Code Archeology

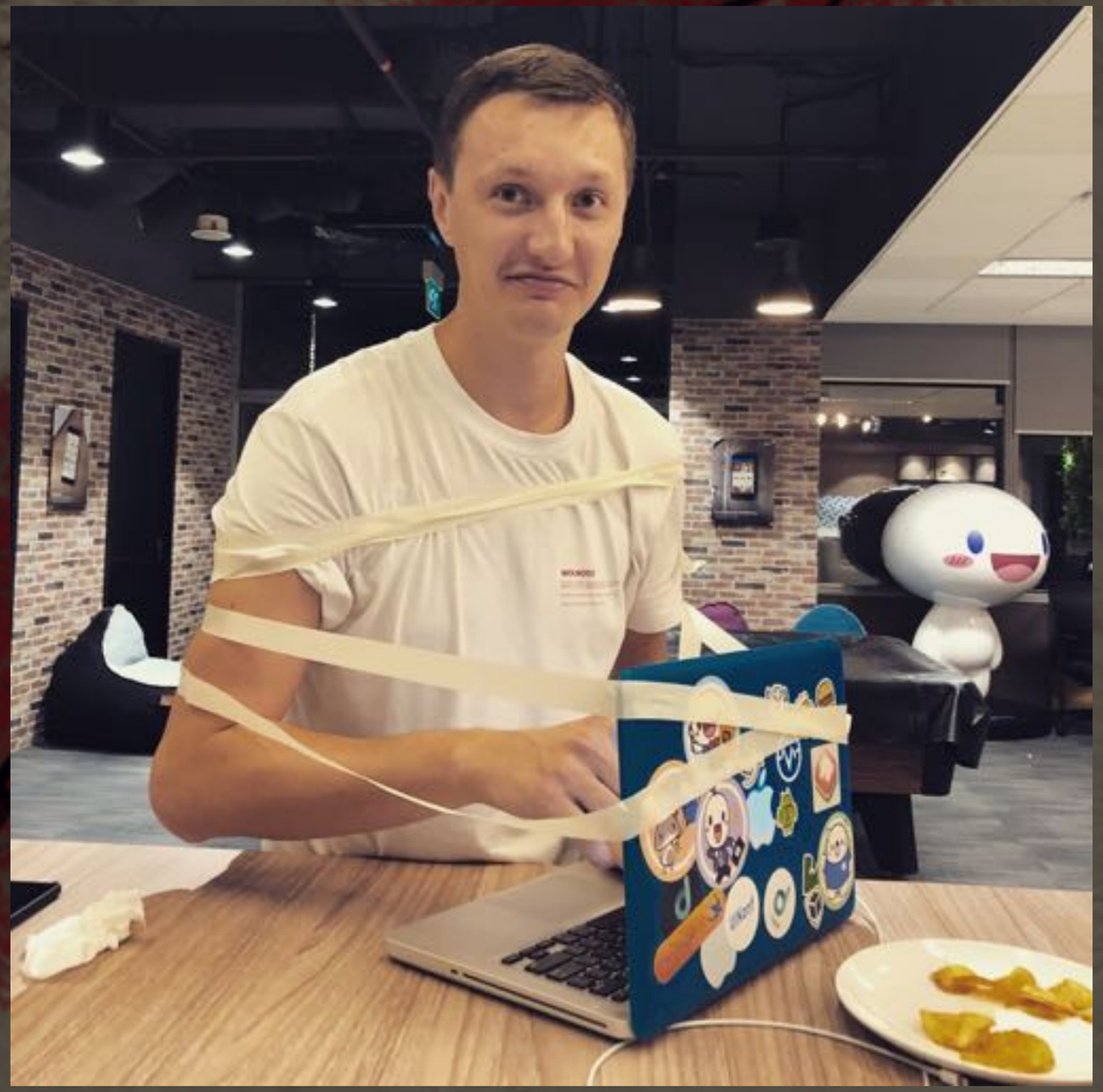
Avoiding the Quicksands of

a Legacy Codebase

Maciej Piotrowski


7:48
allegro
Czego szukasz?
KATEGORIE
Inspiracje
Elektronika
Moda
Dom i ogród
Odbierz 5 monet
na pierwsze zakupy w aplikacji Allegro.
Kolejne 5 monet otrzymasz po zakupie.
Promocja dotyczy nowych użytkowników aplikacji.
ODBIERZ MONETY
SAMSUNG
Galaxy S10+ | Watch Active
Kup Galaxy S10 lub S10+,
a Galaxy Watch Active
otrzymasz w prezencie.
SPRAWDŹ
allegroSMART!
week
TANIEJ NAWET O
70%
Strona główna
Szukaj
Koszyk
Obserwowane
Moje Allegro


We said goodbye to
Consumer on May 31, 2019.
Thank you for being a part of the
experience!
5:31
Thank you!
5:31



Egyptology

the study of the language, history, and culture of ancient Egypt

```
@implementation CACodeArcheologyDig

- (void)digDeeperIfNeeded {
                    [self selectPair];
                    [self digDeeper];
                    [self unearthMysteries];
                    [self preserveArtifacts];

}

@end
```

