### **Lab Exercise: Simple Calculator using Objective-C**

**Objective:**

here's the complete code for creating UI elements programmatically along with the addition logic for both Swift and Objective-C:

**Objective-C Implementation Without Storyboard**

#import "ViewController.h"

@interface ViewController ()

@property (nonatomic, strong) UITextField \*number1TextField;

@property (nonatomic, strong) UITextField \*number2TextField;

@property (nonatomic, strong) UILabel \*resultLabel;

@property (nonatomic, strong) UIButton \*calculateButton;

@end

@implementation ViewController

- (void)loadView {

self.view = [[UIView alloc] init];

self.view.backgroundColor = [UIColor whiteColor];

self.number1TextField = [[UITextField alloc] initWithFrame:CGRectZero];

self.number1TextField.placeholder = @"Enter number";

self.number1TextField.borderStyle = UITextBorderStyleRoundedRect;

self.number1TextField.translatesAutoresizingMaskIntoConstraints = NO;

[self.view addSubview:self.number1TextField];

self.number2TextField = [[UITextField alloc] initWithFrame:CGRectZero];

self.number2TextField.placeholder = @"Enter number";

self.number2TextField.borderStyle = UITextBorderStyleRoundedRect;

self.number2TextField.translatesAutoresizingMaskIntoConstraints = NO;

[self.view addSubview:self.number2TextField];

self.resultLabel = [[UILabel alloc] initWithFrame:CGRectZero];

self.resultLabel.text = @"Result: ";

self.resultLabel.translatesAutoresizingMaskIntoConstraints = NO;

[self.view addSubview:self.resultLabel];

self.calculateButton = [UIButton buttonWithType:UIButtonTypeSystem];

[self.calculateButton setTitle:@"Calculate" forState:UIControlStateNormal];

[self.calculateButton addTarget:self action:@selector(calculate) forControlEvents:UIControlEventTouchUpInside];

self.calculateButton.translatesAutoresizingMaskIntoConstraints = NO;

[self.view addSubview:self.calculateButton];

[NSLayoutConstraint activateConstraints:@[

[self.number1TextField.topAnchor constraintEqualToAnchor:self.view.topAnchor constant:100],

[self.number1TextField.leadingAnchor constraintEqualToAnchor:self.view.leadingAnchor constant:20],

[self.number1TextField.trailingAnchor constraintEqualToAnchor:self.view.trailingAnchor constant:-20],

[self.number2TextField.topAnchor constraintEqualToAnchor:self.number1TextField.bottomAnchor constant:20],

[self.number2TextField.leadingAnchor constraintEqualToAnchor:self.view.leadingAnchor constant:20],

[self.number2TextField.trailingAnchor constraintEqualToAnchor:self.view.trailingAnchor constant:-20],

[self.resultLabel.topAnchor constraintEqualToAnchor:self.number2TextField.bottomAnchor constant:20],

[self.resultLabel.leadingAnchor constraintEqualToAnchor:self.view.leadingAnchor constant:20],

[self.resultLabel.trailingAnchor constraintEqualToAnchor:self.view.trailingAnchor constant:-20],

[self.calculateButton.topAnchor constraintEqualToAnchor:self.resultLabel.bottomAnchor constant:20],

[self.calculateButton.centerXAnchor constraintEqualToAnchor:self.view.centerXAnchor]

]];

}

- (void)viewDidLoad {

[super viewDidLoad];

// Do any additional setup after loading the view.

}

- (void)calculate {

NSString \*number1Text = self.number1TextField.text;

NSString \*number2Text = self.number2TextField.text;

double number1 = [number1Text doubleValue];

double number2 = [number2Text doubleValue];

if (isnan(number1) || isnan(number2)) {

self.resultLabel.text = @"Invalid input";

} else {

double result = number1 + number2;

self.resultLabel.text = [NSString stringWithFormat:@"Result: %.2f", result];

}

}

@end

**Exercise**

* Run the code for both Swift and Objective-C implementations to ensure that the UI elements are created properly.
* Test the addition logic by entering numbers into the text fields and tapping the "Calculate" button to see if the result is displayed correctly in the label.
* Experiment with adding error handling to handle cases such as non-numeric input or empty text fields.