Revised 8/2023

NOVA COLLEGE-WIDE COURSE CONTENT SUMMARY
ITP 137 PROGRAMMING IOS DEVICES (4 CR.)

Course Description
Provides instruction in object-oriented Objective C design and programming concepts for Mac OS X, iPhone and iPad. Introduces the tools and APIs for the latest iOS SDK, and how they fit together to build full-featured iOS and Mac OS X applications. Lecture 4 hours per week.

General Purpose
To give students competence in designing, creating, and implementing applications in Objective C.

Course Prerequisites/Corequisites
None.

Course Objectives
Upon completion of this course, the student will be able to:

- Create iOS applications for the iPhone and iPad using C and Objective-C
- Apply the syntax of the Objective-C language
- Apply object-oriented programming concepts in the design and implementation of iOS applications
- Use XCode to develop programs for Mac iOS
- Describe the core frameworks in both Mac iOS
- Interpret Apple's iOS API documentation
- Create interface files and implementation files
- Write classes using protocols
- Distinguish between object ids and pointers

Major Topics to be Included
- How to use the XCode IDE
- Using Interface Builder to rapidly build applications
- Create iOS application user interfaces programmatically
- Create iOS application user interfaces using storyboards
- Source code revision control
- C syntax
- Objective C Syntax
- Interface and Implementation Files
- Adding functionality to classes with Protocols
- Object Oriented Programming
- Cocoa Touch API
- AppKit API
- Core Data API
- Model-View-Controller design pattern

Student Learning Outcomes

How to use the XCode IDE
- Create new classes and manage existing classes
- Add resources to a project
- Debug and deploy projects to devices

Using Interface Builder to rapidly build applications
- Use storyboard mode to quickly layout multi-screen applications
- Use the inspector to modify the properties of objects in layout
- Use Assistant mode to quickly hook up code to UI widgets

Create iOS Application User Interfaces Programmatically

- Declare properties based on user interface components and programmatically create the component in code
- Define the terms “View” and “Subview”

Create iOS Application User Interfaces using Storyboards

- Create a single-view application using a storyboard
- Create a multi-view application using storyboards
- Facilitate inter-view navigation via segues

Source code revision control

- Store projects in local and remote GIT repositories
- View history of project code.
- Merge with changes other students have made

C Syntax

- Write C functions that are called in response to user interface interactions
- Utilize iOS API C functions to implement iOS applications
- State the relationship between C and Objective-C

Objective C Syntax

- Write Objective C programs
- Distinguish between Interface and Implementation Files
- Use protocols
- Use the core NS framework objects

Interface and Implementation Files

- Declare a class’s public API by declaring public properties and methods in an @interface declaration
- Implement a class’s public API by synthesizing properties and defining methods in an @implementation declaration
- Declare private properties and methods in the class implementation file
- Implement class initializer methods
- Define the purpose of a designated initializer
- State the dependency between an Object-C class’s @interface declaration and @implementation declaration

Adding functionality to classes with Protocols

- State the purpose of a protocol
- List the authorized members of a @protocol declaration
- Describe the difference between @required and @optional protocol methods and properties
- Implement both @optional and @required methods and properties in a Object-C class

Object Oriented Programming

- Describe the MVC pattern as it applies to iOS programming
- Create Model View and Controller objects
- Describe the rationale behind the separation of concerns

Cocoa Touch API

- Identify the purpose of the various frameworks in Cocoa Touch: UIKit, Core Animation, Core Audio, Core Data
- Find reference material on each of the above frameworks

AppKit API

- Use this API to layout flexible user interfaces for multiple screen sizes
- Respond to user interface events.
- Customize user interface components programmatically

Core Data API

- Use the SQLLite database to store data
- Use Core Data to store data locally and to the cloud.

Model-View-Controller design pattern

- State the purpose of the model-view-controller design pattern
- State the purpose of the model
- State the purpose of the view
- State the purpose of the controller

Required Time Allocation per Topic

In order to standardize the core topics of ITP 137 so that a course taught at one campus is equivalent to the same course taught at another campus, the following student contact hours per topic are required. Each syllabus should be created to adhere as closely as possible to these allocations. Of course, the topics cannot be followed sequentially. Many topics are taught best as an integrated whole, often revisiting the topic several times, each time at a higher level. There are normally 60 student-contact-hours per semester for a four-credit course. (This includes 15 weeks of instruction and does not include the final exam week so 15* 4 = 60 hours. Sections of the course that are given in alternative formats from the standard 16 week section still meet for the same number of contact hours.) The final exam time is not included in the time-table. The category, Other Optional Content, leaves ample time for an instructor to tailor the course to special needs or resources.

Topic	Hours	Percentage
How to use the XCode IDE	**4**	**6.67%**
Using Interface Builder to rapidly build applications	**4**	**6.67%**
Create iOS application user interfaces programmatically	**3**	**5%**
Create iOS application user interfaces using storyboards	**3**	**5%**
Source code revision control	**1**	**1.67%**
C Syntax	**4**	**6.67%**
Objective C Syntax	**8**	**13.33%**
Interface and Implementation Files	**1**	**1.67%**
Adding functionality to classes with Protocols	**1**	**1.67%**
Object-Oriented Programming	**8**	**13.33%**
Cocoa Touch API	**4**	**6.67%**
AppKit API	**4**	**6.67%**
Core Data API	**4**	**6.67%**
Model-View-Controller design pattern	**1**	**1.67%**
Other Optional Content	**10**	**16.67%**
Total	**60**	**100%**