Pair Programming

Task: change UI for a message

- 💬 Update existing text styles for message types X, Y, Z
- Add a text style for a NEW message type 💬

if-ology

the science of how ifs and switches work

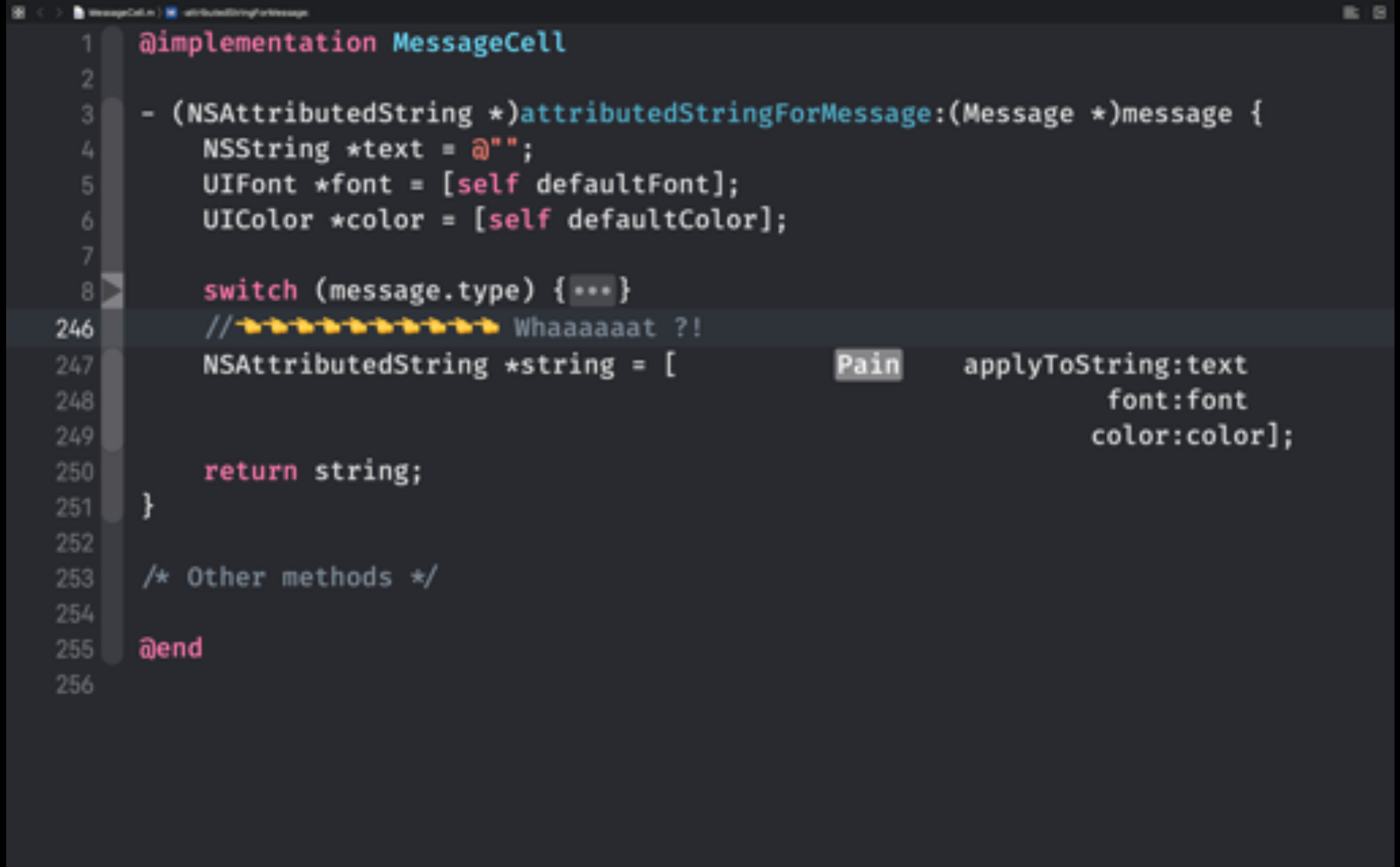
```objc
@implementation MessageCell

- (NSAttributedString *)attributedStringForMessage:(Message *)message {
    NSString *text = @"";
    UIFont *font = [self defaultFont];
    UIColor *color = [self defaultColor];

    switch (message.type) {•••}
    //🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥 Whaaaaaat ?!
    NSAttributedString *string = [AttributesPainter applyToString:text
                                                              font:font
                                                             color:color];
    return string;
}

/* Other methods */

@end
```


@implementation MessageCell
- (NSAttributedString *)attributedStringForMessage:(Message *)message {
NSString *text = @"";
UIFont *font = [self defaultFont];
UIColor *color = [self defaultColor];
switch (message.type) {•••}
// Whaaaaaat ?!
NSAttributedString *string = [Pain applyToString:text
font:font
color:color];
return string;
}
/* Other methods */
@end


if let a = a {
if let b = b {
if let c = c {
if let d = d {
if let e = e {
if let f = f {
if let g = g {
if let h = h {
if let i = i {
if let j = j {
//How many steps down can it go?
}
}
}
}
}
}
}
}
}
}

Task [updated]: change UI for a message

- Understand how things work
- Change existing logic
- Don't break things

Obstacles

- old code 💻
- no documentation 📖
- no tests ⛔
- no one knows how it was suppose to work 🤷

