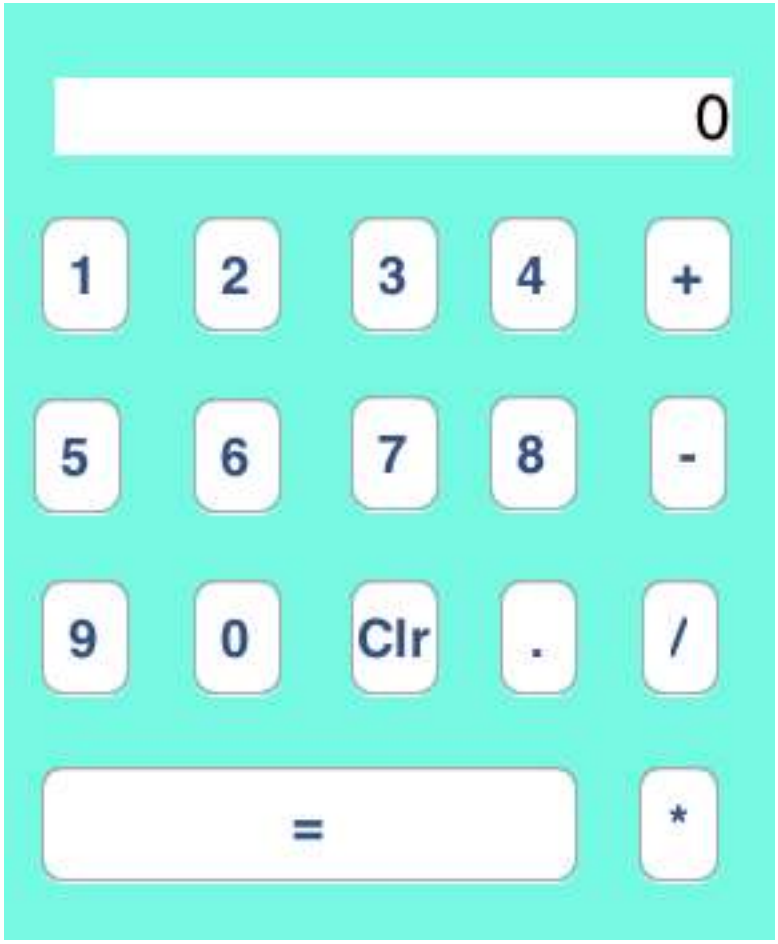
In this tutorial we are going to build a simple calculator using buttons that are all linked together using the same method. We will also add our own method to the source code to create some additional functionality.

Step 1: Open Xcode and select **Create a new Xcode Project** from the Welcome to Xcode menu.

Step 2: Select **Single View Application** using the template. Name the project **Calculator** or whatever you want. Select **Automatic Reference Counting** but do **NOT** select to use the storyboard feature.

Step 3: Xcode automatically creates the directory structure and adds essential frameworks to it. You can explore the directory structure to check out the content of the directory. Click on **ViewController.xib** to open it in the center view pane.

Step 4: Drag 1 Label and 17 Round Rect Buttons to the View so that you make something that looks like this:



Step 5: Control Drag the **Label** to the ViewController.h file to add it as a property. Call it **theTotal** and you should see this:

```
@property (weak, nonatomic) IBOutlet UILabel *theTotal;
```

Next: Add the @synthesize in **ViewController.m** in between the @implementation and the @End. So now you have:

```
@synthesize theTotal;
```

Step 6: Wire **ONLY 1** of the **number** button's **Touch Down event** to ViewController.h. Call the IBAction **numButtons**. You should see this added:

```
- (IBAction)numButtons:(UIButton *)sender;
```

Connect **ALL** of the other buttons to this same IBAction. Click on the little black dot next to this line of code and drag the line to all of the other number buttons. Now all of the numbers are connected to this one IBAction.

Step 7: Same as in step 6, Wire the **Touch Down event** of the +, -, /, * buttons (4 of them total) to another IBAction and call it **opButtons**.

```
- (IBAction)opButtons:(UIButton *)sender;
```

Step 8: Wire the **Touch Down event** for the **Decimal** button to its own IBAction and call it **decimalButton**.

```
- (IBAction)decimalButton:(UIButton *)sender;
```

Step 9: Wire the **Touch Down event** for the **Clear** button to its own IBAction and call it **clrButton**.

```
- (IBAction)clrButton:(UIButton *)sender;
```

Step 10: Wire the **Touch Down event** for the **Equals** button to its own IBAction and call it **eqButton**.

```
- (IBAction)eqButton:(UIButton *)sender;
```

