UIKitFromCode Tutorial

Getting Started: Start a new project in XCode.

-- use the “Single View Application” Template

-- call the project “UIKitFromCode”

-- enable Automatic Reference Counting

-- Do not use Storyboards

-- Set the Device for iPhone only

Part 1

Objective: Add a button to a view using code.

1. Add the following code to the *viewDidLoad* selector of **ViewController.m**.

```
- (void)viewDidLoad
{
      [super viewDidLoad];
      UIButton *myButton = [UIButton
buttonWithType:UIButtonTypeRoundedRect];
      myButton.frame = CGRectMake(50., 50., 100., 50.);
      [myButton setTitle:@"Click me" forState:UIControlStateNormal];
      [self.view addSubview:myButton];
}
```

What this does: This creates a UIButton object and sets it's display type.
The second line sets the button's frame, because without a frame it wouldn't know where to be drawn on the screen.
The third line of code sets the title of the button for the particular “control state”. This allows a different title for each state of the button. (ie: normal, highlight, selected, etc...)
Finally, we add the button as a child view of the main view by using the “addSubview” selector.

NOTE: The scope of the *myButton* variable is only within the *viewDidLoad* selector. Once that function is finished, the button can no longer be referred to by that variable name.

2. **RUN THE APP.** The button should appear on the screen. However, our button doesn't do anything when it's clicked.

3. Add the following code with in the *viewDidLoad* selector just after the line where the title is set.

```
[myButton setTitle:@"Click me" forState:UIControlStateNormal];
[myButton addTarget:self action:@selector(buttonClick:)
forControlEvents:UIControlEventTouchUpInside];
[self.view addSubview:myButton];
```

What this does: This will call a selector called *buttonClick:* that is within the scope of *self* (ie: the same class) when the **TouchUpInside** event is fired off.

4. Add the following code anywhere within the **ViewController.m** file after the **@implementation** directive. Please note that the order of selectors does not matter to the compiler, but it can make it more human readable.

```
-(void)buttonClick:(id)sender{
      NSLog(@"The button was clicked:%@", sender);
}
```

What this does: This creates the selector that the **TouchUpInside** event will call. Note that it passes along "sender" which is a reference to the object that sent the event notification; in this case, the button that was clicked.

5. **RUN THE APP.** This will display the message "The button was clicked" in the console whenever the button is clicked. You'll need to display the console within XCode (⇧⌘C)

Part 2

Objective: Add a UILabel to the screen.

6. Add the following code to after the **@interface** directive in the **ViewController.h** file. Note that there is an @interface directive in the .m file as well. Putting this in the .h file makes it public, whereas putting it in the .h file makes it private.

```
@property (nonatomic, strong) UILabel *myLabel;
```

What this does: This creates a property that will allow the object to be referred to any time the object it lives in is still in memory. (ie: **self.myLabel)**

7. Add the following code to the bottom of the *viewDidLoad:*selector in the **ViewController.m** file.

```
self.myLabel = [[UILabel alloc] initWithFrame:CGRectMake(50., 150., 150.,
50.)];
self.myLabel.text = @"Hello";
[self.view addSubview:self.myLabel];
```

What this does: This creates a UILabel object with a frame, sets the text property, and adds it as a subView to the view of the ViewController.

8. Now let's talk to the button! Add the following code after the NSLog line in the *buttonClick:* selector.

```
NSLog(@"The button was clicked: %@", sender);
self.myLabel.text = @"Goodbye";
```

What this does: This changes the text being displayed by the label.

9. **RUN THE APP.** Now when you click on the button, the text in the label will change.

BONUS

10. Add some logic to the code that will change the text on the label every time the button is clicked.