Obstacles

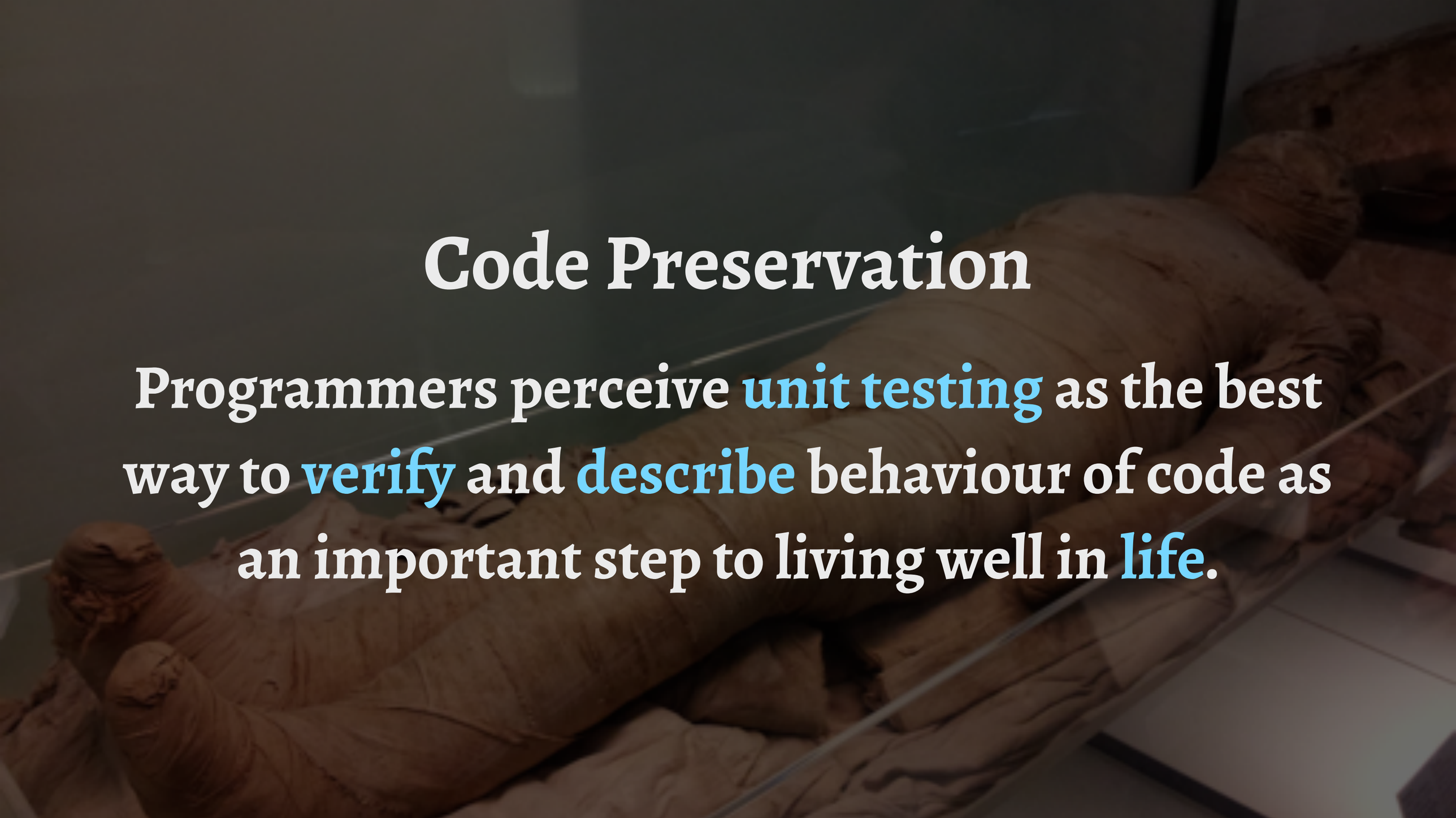
- old code 💻
- no documentation 📖
- no tests ⛔
- no one knows how it was suppose to work 🤷

What can you do?

- 🏃 quit your job immediately
- delete ⌫ the code and rewrite 💻 it
- preserve findings 🔍 and add new functionality ✅

Mummification

Egyptians saw the preservation of the body after death as an important step to living well in the afterlife.


Code Preservation
Programmers perceive unit testing as the best way to verify and describe behaviour of code as an important step to living well in life.

```objc
@implementation MessageCell

- (NSAttributedString *)attributedStringForMessage:(Message *)message {
    NSString *text = @"";
    UIFont *font = [self defaultFont];
    UIColor *color = [self defaultColor];

    switch (message.type) {•••}
    //🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥 Whaaaaaat ?!
    NSAttributedString *string = [AttributesPainter applyToString:text
                                                              font:font
                                                             color:color];
    return string;
}

/* Other methods */

@end
```

Unit Tests

```
prepare( Input ) -> putInto( Black Box ) -> verify( Output )
```

Unit Tests

```
prepare( Message ) -> putInto( MessageCell ) -> verify( AttributedString )
```

Unit Tests

```
func testAttributedStringForMessageWithParamXYZ {
    //Arrange
    let message = Message(type: type1,
                          body: "Hello Code Archeologists!")

    //Act
    let artefact = cell.attributedStringForMessage(message)

    //Assert
    //...
    artefact.color.getRed(&red, green: &green, blue: &blue, alpha: &alpha)
    XCTAssertEqual(red, expectedRed)
    XCTAssertEqual(green, expectedGreen)
    //...
}
```

Logic Shift

```
func testAttributedStringForMessageWithParamXYZ {
    //Arrange
    let message = Message(type: type1,
                          body: "Hello Code Archeologists!")

    //Act
    let artefact = 👉viewModel👈.attributedStringForMessage(message)

    //Assert
    XCTAssertEqual(artefact.font, expectedFont)
    XCTAssertEqual(artefact.color, expectedColor)
}
```

No Doom

When going through the pyramid

Archeology

the study of human history through the excavation of sites and the analysis of artefacts

