ATLS 4120/5120: Mobile Application Development
Week 3: UIKit Framework

Beatles
(beatles)
New Project, Single View application, iPhone
We will use 3 images of close to the same size, around 400x400.
Select the Images.xcassets item and click the plus button in the lower-left corner of the editing area. This brings up a small menu of choices, from which you should select New Image Set. This creates a new spot for adding your image file.
Copy the main picture into the 3x spot(Beatles_Abbey_Road.png) and rename the image to beatles_abbey_road. (you should also create one for 2 x and 1x).
Repeat the process 2 more times for beatles1.png and beatles2.png and rename those to beatles1 and beatles2 respectively.
We're giving them each a unique name, so we can refer to it elsewhere in the project.

Go into the Main storyboard.
Add an image view. It will try to scale to fit the whole view. Don't worry about the size yet.
In the attributes inspector choose Beatles_Abbey_Road for the image.
Click off the image view and then selected it again.
Size it by Editor | Size to Fit Content. This changes the size of the image view to be the size of the image.
Once it's resized, move it to be centered at the top of the view.
In the attributes inspector under View in Mode choose Aspect Fit. This will help with your other images if they don't all have the same aspect ratio it will make sure it keeps their aspect ratio.

Add a label right under it and change the text to say The Beatles.

Add a segmented control with two segments named Early 60s and Late 60s.
Look at the attributes inspector and note the segments are numbered 0 and 1.

We're not disabling size classes on this app so for today use the left side of the storyboard.
Go into Assistant Editor and in the jump bar click on Automatic and chose Preview. This lets you preview your layout without running your app in the simulator. You can reposition objects in your storyboard and see them move in the Preview. You can preview for as many devices as you want.

Connections
Connect the image and label as outlets since we'll be changing the image and the text in the label.
Open the assistant editor and make the connections to ViewController.swift.
Name the image beatlesImage, the rest is fine.
Name the label titleLabel, the rest is fine.
This should add in the ViewController.swift file

```
    @IBOutlet weak var beatlesImage: UIImageView!
    @IBOutlet weak var titleLabel: UILabel!
```

The segmented control needs to be an outlet so we can access which segment is chosen.
Connect the control as an outlet called imageControl.
It also needs to be connected as an action so we can change the image when the user choses a segment.
Then make an Action connection named changeInfo, type UISegmentedControl, event is value changed.

This created in the swift file

```
@IBOutlet weak var imageControl: UISegmentedControl!
@IBAction func changeInfo(sender: UISegmentedControl) {
}
```

Now let's implement the method so different images show depending on when segment is chosen.

```
@IBAction func changeInfo(sender: UISegmentedControl) {
    if imageControl.selectedSegmentIndex==0 {
        titleLabel.text="Young Beatles"
        beatlesImage.image=UIImage(named: "beatles1")
    }
    else if imageControl.selectedSegmentIndex==1 {
        titleLabel.text="Not so young Beatles"
        beatlesImage.image=UIImage(named: "beatles2")
    }
}
```

I really don't want either segment to be choosen initially. Go into the storyboard, select the segmented index and in the attributes inspector chose Segment 0 and uncheck Selected. If both segments are not selected then neither will be initially selected.

Snapshot: segmented control

Switch

Go back into the storyboard and add a switch. Make its State Off.
Put a label next to it that says Capitalization.
Connect the switch as an Outlet called capitalSwitch so we can easily access its value.
Now connect it as an Action and name it updateFont, type UISwitch, event Value Changed.
Go into the swift file to implement the method.

```
@IBAction func updateFont(sender: UISwitch) {
    if capitalSwitch.on {
        titleLabel.text=titleLabel.text?.uppercaseString
    } else {
        titleLabel.text=titleLabel.text?.lowercaseString
    }
}
```

`capitalizedString` does mixed case

Snapshot: switch

Slider

Add a slider to control the font size so a min of around 6 and a max of 22 is good. Initial can be 16 or something mid-sized. Change these in the attributes inspector.
Add a label next to it to show the font size. You can use that initial value in the label field.
Connect the label as an outlet called fontSizeLabel.
Connect the slider as an action called changeFontSize, type UISlider, event Value Changed.

Now implement the method.

```
    @IBAction func changeFontSize(sender: UISlider) {
        let fontSize=sender.value
        fontSizeNumberLabel.text=String(format: "%.0f", fontSize)
        let fontSizeCGFloat=CGFloat(fontSize)
        titleLabel.font=UIFont.systemFontOfSize(fontSizeCGFloat)    }
```

Create a constant called fontSize with the UISlider value.
The text label text property expects a string so we use the String class format initializer to convert the slider value which is a float to a string.
To change the font of the label we have to create a UIFont object with the new font size.
The UIFont class method systemFontOfSize returns a UIFont object.
We cast the slider value which is a float to a CGFloat which is the required parameter type for systemFontOfSize. This returns a UIFont object that we assign to the font property of titlelabel.

Snapshot: slider

Now you'll notice that almost everything works but if you change the image then we lose the caps setting. That's because we're changing the label's text and not taking case into account.
We don't want to rewrite the code to check the caps switch so instead we're going to use helper methods that we can easily call whenever we need them.

Before we do this, using snapshots is a good habit to get into so if you ever need to go back you can. File | Create Snapshot and give it a descriptive name so you can tell them apart. Use these a lot!

So let's move the image code into a helper method `updateImage()` and the caps code into a method `updateCaps()`

```
    func updateImage() {
        if imageControl.selectedSegmentIndex==0 {
            titleLabel.text="Young Beatles"
            beatlesImage.image=UIImage(named: "beatles1.png")
        }
        else if imageControl.selectedSegmentIndex==1 {
            titleLabel.text="Not so young Beatles"
            beatlesImage.image=UIImage(named: "beatles2.png")
        }
    }

    func updateCaps() {
        if capitalSwitch.on {
            titleLabel.text=titleLabel.text?.uppercaseString
        } else {
            titleLabel.text=titleLabel.text?.lowercaseString
        }
    }
```

Then the methods will call those.

```
    @IBAction func changeInfo(sender: UISegmentedControl) {
        updateImage()
        updateCaps()
    }
```

```
    @IBAction func updateFont(sender: UISwitch) {
        updateCaps()
    }
```

Snapshot: code done

App icons
Drag the icon files into the space for app icons. (Beatles_Abbey_Road_icon180.png, Beatles_Abbey_Road_icon120png)

Snapshot: app icons, final