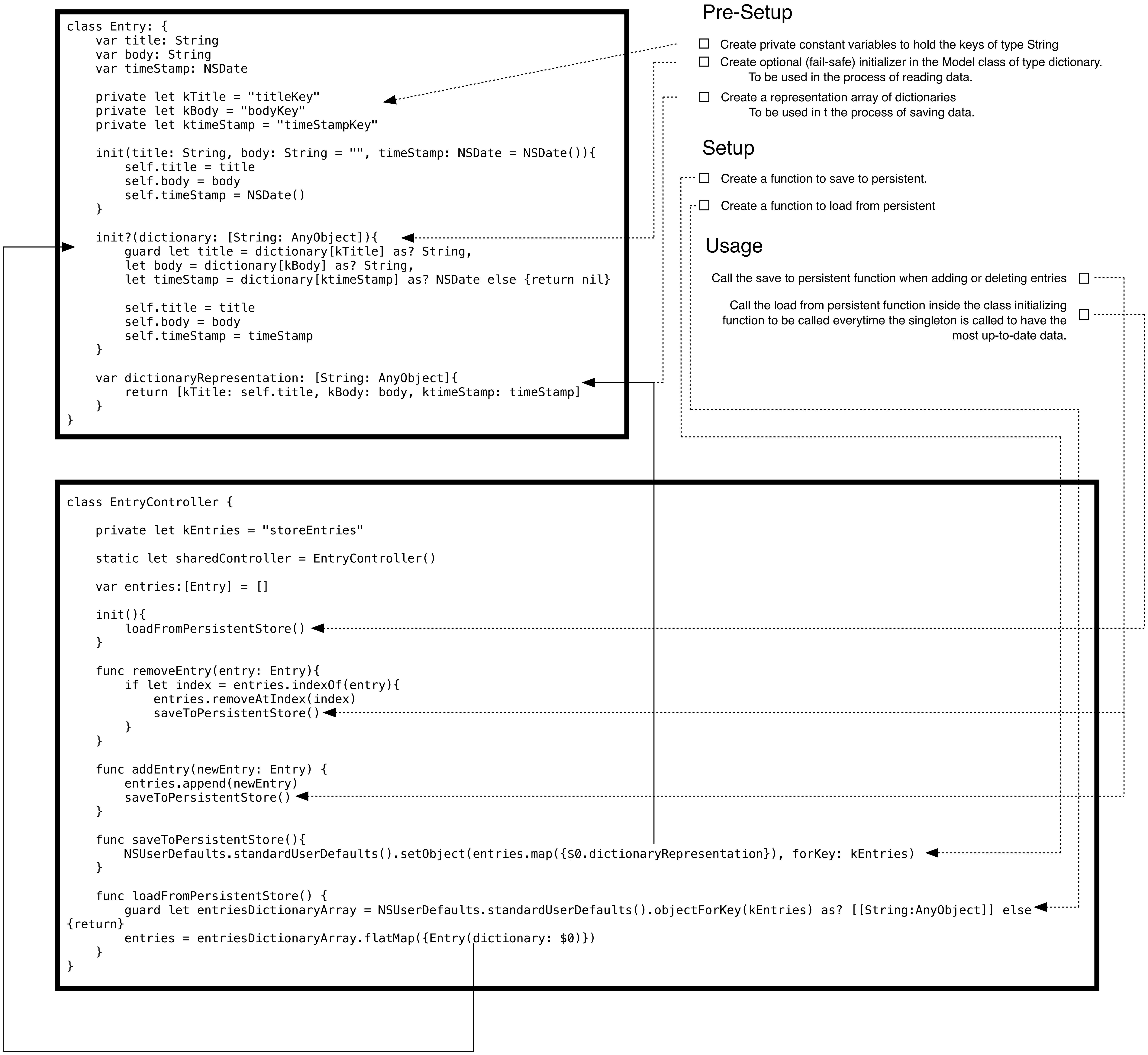
NSDefaultUser : Data Persistent



```
class Entry: {
    var title: String
    var body: String
    var timeStamp: NSDate

    private let kTitle = "titleKey"
    private let kBody = "bodyKey"
    private let ktimeStamp = "timeStampKey"

    init(title: String, body: String = "", timeStamp: NSDate = NSDate()){
        self.title = title
        self.body = body
        self.timeStamp = NSDate()
    }

    init?(dictionary: [String: AnyObject]){
        guard let title = dictionary[kTitle] as? String,
        let body = dictionary[kBody] as? String,
        let timeStamp = dictionary[ktimeStamp] as? NSDate else {return nil}

        self.title = title
        self.body = body
        self.timeStamp = timeStamp
    }

    var dictionaryRepresentation: [String: AnyObject]{
        return [kTitle: self.title, kBody: body, ktimeStamp: timeStamp]
    }
}
```

Pre-Setup

□ Create private constant variables to hold the keys of type String

□ Create optional (fail-safe) initializer in the Model class of type dictionary.
To be used in the process of reading data.

□ Create a representation array of dictionaries
To be used in t the process of saving data.

Setup

□ Create a function to save to persistent.

□ Create a function to load from persistent

Usage

Call the save to persistent function when adding or deleting entries □

Call the load from persistent function inside the class initializing function to be called everytime the singleton is called to have the most up-to-date data. □

```
class EntryController {

    private let kEntries = "storeEntries"

    static let sharedController = EntryController()

    var entries:[Entry] = []

    init(){
        loadFromPersistentStore()
    }

    func removeEntry(entry: Entry){
        if let index = entries.indexOf(entry){
            entries.removeAtIndex(index)
            saveToPersistentStore()
        }
    }

    func addEntry(newEntry: Entry) {
        entries.append(newEntry)
        saveToPersistentStore()
    }

    func saveToPersistentStore(){
        NSUserDefaults.standardUserDefaults().setObject(entries.map({$0.dictionaryRepresentation}), forKey: kEntries)
    }

    func loadFromPersistentStore() {
        guard let entriesDictionaryArray = NSUserDefaults.standardUserDefaults().objectForKey(kEntries) as? [[String:AnyObject]] else
{return}
        entries = entriesDictionaryArray.flatMap({Entry(dictionary: $0)})
    }
}
```