Archeological Dig

Task: Performance improvement

- improve the cold start of the application 📱

Finding an entry point

```
@implementation UIApplicationDelegate

- (BOOL)application:(UIApplication *)application
        didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

        /*...*/

}

@end
```

```objc
#import <Foundation/Foundation.h>
#import "AppDelegate.h"

#import "Component1.h"
#import "Component2.h"
#import "Component3.h"
#import "Component4.h"
#import "Component5.h"
#import "Component6.h"
#import "Component7.h"
#import "Component8.h"
#import "Component9.h"
#import "Component10.h"
#import "Component11.h"
#import "Component12.h"
#import "Component13.h"
#import "Component14.h"
#import "Component15.h"
#import "Component16.h"
#import "Component17.h"
#import "Component18.h"
#import "Component19.h"
#import "Component20.h"
#import "Component21.h"
#import "Component22.h"
#import "Component23.h"

#import "Component24.h"
#import "Component25.h"
#import "Component26.h"
#import "Component27.h"
#import "Component28.h"
#import "Component29.h"
#import "Component30.h"
#import "Component31.h"
#import "Component32.h"
#import "Component33.h"
#import "Component34.h"
#import "Component35.h"
#import "Component36.h"
#import "Component37.h"
#import "Component38.h"
#import "Component39.h"
#import "Component40.h"
#import "Component41.h"
#import "Component42.h"
#import "Component43.h"
#import "Component44.h"
#import "Component45.h"
#import "Component46.h"
#import "Component47.h"
#import "Component48.h"
#import "Component49.h"
#import "Component50.h"
#import "Component51.h"
#import "Component52.h"
#import "Component53.h"
#import "Component54.h"
#import "Component55.h"
#import "Component56.h"
#import "Component57.h"
#import "Component58.h"
#import "Component59.h"
#import "Component60.h"
#import "Component61.h"
#import "Component62.h"
#import "Component63.h"
#import "Component64.h"
#import "Component65.h"
#import "Component66.h"
#import "Component67.h"
#import "Component68.h"
#import "Component69.h"
#import "Component70.h"
#import "Component71.h"
#import "Component72.h"
#import "Component73.h"
#import "Component74.h"
#import "Component75.h"
#import "Component76.h"
#import "Component77.h"
#import "Component78.h"
#import "Component79.h"
#import "Component80.h"
#import "Component81.h"
#import "Component82.h"
#import "Component83.h"
#import "Component84.h"
#import "Component85.h"
#import "Component86.h"
#import "Component87.h"

#import "Component88.h"
#import "Component89.h"
#import "Component90.h"
#import "Component91.h"
#import "Component92.h"
#import "Component93.h"

#import "Component94.h"
#import "Component95.h"
#import "Component96.h"
#import "Component97.h"
#import "Component98.h"

@interface AppDelegate ()<SomeProtocol>
{
    id ivar_1_with_not_necessarily_meaningful_name;
    id ivar_2_with_not_necessarily_meaningful_name;
    BOOL ivar_3_with_not_necessarily_meaningful_name;
    BOOL ivar_4_with_not_necessarily_meaningful_name;
    BOOL somethingUpdated;
}

@property (strong) id property1;
@property (strong) id property2;

@property (assign) BOOL flag_1_responsible_for_something_super_important;
@property (assign) BOOL flag_2_responsible_for_something_super_important;
@property (assign) BOOL flag_3_responsible_for_something_super_important;
@property (assign) BOOL flag_4responsible_for_something_super_important;

@end


@implementation AppDelegate

void uncaughtExceptionOccured(NSException *exception) {
    //Log and save crash
}

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

    NSSetUncaughtExceptionHandler(&uncaughtExceptionOccured);

    [Component56 start];

    [Component34 startLogging];

#if UI_TESTS_RUN
    [[[[[UIApplication sharedApplication] windows] objectAtIndex:0] layer] setSpeed:100.f]; // ❤️❤️❤️❤️❤️❤️❤️❤️❤️❤️
#endif

    [Component11 startMeasuringAppStartTime]; //❤️❤️❤️❤️❤️❤️❤️❤️❤️❤️ ??? ⬆️

    // Description why the code is needed
    [Component3 persistAndLogApplicationVersion];

    [[Component9 sharedInstance] doSomethingImportant]; //❤️❤️❤️❤️❤️❤️❤️❤️❤️❤️+ system singletons: NotificaionCenter, AVAudioSession

    // Description why the code is needed
    [[Component17 sharedInstance] registerNotificationActions];

    [self configureWhatAppShouldDoWhenWeGetNotifications];

    //Start services only if /*explanation when it is possible*/
    [self startAllServices_OnlyDoThatIfTheConditionIsMet]; // 🤢🤢🤢🤢🤢🤢🤢

    if (![Component7 hasStaleFiles]) { //❤️❤️❤️❤️❤️❤️❤️❤️❤️❤️
        [Component7 deleteStaleFiles]; //❤️❤️❤️❤️❤️❤️❤️❤️❤️❤️
    }

    if(launchOptions != nil && [[Component18 sharedInstance] isStarted]) {
        [self doThatOnlyIfNeeded];
    }

    // Description why the code is needed
    [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(applicationProtectedDataDidBecomeAvailable:) name:UIApplicationProtectedDataDidBecomeAvailable object:nil];

    // Another description why the code is needed
    [[AVAudioSession sharedInstance] setCategory:AVAudioSessionCategoryAmbient error:nil];

    dispatch_async(dispatch_get_main_queue(), ^{
        //This is needed to initialize the manager
        [Component45 sharedInstance];
    });

    // Yet another description why the code is needed
    NSDateFormatter *dateFormatter = [NSDateFormatter new];
