ATLS 4120/5120: Mobile Application Development
Week 2: OOP

More UI elements
(daVinci)
File | New | Project
iOS Application: Single View Application
Product name: daVinci
Organization identifier: Your name or initials
Device: iPhone
Leave Core Data Unchecked.
Next
Choose a folder for all your iOS projects.
Leave create local git repository unchecked.
Create.

We're going to be using 3 images in this app, all of them with width 440, height 598 (daVinci.png, daVinci_MonaLisa.png, daVinci_Virtruvian.png)
Find/edit/create your images (use Photoshop if needed) and save them as png files.

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(daVinci.png) and rename it from Image to DaVinci (you should also create one for 1x).
Repeat the process 2 more times for the other two images renaming them to MonaLisa and Vitruvian. Make sure they each have a unique name, so we can refer to it elsewhere in the project.

Click on Main.storyboard
In the File Inspector uncheck Use Size Classes.
Keep size class data for iPhone.
Go into the Object library (square with a circle)
Scroll or search for label
Drag a Label from the library into the View.
Change the label text to say "Leonardo da Vinci"

Now drag a button into the view.
Change the text to "Paintings" and go into the Attributes inspector under View make Tag=1.
Now add another button and give it a title "Drawings" and tag=2. (or command D to duplicate the first one)

In the object library select an image view and drag in onto the view. It might resize it, don't worry.
Select the image view and go into the size inspector to make the width 220 height 299.
Once it's resized, move it to be centered and under the buttons.
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.

At the top of the attributes inspector tab and in the image field choose your main image file (daVinci)
Add one more label underneath the image and remove the text so it's empty.

Snapshot: UI

Now let's make the buttons do something. Depending on which button the user presses, a different image and text will show up.
Now we have to connect the interface and the code.
We also need to see ViewController.swift so open up the assistant window. (middle editor button)
Click on the first 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: chooseArt
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 chooseArt(sender: UIButton) {
}
```

This is the method header for chooseArt that will be called when the user taps the button.

Now let's connect the second button.
We could create a new method as well, but we can use the one we already have.
Make sure the swift file is open in the assistant editor.
Cntrl click on the second button and drag the blue line to the swift file.
This time go to the method you already have, chooseArt, and you'll see a grey popup saying Connect Action. When you see that, release the mouse button to connect the button to chooseArt.

Since we'll want the image to change based on which button is pressed, let's hook up the image as an Outlet and call it artImage.
Control-click from the image to the swift file to make the connection.
Connection: Outlet
Name: artImage
Type:UIImageView
Leave storage as weak.
Connect.
In your swift file it should have added

```
@IBOutlet weak var artImage: UIImageView!
```

Now let's connect our label.

Instead of hunting for that empty label you can click on label in the object hierarchy.
(use the grey button on the bottom left to expand the hierarchy if you can't see it)
(or Editor | Canvas | Show bounds rectangles will show all UI element bounds)
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!
```

Click on the View Controller icon and go into the connections inspector and you can see your connections and that both buttons are connected to chooseArt.
I'll show you how to fix any misspelled connections, for now, just leave them.

Snapshot: connections

Outlets and Actions

Outlets are special properties that point to an object in an Interface Builder file.
IBOutlet is a keyword that tells IB that this instance variable is an outlet that will connect to an object in a storyboard
IB will let you make connections *only* to **IBOutlet** instance variables.

Actions are methods from your controller class.
IBAction is a keyword that tells IB that this method is an action that can be triggered by a control.
IB will let you make connections *only* to **IBAction** methods.

Now we're ready to implement the method for the button. Let's first get the image working. Go into ViewController.swift

```
@IBAction func chooseArt(sender: UIButton) {
    if sender.tag==1{
        artImage.image=UIImage(named: "MonaLisa.png")
    }
    else if sender.tag==2{
        artImage.image=UIImage(named: "Vitruvian.png")
    }
}
```

`sender` is the name of the parameter of type UIButton.
It has a tag property which has the value we gave Tag in IB for each button.
The image property is of type UIImage. It is initially set when we choose our main image in IB.
Now we initialize a new UIImage object with the name of the file we want to use.
Look at the quick help or option-click to see the definitions of the image property and the UIImage class.
We assign the new UIImage object to the image property of artImage to change the image.

Snapshot: image

Now let's add a message into the label to tell the user the name of the piece of art.

```
    @IBAction func chooseArt(sender: UIButton) {
        if sender.tag==1{
            artImage.image=UIImage(named: "MonaLisa.png")
            messageText.text="Mona Lisa"
        }
        else if sender.tag==2{
            artImage.image=UIImage(named: "Vitruvian.png")
            messageText.text="Vitruvian man"
        }
    }
```

Snapshot: working

Finishing Touches

Don't forget we need app icons.
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.
We will need a120x120 pixel image for the 2x icon, and a 180x180 pixel image for the 3x icon.
They MUST be png files.
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: final