exam) to pass the course. Assignments are individual. Projects may be done by groups of two.
Important dates:

- Project proposal due on March 01
- Project Implementation due on April 2, at noon.
- Project presentations on April 3, class time.

The assignments and assignments are individual works. The projects will be presented on XX (tentative date). Following discussions with the class, a more definitive date will be announced. The presentation is worth 10% of the project marks. There will be a competition for best app. If you win this competition, your final grade will be A+ regardless of your class performance.

For some assignments, you will be asked to make a presentation and defense of your code. The assignment marks will depend on this criterion.

Date	Chapter #	Topics	Events
Jan 4	2	Introduction to XCode and Interface Builder, the first app	
Jan 9	Appendix A	Introduction to Swift	
Jan 11	Class note	Advanced Swift	
Jan 16	3,4	Handling basic interactions, The MVC Paradigm	Assign 1 posted
Jan 18	5	Rotations and Adaptive Layout	
Jan 23	6	Multiview applications	Assign 1 due
Jan 25	12	Application Settings, NSUserDefaults	
Jan 30	13	Basic Data Persistence	Assign 2 posted
Feb 1	10	Data Persistence (cont.)	
Feb 6	7	Tabbar and pickers	Assign 2 due Assign 3 posted
Feb 8	8,9	Navigation and table view	
Feb 13	8,9	Table view	Assign 3 due Assign 4 posted
Feb 15	21	Camera and Photo Library	Mid term test (tentative date)
Feb 20	14	Reading Week	
Feb 22	15	Reading Week	
Feb 27	11	Split views (chap 11)	Assign 4 due Assign 5 posted
Mar 1	17, class note	SpriteKit (chap 17)	Project proposal due
Mar 6	16	Grand Central Dispatch, Background processing	
Mar 8		GCD (cont.)	Assign 5 due Assign 6 posted
Mar 13	Class note	Networking, URL Session Programming	
Mar 15		Networking (cont.)	
Mar 20	Class notes	XML Programming, String Programming	Assign 6 due
Mar 22	19	Core Location	
Mar 27	18	Gesture	
Mar 29	22	Localization	
Apr 3		Project presentations	Project presentation (Implementation due on Apr 2)

Assignment policy

- Assignments and projects must be submitted at mylearningspace.
- Assignments and projects are not accepted by email.
- Assignments will be posted at least one week before the due date.
- Programs should be written in the generally accepted style (For example, see the course note, or the book Code Complete: A Practical Handbook of Software Construction, Second Edition 2nd Edition, by Steve McConnell)
- Programs are marked on correctness and style, including internal documentation.

- Even though we do look at your code, the primary objective of the assignments is to implement the required functionality.
- Programs should be user-friendly and should not crash on bad input. Programs should warn user on bad input when this is feasible.
- Your assignment will be graded "fail" if it (i) does not compile, or (ii) has warnings, or (iii) crashes or hangs, or (iv) does not implement all requirements, or (v) does not have adequate internal documentation.
- The markers should be able to run your assignment without making any change to it.
- In case of dispute on the correctness of your assignment, it will be compile on a Mac in the lab.
- Late assignments are accepted up to 24 hours after the deadline and will be applied a 20% mark reduction. A later submission will override an earlier submission. Only the latest submissions are marked.

How to name the programs: suppose your Laurier email is *shoe3453@mylaurier.ca* and you are submitting assignment 2, then the program/Xcode project should be named *shoe3453_a2*, that is, the name of the XCode project is <*author_email*>_<*assignment_number*> where <*author_email*> is your Laurier email without the @-part, and <assignment_number> is obvious. The project should be submitted in a zip file. For example, the assignment 7 submission of Shoemaker is named *shoe3453_a7.zip*. Submit this zip file. This naming convention facilitates the tasks of marking for the course markers and instructor. It also helps you in organizing your course work. Failure to follow the requirements will result in 20% mark reduction.

Frequently encountered problems with assignment submission

Problem: I completed my assignment, but I did not upload my program to mylearningspace by the deadline because my Internet connection was down (or, because ftp did not work, etc.)

Solution: Do not wait to the last hour to submit the assignment. If you are trying to submit the assignment from home, and your Internet goes down, that is your own problem. Try to submit it 3 hours before the deadline.

Problem: I submitted the wrong file, or my zip cannot be uncompressed and I don't know why.

Solution: You can always download your submission and verify that it contains the right files. This does not take more than three minutes. You may resubmit as many times as you like, the newly submitted file will replace the existing file in mylearningspace. Note that you have to submit a zip file. If you resubmit your assignment after the deadline, it will be considered late.