    dateFormatter.dateFormat = @"yyyy-MM-dd'T'HH:mm:ssZ";
    // US English Locale (en_US) // ❤️❤️❤️❤️❤️❤️❤️❤️❤️❤️
    dateFormatter.locale = [NSLocale localeWithLocaleIdentifier:@"en_US"];
    [Component47 log:[NSString stringWithFormat:@"Today is %@", [dateFormatter stringFromDate:[NSDate date]]]];

    if (!self.launchSuccessful) {
        // Again a description why the code is needed
        if (![Component21 iseUserLogged]) {
            _ivar_2_with_not_necessarily_meaningful_name = YES;
        }
    }
    [[UIApplication sharedApplication] setStatusBarStyle:[Component33 statusBarStyle]]; //❤️❤️❤️❤️❤️❤️❤️❤️❤️❤️

    // Multiline description of the fix for some problem //❤️❤️❤️❤️❤️❤️❤️❤️❤️❤️
    // another line
    // another line
    // another line
    // another line
    // another line
    // another line
    // Can't touch this! //❤️❤️❤️❤️❤️❤️❤️❤️❤️❤️

    UITextField *textField = [[UITextField alloc] initWithFrame:CGRectZero];
    [application.keyWindow addSubview:textField]; //❤️❤️❤️❤️❤️❤️❤️❤️❤️❤️dirty fix can be tested
    [textField becomeFirstResponder];
    [textField resignFirstResponder];
    [textField removeFromSuperview];

    // Multiline description of the fix for some problem
    // another line
    // another line
    // another line
    // another line
    // another line
    // another line

    if([self isRunningOnOldIOS]) {
        for (UIWindow *window in [[UIApplication sharedApplication] windows]) {
            if (window.rootViewController == nil) {
                window.rootViewController = [[UIViewController alloc] initWithNibName:nil bundle:nil];
            }
        }
    }

    self.flag_2_responsible_for_something_super_important = YES;

    [Component76 startAndDo:^(NSError *error) {
        NSLog(@"Component76 started.");
    }];

    return YES;
}

- (void)startAllServices_OnlyDoThatIfTheConditionIsMet {

    //🤢🤢🤢🤢🤢🤢🤢again a few ifs that could be easily tested
    if(self.flag_3_responsible_for_something_super_important) {
        return;
    }

    // Description why the code is needed
     if(![[Component57 sharedInstance] start]) {
        self.flag_1_responsible_for_something_super_important = NO;
        return;
    }

    //🤢🤢🤢🤢🤢🤢🤢singletonos -> DI + interaction tests

    [[Component39 sharedInstance] startWithPath:[[Component38 sharedInstance] path]];

    self.flag_4responsible_for_something_super_important = NO; //🤢🤢🤢🤢🤢🤢🤢state change tests

    [[Component24 sharedInstance] startSomething];
    [Component41 start];

    [self.window makeKeyAndVisible];
    [self.window setBackgroundColor:[UIColor blackColor]];

    [[Component87 sharedInstance] listenForBadgeNotifications];

    // Description why the code is needed
    [self setAppearanceOfUIElements];

    Component75 *component75 = [[Component75 alloc] init];
    [Componoent88 sharedInstance].component75 = component75;
    [component75 start];

    // Description why the code is needed
    // Try-Catch is used here just for safety
    dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_BACKGROUND, 0), ^{
        //🤢🤢🤢🤢🤢🤢🤢🤢🤢🤢🤢
        State state = StateError;
        @try {
            state = isSomething();
            [Component34 log:[NSString stringWithFormat:@"State: %d", (int)state]];
        } @catch(NSException *e) {
            [Component34 log:[NSString stringWithFormat:@"Check failed"]];
        }

        [[Component48 sharedInstance] doSomethingWithState: state];
    });

    if(self.property2) {
        [[Component79 sharedInstance] notifyAbout:self.property2];
        self.property2 = nil;
    }

    // Description why the code is needed
    if([[UIApplication sharedApplication] applicationState] == UIApplicationStateActive) { //🤢🤢🤢🤢🤢🤢🤢input param of the function
        [Component1 sendUpdate:kUpdateAppActive];
    }

    [self activateAllComponents];

    self.flag_3_responsible_for_something_super_important = YES;
}

// Description when the method should be called.
- (void)activateAllComponents {
    if(self.flag_3_responsible_for_something_super_important) {
        return;
    }

    // Description why the code is needed
    [[Component89 sharedInstance] activateComponentFeatureIfNotStarted];

    // Multiline description of the fix for some problem
    // another line
    // another line
    // another line
    // another line
    // another line
    // another line

    [Component1 sendUpdate:kUpdateAppVisible];

    [[Component86 sharedInstance] activateIfNecessary]; // Description why the code is needed

    // Description why the code is needed
    if (_somethingUpdated) {
        [[Component96 sharedInstance] updateSomething];
        _somethingUpdated = NO;
    }

    // Multiline description of the fix for some problem
    // another line
    dispatch_async(dispatch_get_main_queue(), ^{
        [[Component92 sharedInstance] startSomething];
    });

    // Multiline description of the fix for some problem
    // another line
    static dispatch_once_t onceToken;
    dispatch_once(&onceToken, ^{
        [Component1 updateSomethingElse];
    });

    [[Component1 sharedInstance] setSomethingImportant:[[Component2 aProperty] somethingImportant]];

    self.flag_3_responsible_for_something_super_important = YES;
}

@end
```