Step 11: Add a function prototype to ViewController.h for a method we are going to define ourselves. The method should be called, **doEquals** and its not connected to any GUI controls.

```
-(void) doEquals;
```

Step 12: While you still have the **ViewController.h** file open, we need to add the data members that we will be using to keep track of the calculator data. Add 3 database members (variables) to ViewController.h. You will also need to add the opening and closing brackets, { and }. The finished code is:

```
@interface ViewController : UIViewController
{
    int operator;
    NSString *previous;
    NSString *current;
}
```

Step 13: Switch over to the **ViewController.m** file. We are now going to add the functionality to make the methods work for our calculator. First we need to initialize our variables to starting values. We can do this in our ViewDidLoad method. Initialize the 3 data members to "0" in the viewDidLoad method of **ViewController.m** by including the following code:

```
- (void)viewDidLoad
{
    operator=0;
    current =@"0";
    previous =@"0";
    [super viewDidLoad];
}
```

Step 14: Fill in the method for the number button control to capture the buttons pressed and update the total label with the value entered. The finished method looks like:

```
- (IBAction)numButtons:(UIButton *)sender {

    NSString *str= (NSString* )[sender currentTitle];
    if([current isEqualToString:@"0"])
    {
        current= str;
    }else
    {
        current=[current stringByAppendingString:str];
    }
```

```
    [theTotal setText:current];

}
```

Step 15: Go ahead and try it out to make sure the number buttons are working properly.

Step 16: Fill in the code for the operator buttons in **opButtons**. Add the following code to the **opButtons** method in **ViewController.m**.

```
- (IBAction)opButtons:(UIButton *)sender {

    NSString *tmpstr =[current substringFromIndex:([current
length]-1)];

    if([tmpstr isEqualToString:@"."])
    {
        current = (NSString *)[current substringToIndex:([current
length] - 1)];

        [theTotal setText:current];
    }

    NSString *str=(NSString *)[sender currentTitle];
    if(operator >=1 && operator<=4)
    {
        [self doEquals];
    }
    if(operator!=5)
    {
        previous = [current copy];
        current =@"0";
    }
    if ([str isEqualToString:@"+"])
    {
        operator=1;

    } else if([str isEqualToString:@"-"])
    {
```

```
        operator=2;

    } else if([str isEqualToString:@"*"])
    {
        operator=3;

    } else if([str isEqualToString:@"/"])
    {
        operator=4;
    }
}
```

Step 17: Fill in the code for the decimal button functionality. Add the following code to the **decimalButton** method in **ViewController.m**.

```
- (IBAction)decimalButton:(UIButton *)sender {
    NSRange range = [current
rangeOfString:@"."options:(NSCaseInsensitiveSearch)];
    if (range.location == NSNotFound)
    {
    current =[current stringByAppendingString:@"."];
    }
    [theTotal setText:current];
}
```

Step 18: Add the **doEquals** implementation to **ViewController.m**.

```
-(void) doEquals
{
    if (operator >= 1 && operator <= 4) {
        NSDecimalNumber* num1 = 0;
        num1 = [NSDecimalNumber decimalNumberWithString:previous];
        NSDecimalNumber* num2 = 0;
        num2 = [NSDecimalNumber decimalNumberWithString:current];
        if (operator == 1)
        {
            num1 = [num1 decimalNumberByAdding:num2];
            current = [NSString stringWithString:[num1 stringValue]];
        } else if (operator == 2){
```

```
            num1 = [num1 decimalNumberBySubtracting:num2];
            current = [NSString stringWithString:[num1 stringValue]];
        } else if (operator == 3){
            num1 = [num1 decimalNumberByMultiplyingBy:num2];
            current = [NSString stringWithString:[num1 stringValue]];
        } else if (operator == 4){
            if (![current isEqualToString:@"0"]) {
                num1 = [num1 decimalNumberByDividingBy:num2];
                current = [NSString stringWithString:[num1 stringValue]];
            }
        }
        [theTotal setText:current];
        previous = [current copy];
        current = @"0";
        operator = 5;
    }
}
```

Step 19: Tell the equals button to run the **doEquals** method when the Touch Down event occurs. Add the following code to the **eqButton** method in **ViewController.m**.

```
- (IBAction)eqButton:(UIButton *)sender {
        [self doEquals];
}
```

Step 20: Run the project and try out the +, - and = buttons.

Step 21: Tell the Clear button to reset the data to 0 when the Touch Down event occurs. Add the following code to the **clearButton** method in **ViewController.m**.

```
- (IBAction)clrButton:(UIButton *)sender {
    current=@"0";
    previous=@"0";
    operator=0;
    [theTotal setText:current];
}
```

Step 22: You are done. You have a working calculator!