Problem: The project runs on my computer, but not on the marker's computer.

Solution: The likely cause is you added a resource to the project folder using the option "Link to files". In this case only the link to the resource is added to the project. When the project is run on another computer, the resource cannot be found. So, when you add a resource, choose the option "Copy files".

Problem: My assignment is strikingly similar to that of another student because we "worked together" on it.

Solution: The assignments are individual. ***Do not work with another student on them. Do not give your work to another student.*** *If you are charged with plagiarism and it is your first offense, your submitted work will receive a mark of 0, ten final marks will be deducted from you final mark total, a letter will go into your permanent record, and copies will be sent to the Chair of Physics & Computer Science, and the Dean of Faculty of Science. If it is your second offense, then … you should not even think about it. Since this is a serious matter, and plagiarism occurs frequently, we will make clear of this course's policy (modeled after that of several North American universities, in particular, Stanford University's policy):*

It is usually appropriate to ask others—the TA, the instructor, or other students—for hints and debugging help or to talk generally about problem-solving strategies and program structure.
The important point, however, is embodied in the following rule:

Rule 1: You must indicate on your submission any assistance you received.

If you make use of such assistance without giving proper credit, you may be guilty of plagiarism.
In addition to providing proper citation—usually as part of the comments at the beginning of the program—it is also important to make sure that the assistance you receive consists of general advice that does not cross the boundary into having someone else write the actual code. ***If you used code from the Internet, say it in your submission and provide the link to the code****. It is fine to discuss ideas and strategies with others, but you should be careful to write your programs on your own. This provision is expressed in the following rule:*

Rule 2: You must not share actual program code with other students.
In particular, you should not ask anyone to give you a copy of their code or, conversely, give your code to another student who asks you for it. Similarly, you should not discuss your algorithmic strategies to such an extent that you and your collaborators end up turning in exactly the same code. Discuss ideas together, but do the coding on your own. The prohibition against looking at the actual code for a program has an important specific application in computer science courses. Developing a good programming assignment often takes years. When a new assignment is created, it invariably has problems that require a certain amount of polishing. To make sure that the assignments are as good as they can be, this department—like most others—reuses assignments over the years, incorporating a few changes each time to make them more effective. The following rule applies:

Rule 3: You must not look at solution sets or program code from other years.
Beyond being a clear violation of academic integrity, making use of old solution sets is a dangerous practice. Most assignments change in a variety of ways from year to year as we seek to make them better. Each year, however, some student turns in a solution to an assignment from some prior year, even though that assignment has since changed so that the old solution no longer makes sense. Submitting a program that solves last year's assignment perfectly while failing to solve the current one is particularly damaging evidence of an academic integrity violation. Whenever you seek help on an assignment, your goal should be improving your level of understanding and not simply getting your program to work. Suppose, for example, that someone responds to your request for help by showing you a couple of lines of code that do the job. Don't fall into the trap of thinking about that code as if it were a magical incantation—something you simply include in your program and don't have to understand. By doing so, you will be in no position to solve similar problems on exams. The need to understand the assistance you receive can be expressed in the following rule:

Rule 4: You must be prepared to explain any program code you submit at any time.
We may perform the following procedure to detect academic violations. We may use plagiarism detection tools. We archive all submissions, both from this semester and previous semesters, and cross-compare for unusual resemblance. We do not target specific students, all assignments are subject to the same scrutiny. Any similarity detected by the tools is then examined more closely by the course's staff and appropriate actions will be taken. The tools are very adept at identifying all variants of improper collaboration, from major to minor.

Rule 5: All submissions are subject to automated plagiarism detection.

In summary
Although you should certainly keep these rules in mind, it is important to recognize that the cases that we bring forward to the Dean's office are not those in which a student simply forgets to cite a source of legitimate aid. Most of the students we charge have committed fairly egregious violations. Students, for example, have rummaged through paper recycling bins or undeleted trash folders to come up with copies of other students' programs, which they then turn in as their own work. In many cases, students take deliberate measures—rewriting comments, changing variable names, and so forth—to disguise the fact that their work is copied from someone else. Despite these cosmetic changes, it is usually easy to determine that a copy has been made. Programming style is highly idiosyncratic, and the chance that two submissions would be that similar is vanishingly small. We have no desire to create a climate in which students feel as if they are under suspicion. The point is that we all benefit from working in an atmosphere of mutual trust. Students who deliberately take advantage of that trust, however, poison that atmosphere for everyone. As members of the Laurier community, we have a responsibility to protect academic integrity for the benefit of the community as a whole.

Additional Information