AppDelegate.m

```
#import <Foundation/Foundation.h>
#import "AppDelegate.h"

#import "Component1.h"
#import "Component2.h"
#import "Component3.h"
#import "Component4.h"
#import "Component5.h"
#import "Component6.h"
#import "Component7.h"
#import "Component8.h"
#import "Component9.h"
#import "Component10.h"
#import "Component11.h"
#import "Component12.h"
#import "Component13.h"
#import "Component14.h"
#import "Component15.h"
#import "Component16.h"
#import "Component17.h"
#import "Component18.h"
#import "Component19.h"
#import "Component20.h"
#import "Component21.h"
#import "Component22.h"
#import "Component23.h"
```

AppDelegate.m

```
#import "Component94.h"
#import "Component95.h"
#import "Component96.h"
#import "Component97.h"
#import "Component98.h"

@interface AppDelegate() <SomeProtocol>
{
    id ivar_1_with_not_necessarily_meaningful_name;
    id ivar_2_with_not_necessarily_meaningful_name;
    BOOL ivar_3_with_not_necessarily_meaningful_name;
    BOOL ivar_4_with_not_necessarily_meaningful_name;
    BOOL somethingUpdated;
}

@property (strong) id property1;
@property (strong) id property2;

@property (assign) BOOL flag_1_responsible_for_something_super_important;
@property (assign) BOOL flag_2_responsible_for_something_super_important;
@property (assign) BOOL flag_3_responsible_for_something_super_important;
@property (assign) BOOL flag_4responsible_for_something_super_important;

@end
```

AppDelegate.m

```
- (BOOL)application:(UIApplication *)application
        didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

    NSSetUncaughtExceptionHandler(&uncaughtExceptionOccured);

    [Component56 start];

    [Component34 startLogging];

#if UI_TESTS_RUN
    [[[[UIApplication sharedApplication] windows] objectAtIndex:0]
        layer] setSpeed:100.f];
#endif

    //...
```

AppDelegate.m

```
- (BOOL)application:(UIApplication *)application
        didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

    NSSetUncaughtExceptionHandler(&uncaughtExceptionOccured);

    [Component56 start];

    [Component34 startLogging];

#if UI_TESTS_RUN
    [[[[👉[UIApplication sharedApplication]👈 windows] objectAtIndex:0]
        layer] setSpeed:100.f];
#endif

    //...
```

AppDelegate.m

```
- (BOOL)application:(UIApplication *)application 👈
        didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

    NSSetUncaughtExceptionHandler(&uncaughtExceptionOccured);

    [Component56 start];

    [Component34 startLogging];

#if UI_TESTS_RUN
    [[[[👉[UIApplication sharedApplication]👈 windows] objectAtIndex:0]
        layer] setSpeed:100.f];
#endif

    //...
```

Dependency Injection

```
protocol Dehydrating {

    func dehydrate(body: Body) -> DehydratedBody

}

protocol OrgansRemoving {

    func insert(_ hook: Hook, to: Body.Part)

    func pullOutBrain(from: Body) throws

    func remove(_ organ: Organ, from: Body)

}
```

Dependency Injection

```
class Mummyficator {

    let dehydrator: Dehydrating
    let organsRemover: OrgansRemoving
    let bandaging: Bandaging

    init(dehydrator: Dehydrating👈,
         organsRemover: OrgansRemoving👈,
         bandaging: Bandaging👈) {
         self.dehydrator = dehydrator
         //...
    }
}
```

Dependency Injection

```
class Mummyficator {
    //...

    func mummyfy(_ body: Body👈) -> Mummy🧟 {
        dehydrator.dehydrate(body)
        //...
    }

}
```

Dependency Injection

Dependency Injection

```
- (BOOL)application:(UIApplication *)application 👈
        didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

    NSSetUncaughtExceptionHandler(&uncaughtExceptionOccured);

    [Component56 start];

    [Component34 startLogging];

#if UI_TESTS_RUN
    [[[[👉[UIApplication sharedApplication]👈 windows] objectAtIndex:0]
        layer] setSpeed:100.f];
#endif

    //...
```

Dependency Injection

```
- (BOOL)application:(UIApplication *)application 👈
        didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

    NSSetUncaughtExceptionHandler(&uncaughtExceptionOccured);

    [Component56 start];

    [Component34 startLogging];

#if UI_TESTS_RUN
    [[[[👉application👈 windows] objectAtIndex:0]
        layer] setSpeed:100.f];
#endif

    //...
```

Measurements

```
- (BOOL)application:(UIApplication *)application
        didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    /* ... */

#if UI_TESTS_RUN
    [[[[[UIApplication sharedApplication] windows] objectAtIndex:0]
        layer] setSpeed:100.f];
#endif

    [Component11 startMeasuringAppStartTime]; // 👈 ⬆️⬆️⬆️

    // Description why the code is needed
    [Component3 persistAndLogApplicationVersion];
```

Measurements

```
- (BOOL)application:(UIApplication *)application
        didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    [Component11 startMeasuringAppStartTime]; // 👈 ⬆️⬆️⬆️
    /* ... */

#if UI_TESTS_RUN
    [[[[UIApplication sharedApplication] windows] objectAtIndex:0]
        layer] setSpeed:100.f];
#endif

    // Description why the code is needed
    [Component3 persistAndLogApplicationVersion];
```

Code Smells 💩

Code Fragrances 🌸

Comments (Everywhere)

```
- (BOOL)application:(UIApplication *)application
      didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

    /* ... */

    [Component11 startMeasuringAppStartTime];

    // Description why the code is needed 👈
    [Component3 persistAndLogApplicationVersion];

    [[Component9 sharedInstance] doSomethingImportant];
```

Comments (Everywhere)

```
- (BOOL)application:(UIApplication *)application
      didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

    /* ... */

    // Multiline description of the fix for some problem 👈
    // another line
    // another line
    // another line
    // another line
    // another line
    // another line
    // another line

    if([self isRunningOnOldIOS]) {
```

Comments (Everywhere)

```
- (BOOL)application:(UIApplication *)application
      didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

    /* ... */

    // Yet another description why the code is needed
    NSDateFormatter *dateFormatter = [NSDateFormatter new];
    dateFormatter.dateFormat = @"yyyy-MM-dd'T'HH:mm:ssZ";
    // US English Locale (en_US)
    dateFormatter.locale = [NSLocale localeWithLocaleIdentifier:@"en_US"🤔];
```

