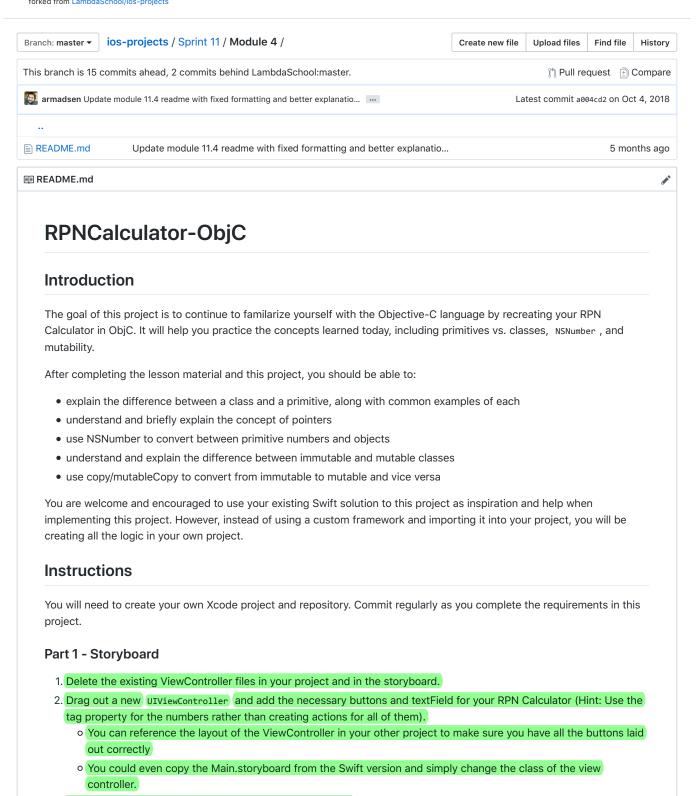
IvanCaldwell / ios-projects



Part 2 - Model/Model Controller

1. Create a new file called xxxstack (XXX = your class prefix).

3. Make sure your view controller is the initial view controller.

1 of 2 2/28/19, 3:06 PM

- .h file: Declare the methods to push, pop and peek the stack. Declare an initwith initalizer that takes an NSArray .
- .m file: Create a private NSMutableArray property called values that will be used every time your methods are called. Implement the methods you declared in your .h file to alter your values array. Inside of your initializer assign a mutableCopy of your array property to your internal values array.
- 2. Create another file called XXXCalculator.
- .h file: Create an enum for the operations that will be calculated using typedef NS_ENUM(rawType, nameOfEnum) {}; . Inside the curly braces you will just need to separate each item with a comma. Declare three methods (void)pushNumber:(double)value; , (void)applyOperator:(YourEnumType)operator; and (void)clear; . Also declare a topValue property that will be a computed property in your .m file.
- .m file: Every Calculator should have one Stack and only that calculator should know about it's Stack. Create a property XXXStack *stack in this file that we will use to implement the methods. Implement the methods in this file by accessing the methods on your Stack model.
- 3. Create a new file called XXXDigitAccumulator. This will be used to accumulate digits as they're entered, then convert the list of digits into a number.
- .h file Your XXXDigitAccumulator class should have three methods and one property: -addDigitWithNumericValue: -addDecimalPoint -clear value a readonly double property
- .m file Implement the digit accumulator. Use the Swift version for inspiration. Note that the Swift version uses an enum with associated values for digits, which you can't do directly in Objective-C. You'll have to come up with another solution! **Note**: For now, for simplicity, you can ignore the error handling that the Swift version does. Ignore repeated decimal digits instead of throwing an error, and assume that values will be in the range 0-9.

Part 3 - View Controller

Implement the view controller. Use the Swift version for inspiration. The implementation here will be nearly identical except that it will be in Objective-C.

Go Further

If you finish early or want to push yourself, here are a few additional features you can implement:

- Split the logic out into a framework as we did in Swift
- Implement error handling. What should happen if the user enters more than one decimal digit? Note that <code>@throws</code> is not available in Objective-C. Objective-C uses a different mechanism for error handling, which we'll learn about in a future lesson. Without knowing that, how would you solve this?

2 of 2 2/28/19, 3:06 PM