ATLS 4120/5120: Mobile Application Development
Week 1: First App

Xcode
Use Spotlight to search for Xcode
You should get a welcome screen with a choice to create a new project.
Or menu: File | New | Project

(chapter 2)
Creating Hello World
(helloworld)
iOS Application: Single View Application
Product name: helloworld
Organization name: Your name
This will be used in a copyright notice that Xcode automatically creates.
Organization identifier: TAM or something else
xCode will combine the product name and orgaization identifier to create a unique bundle identifier for your app.
Language: Swift
Device: iPhone
Leave Use Core Data Unchecked.
Next
Choose a folder for all your iOS projects.
Leave create local git repository unchecked.
Create.
xCode automatically creates the projects and files needed for helloworld.

Xcode Tour
Toolbar
Lets you perform tasks to build and run your project
Left drop down lets you choose how you want to build and run your project
In the middle is the activity view which shows what's going on at any given time
Editor

- Standard editor-single pane to edit a file
- Assistant editor-2 panes to see related files
- Version editor-compare versions

View lets you show or hide the navigator, debug, and utilities pane.
Organizer shows related information including the documentation.

Navigator
Let's you view all the files that make up your project.
Organized in Xcode but it does not relate to where the files are really organized on your hard drive.
Project view:

- Project files – the files that make up your project
- Supporting files – all other files including images
- Frameworks – the iOS libraries your project uses
- Products – the application binary

Symbols
Search

Issues – errors or warnings
Debug
Breakpoint
Log

Jump Bar
Lets you quickly jump to different elements in your project.

- Popup menu
- Forward/back files
- Segmented popup arranged as the hierarchy of your project

Utility Pane
Provides context-sensitive information.

- File inspector
- Quick help

Interface Builder Tour
Click on Main.storyboard
A blank screen will show up in IB's editor pane.
Right now your storyboard has one view. Click it.

Hierarchy view
Click the little button in the lower-left corner of the editing area to bring up the hierarchy view.
This shows the content of the storyboard, split up into scenes and content.
We have just one scene, called View Controller Scene that has a View Controller, which in turn has a View (along with some other things you'll learn about later).
View is the object with the area that a user can see and interact with. It was created when we selected the Single View Application template.
All gui objects will go in the view.
This will be a useful way to see everything in your storyboards.
First Responder is the object that the user is currently interacting with.
Exit will be useful when we have multiple views.
You can also see these by clicking on the top bar of the view.

Utility View
Information for the currently selected objects. This is where you set various properties.
File inspector
Quick Help
Identity inspector
Attributes inspector
Size inspector
Connections inspector

Library
File templates
Code Snippets
Objects(square in a circle): Contains all the UI objects you can use in your user interface by simply dragging and dropping them onto your view.
Media library

Search field at the bottom of the library.

Building Hello World
Click on Main.storyboard
You'll notice the view is a square, which is an abstract view of a screen.
Go into the Object library (square with a circle)
Scroll or search for label
Drag a Label from the library into the View window.
Place it towards the top center.
Double click the label and type Hello World! in it.
Save the file.
Build & Run to compile it and run it in the Simulator.
You'll notice it's not centered like you thought it would be. Remember the square is an abstract view of the screen, so the positioning won't be what you see on a device. We'll be using auto layout and size classes to deal with this later on.
For now let's keep it simple by selecting the File Inspector icon and deselect Use Size Classes.
Xcode will prompt you, chose Keep size class data for: iPhone and click Disable Size Classes.
Now the view will look more like an iPhone. Click and drag your label to where you want it.
Run it again in the simulator and it should look the same as in IB.
You can play around with your text by going into the attributes inspector and changing its color, font, shadow, background color, etc.

Snapshot: label

<u>Interactivity</u>
Now let's make it interactive so our app does something.
Now drag a button into the view.
Place it below the label.
Double click the button and add some text like "Say hello"(this changes the title in the button attributes)
Double click the label and remove the text so it's blank (or change it in attributes)
Run to compile it and run it in the Simulator.
Looks fine but doesn't do anything yet.

Snapshot: label and button

Xcode created a file called ViewController.swift
This view controller will control our view.
// are comments
`import UIKit` imports the UIKit framework which is the foundation of all iOS UI (import is like include in C)
`class ViewController: UIViewController` defines our class ViewController.
`UIViewController` is the superclass (which is in UIKit which is why we imported that)
All the code for this class will be in the curly braces.

Now we have to connect the interface and the code.
Go back into your storyboard.
We also need to see the swift file so open up the assistant window. (middle editor button)
Click on the button and then hold down the control key.
Then click and drag from the button over to the swift file.

Notice the blue fishing line being drawn between these two. This is how we'll connect them.
Move your cursor between the curly braces for the class.
When you see a grey box appear release the mouse button.
This window lets you set up the connection between the button and your code.
Connection: Action
Name: buttonPressed
Type: UIButton
Event: Touch Up Inside is the standard event to use for buttons.
Arguments: Sender
Now hit Connect.
You should now see in the swift file

```
  @IBAction func buttonPressed(sender: UIButton) { }
```

This is a method called buttonPressed that will be called when the user taps the button.

Before we implement buttonPressed we have to connect our label.
Instead of hunting for that empty label you can click on label in the object hierarchy.
(use the button on the bottom left to expand the hierarchy if you can't see it)
(Editor | Canvas | Show Bounds Rectangles to see ui element borders)
Control-click from the label to the swift file.
Connection: Outlet
Name: messageText
Type:UILabel
Leave storage as weak.
Connect

Notice this created in the swift file

```
@IBOutlet weak var messageText: UILabel!
```

If you named either of these differently, don't change them, it will break it, just go with what you have.

Now we're ready to implement the method for the button. Go into the swift file

```
    @IBAction func buttonPressed(sender: UIButton) {
        messageText.text="Hello World!"
    }
```

This assigns the string "Hello World" as the text of label when that action occurs (the user clicks the button).
Notice the autocomplete. Autocomplete is your friend. Get used to hitting return or tab to accept a choice. It makes the typing easier, guards against typos, and helps signal when something is wrong.

Save and Run.

Snapshot: button working

Finishing Touches
In the simulator hit the home button to go back to the home screen. (Hardware | Home)
We need an icon!
In the Project Navigator click on the Images.xcassets folder.
Click on AppIcon

You'll notice 3 spots, for settings, spotlight, and app.
Let's just add the app icon for the home screen, you can do the others later.
60 pt is points, not pixels. Depending on the resolution there is a different ratio between points and pixels. On early iPhones 1 pt=1 px but On most of the later devices with a Retina display, a single point is actually a 2 × 2 pixel square. On the iPhone 6 Plus 1 point is a 3 × 3–pixel square.
We will need a120x120 pixel image for the 2x icon, and a 180x180 pixel image for the 3x icon.
They MUST be png files.
Png is the best format to use as Xcode optimizes them to make them the most efficient to use in iOS apps
iOS will automatically add a mask to make it looks like the others.
Drag the 120x120 png file into the app 2x space.
Drag the 180x180 png file into the app 3x space.
Save, and run.
Now go to the home screen and see your app icon!

Snapshot: app icon