Comments (Everywhere)

```
- (BOOL)application:(UIApplication *)application
      didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

    /* ... */

    // Yet another description why the code is needed
    NSDateFormatter *dateFormatter = [NSDateFormatter new];
    dateFormatter.dateFormat = @"yyyy-MM-dd'T'HH:mm:ssZ";
    // US English Locale (en_US) 🤔
    dateFormatter.locale = [NSLocale localeWithLocaleIdentifier:@"en_US"🤔];
```

Singletons (Everywhere)

```
- (BOOL)application:(UIApplication *)application
      didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

    /* ... */

    // Description why the code is needed
    [Component3 persistAndLogApplicationVersion];

    [👉[Component9 sharedInstance]👈 doSomethingImportant];

    // Description why the code is needed
    [👉[Component17 sharedInstance]👈 registerNotificationActions];;
```

Singletons (Everywhere)

```
- (BOOL)application:(UIApplication *)application
      didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

    /* ... */

    // Description why the code is needed
    [Component3 persistAndLogApplicationVersion];

    [self.component9👈 doSomethingImportant];

    // Description why the code is needed
    [self.component17👈 registerNotificationActions];
```

Singletons (Everywhere)

```
UserDefaults.standard // 👈 inject
URLSession.shared // 👈 inject

DispatchQueue.main.async // 👈 write a wrapper to use sync
```

Singletons (Everywhere)

```
protocol WorkDispatching { //👈
    func dispatch(_ work: @escaping () -> Void)
}
```

Singletons (Everywhere)

```
protocol WorkDispatching { //👈
    func dispatch(_ work: @escaping () -> Void)
}

class AsyncWorkDispatcher: WorkDispatching {

    override func dispatch(_ work: @escaping () -> Void) {

        queue.async/* 👈 IMPORTANT! 👈 */(execute: work)

    }

}
```

Singletons (Everywhere)

```
protocol WorkDispatching { //👈
    func dispatch(_ work: @escaping () -> Void)
}

class SyncWorkDispatcher: WorkDispatching {

    override func dispatch(_ work: @escaping () -> Void) {

        queue.sync/* 👈 IMPORTANT! 👈 */(execute: work)

    }

}
```

Singletons (Everywhere)

```
@implementation Component1

+ (id)sharedInstance {
    static Component1 *shared = nil;
    static dispatch_once_t onceToken;
    dispatch_once(&onceToken, ^{
        shared = [[self alloc] init];
    });
    return shared;
}

@end
```

Singletons (Everywhere)

```
@implementation Component1

- (id)performOperations {
    [self performSuperImportantLogic];
    //...
    [[Component1 sharedInstance] getData];
    //...
    [self writeToDatabase];
}

@end
```

Singletons (Everywhere)

```
@implementation Component1

- (id)performOperations {
    [self performSuperImportantLogic];
    //...
    [👉[Component1 sharedInstance]👈 getData];
    //...
    [self writeToDatabase];
}

@end
```

Singletons (Everywhere)

```
@implementation Component1

- (id)performOperations {
    [self performSuperImportantLogic];
    //...
    [👉self👈 getData];
    //...
    [self writeToDatabase];
}

@end
```

Class Methods

```
- (BOOL)application:(UIApplication *)application
      didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

    /* ... */

    if (![Component7 hasStaleFiles]) {  👈
        [Component7 deleteStaleFiles];  👈
    }

    if(launchOptions != nil && [[Component18 sharedInstance] isStarted]) {
        [self doThatOnlyIfNeeded];
    }
```

Class Methods

```
@implementation Component7

+ (BOOL)hasStaleFiles {
    //...
    return NO;
}

@end
```

Class Methods

```
@implementation Component7
+ (id)sharedInstance { /*...*/ }

+ (BOOL)hasStaleFiles {
    //...
    return NO;
}

@end
```

Class Methods

```
@implementation Component7
+ (id)sharedInstance { /*...*/ }

- (BOOL)hasStaleFiles {
    //...
    return NO;
}

@end
```

Class Methods

```
@implementation Component7
+ (id)sharedInstance { /*...*/ }

- (BOOL)hasStaleFiles {
    //...
    return NO;
}

+ (BOOL)hasStaleFiles {
    return [[Component7 sharedInstance] hasStaleFiles];
}

@end
```

Class Methods

```
- (BOOL)application:(UIApplication *)application
      didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

    /* ... */

    if (![Component7 hasStaleFiles]) {  👈
        [Component7 deleteStaleFiles];  👈
    }

    if(launchOptions != nil && [[Component18 sharedInstance] isStarted]) {
        [self doThatOnlyIfNeeded];
    }
```

Class Methods

```
- (BOOL)application:(UIApplication *)application
      didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

    /* ... */

    if (![self.component7 hasStaleFiles]) {  👈
        [self.component7 deleteStaleFiles];  👈
    }

    if(launchOptions != nil && [[Component18 sharedInstance] isStarted]) {
        [self doThatOnlyIfNeeded];
    }
```

Untested Fixes

```
- (BOOL)application:(UIApplication *)application
      didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    /*...*/
    // Multiline description of the fix for some problem
    // another line
    // another line
    // Can't touch this! 💥

    if([self isRunningOnOldIOS]) {
        for (UIWindow *window in [[UIApplication sharedApplication] windows]) {
            if (window.rootViewController == nil) {
                window.rootViewController = [[ViewController alloc]
                      init];
            }
        }
    }
```

Untested Fixes

```
- (BOOL)application:(UIApplication *)application 👈
        didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    /*...*/
    // Multiline description of the fix for some problem
    // another line
    // another line
    // Can't touch this! 💥

    if([self isRunningOnOldIOS]) {
        for (UIWindow *window in [[UIApplication sharedApplication] 👈 windows]) {
            if (window.rootViewController == nil) {
                window.rootViewController = [[ViewController alloc]
                      init];
            }
        }
    }
```

Untested Fixes

```
- (BOOL)application:(UIApplication *)application 👈
        didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    /*...*/
    // Multiline description of the fix for some problem
    // another line
    // another line
    // Can't touch this! 💥

    if([self isRunningOnOldIOS]) {
        for (UIWindow *window in [application 👈 windows]) {
            if (window.rootViewController == nil) {
                window.rootViewController = [[ViewController alloc]
                        init];
            }
        }
    }
```

Unit Tests

```
prepare( Dependencies ) -> test( App Delegate ) -> verify( Interactions )
```

Unit Tests

```
func testApplicationDidFinishLaunchingWithOptions {
    //Arrange
    let app = ApplicationStub()

    //Act
    appDelegate.application(app, didFinishLaunchingWithOptions: [:])

    //Assert
    XCTAssertTrue(component1Stub.startCalled)
    XCTAssertTrue(component97Stub.activateCalled)
    //...
}
```

Happy End

- approach paid off 🤑
- logic maintained 👍
- ♻️ code ready for refactoring
- goal achieved -> performance improved 💪


Quicksands

Quicksands

- ✅ tests that don't actually test anything
- many 3rd party libraries without a wrapper 🎁
- many libraries doing the same ⚙️
- 🤴 complicated inheritance `class D: C: B: A: BaseClass`
- `ObjC` Massive View Controllers hidden 🦚 by `Swift` extensions

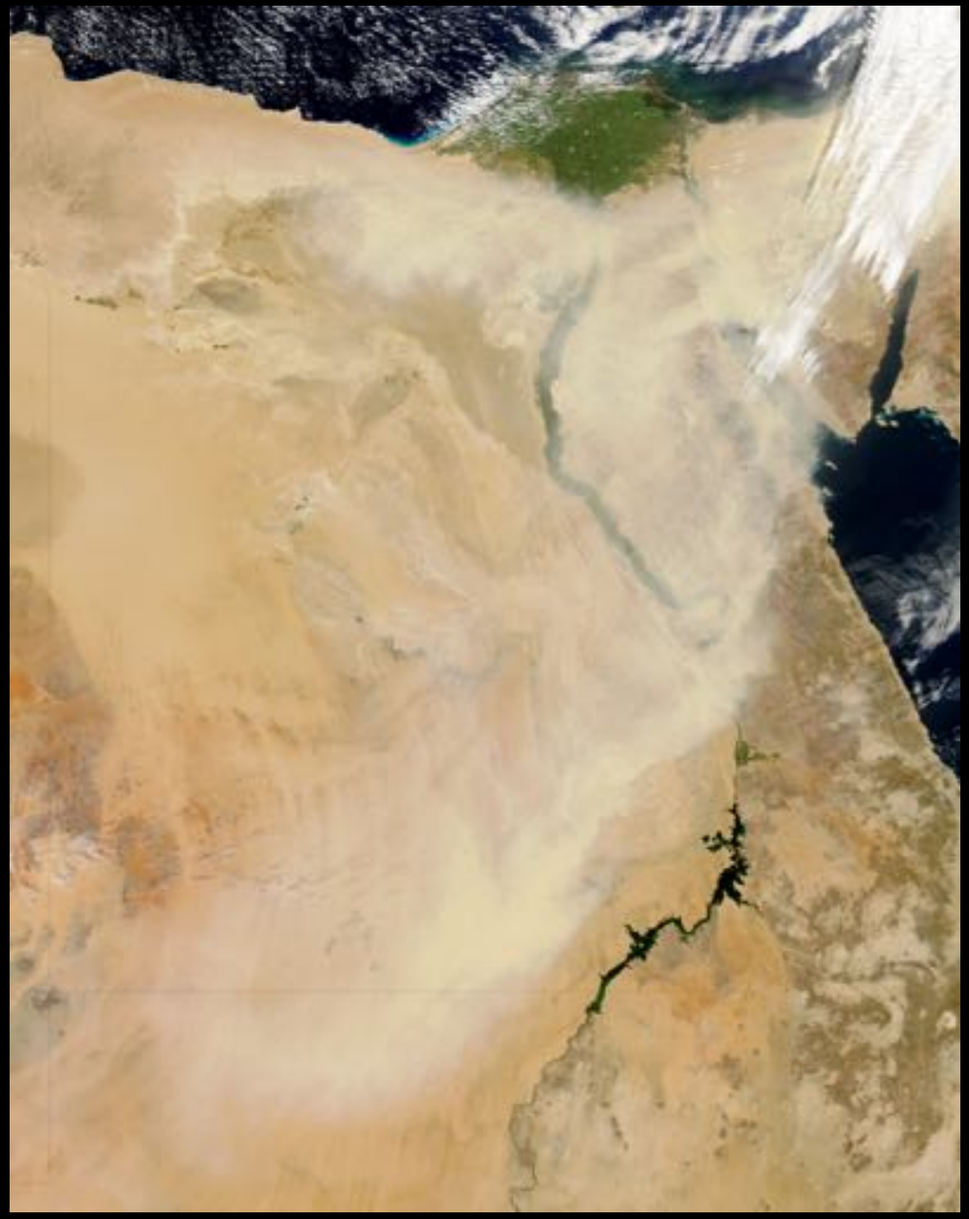

Legacy Code

Legacy Code

Every piece of code not covered by unit tests

Test Driven Development [TDD]

- red ⛔-> green ✅ -> refactor ♻️
- save time for you and others in the future ⏳
- 🧛 preserve information how your code works
- make reality less blurry 👀



Grazie mille!

swifting.io

maciej.piotrowski@swifting.io

What quicksands have you encountered in the codebase you worked on?