

7 Deadly Coding Sins

Presented By: Gary Rattray
Mobile Dev NJ MeetUp
November 9th (#11)

These will Kill you!

1. Duplication
2. Lack of Unit Tests
3. Coding Standards
4. Comments
5. Design Spaghetti
6. Complexity
7. Potential Bugs

Google them all...

- Programming
- Programmers
- Coding...

Nice Code Right?

```
- (void) setup {

    CGRect frame = CGRectMake(20, 100, 50, 22);
    UIButton *button1 = [[UIButton alloc] initWithFrame:frame];
    [button1 setTitle:@"Test1" forState:UIControlStateNormal];
    [button1 setBackgroundImage:[UIImage imageNamed:@"test1"]
                           forState:UIControlStateNormal];
    [button1 addTarget:self action:@selector(doSomething) forControlEvents:
     UIControlEventTouchUpInside];

    frame = CGRectMake(50, 100, 50, 22);
    UIButton *button2 = [[UIButton alloc] initWithFrame:frame];
    [button2 setTitle:@"Test2" forState:UIControlStateNormal];
    [button2 setBackgroundImage:[UIImage imageNamed:@"test2"]
                           forState:UIControlStateNormal];
    [button2 addTarget:self action:@selector(doSomething) forControlEvents:
     UIControlEventTouchUpInside];

}
```

Don't Duplicate Code

```
+ (UIButton *)buttonWithTitle: (NSString *)title
    target:(id)target
    selector:(SEL)selector
    frame:(CGRect)frame
    image:(UIImage *)image
{
    UIButton *button = [[UIButton alloc] initWithFrame:frame];
    [button setTitle:title forState:UIControlStateNormal];
    [button setBackgroundImage:image forState:UIControlStateNormal];
    [button addTarget:target action:selector
        forControlEvents:UIControlEventTouchUpInside];
    button.backgroundColor = [UIColor clearColor];
    return button;
}
```

Create Unit Tests

```
#import "Testable.h"

@implementation Testable

- (NSUInteger)add4:(NSUInteger)value
{
    return value + 4;
}

@end
```

Example

```
#import "UnitTestable.h"

@implementation Testable

- (void)setUp
{
    [super setUp];
    // Set-up code here.
    test_subject = [[Testable alloc] init];
    STAssertNotNil(test_subject, @"Could not create test subject");
}

- (void)tearDown
{
    // Tear-down code here.
    [test_subject release];
    [super tearDown];
}

- (void)testExample
{
    STAssertEquals([test_subject add4:5], (NSUInteger)9, @"should add 4");
}
```

Naming Methods

Code	Commentary
insert:at:	not clear; what is being inserted? what does "at" signify?
remove:	not clear; what is being removed?

Naming Methods

Method Name	Correctness
- (void)setGlyphInfoAccepted:(BOOL)flag;	wrong
- (BOOL)glyphInfoAccepted;	wrong

Naming Functions

Correct	Wrong
- <code>(NSSize)calcCellSize;</code>	<code>wrong</code>
- <code>(NSSize)getCellSize;</code>	<code>wrong</code>

Naming Methods

```
- (int)runModalForDirectory:(NSString *)path andFile:(NSString *)name andTypes:  
NSArray *)fileTypes;
```

wrong

Naming Functions

- (void)sendAction:(SEL)aSelector :(id)anObject :(BOOL)flag;	wrong

Naming Functions

Good	Bad
- (id)taggedView:(int)aTag;	wrong

Comments

```
#pragma mark
#pragma mark Button with Image
#pragma mark
- (void)createImageButton
{
    // create a UIButton with just an image instead of a title

    UIImage *buttonBackground = [UIImage imageNamed:@"whiteButton.png"];
    UIImage *buttonBackgroundPressed = [UIImage imageNamed:@"blueButton.
png"];

    CGRect frame = CGRectMake(0.0, 0.0, kStdButtonWidth, kStdButtonHeight)
    ;

    imageButton = [ButtonsViewController buttonWithType:@"
        target:self
        selector:@selector(action:)
        frame:frame
        image:buttonBackground
        imagePressed:buttonBackgroundPressed
        darkTextColor:YES];
}

[imageButton setImage:[UIImage imageNamed:@"UIButton_custom.png"]
forState:UIControlStateNormal];
}
```

What's this called?

```
10 i = 0
20 i = i + 1
30 PRINT i; " squared = "; i * i
40 IF i >= 10 THEN GOTO 60
50 GOTO 20
60 PRINT "Program Completed."
70 END
```

Here is the same code written in a structured programming style:

```
FOR i = 1 TO 10
    PRINT i; " squared = "; i * i
NEXT i
PRINT "Program Completed."
END
```

Spaghetti Code

Spaghetti code is a pejorative term for source code that has a complex and tangled control structure, especially one using many GOTOs, exceptions, threads, or other "unstructured" branching constructs.

Solutions

Rethink Control Structure, Classes, and
Libraries

Use Modeling Tools, Mockups, Use Cases
Build Modular Apps

Ask Yourself:

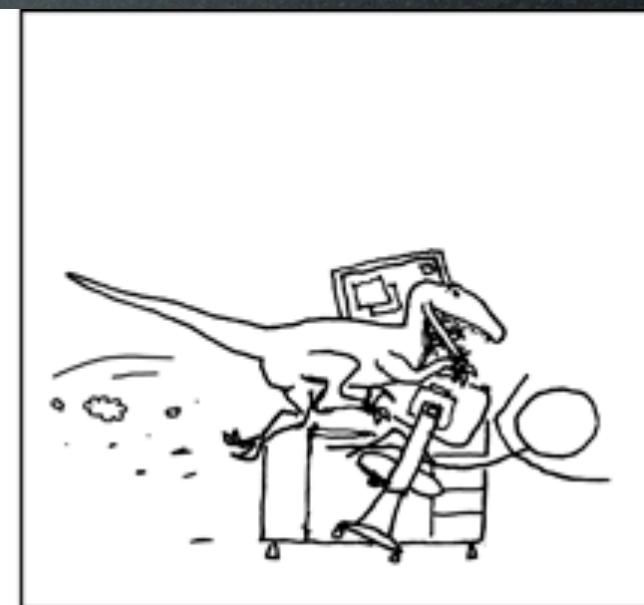
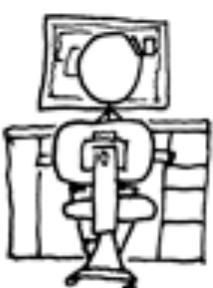
What do I need this particular piece of code
to do ?

What is this Class/Method responsible for?

Solutions

- ✓ Plan first, code only after you have a very good idea on how to solve the problem.
- ✓ Reuse first, create functional libraries if necessary.
- ✓ Use Object oriented inheritance and subclasses helps you add changes over time.
- ✓ Focus on encapsulation - e.g Button example.

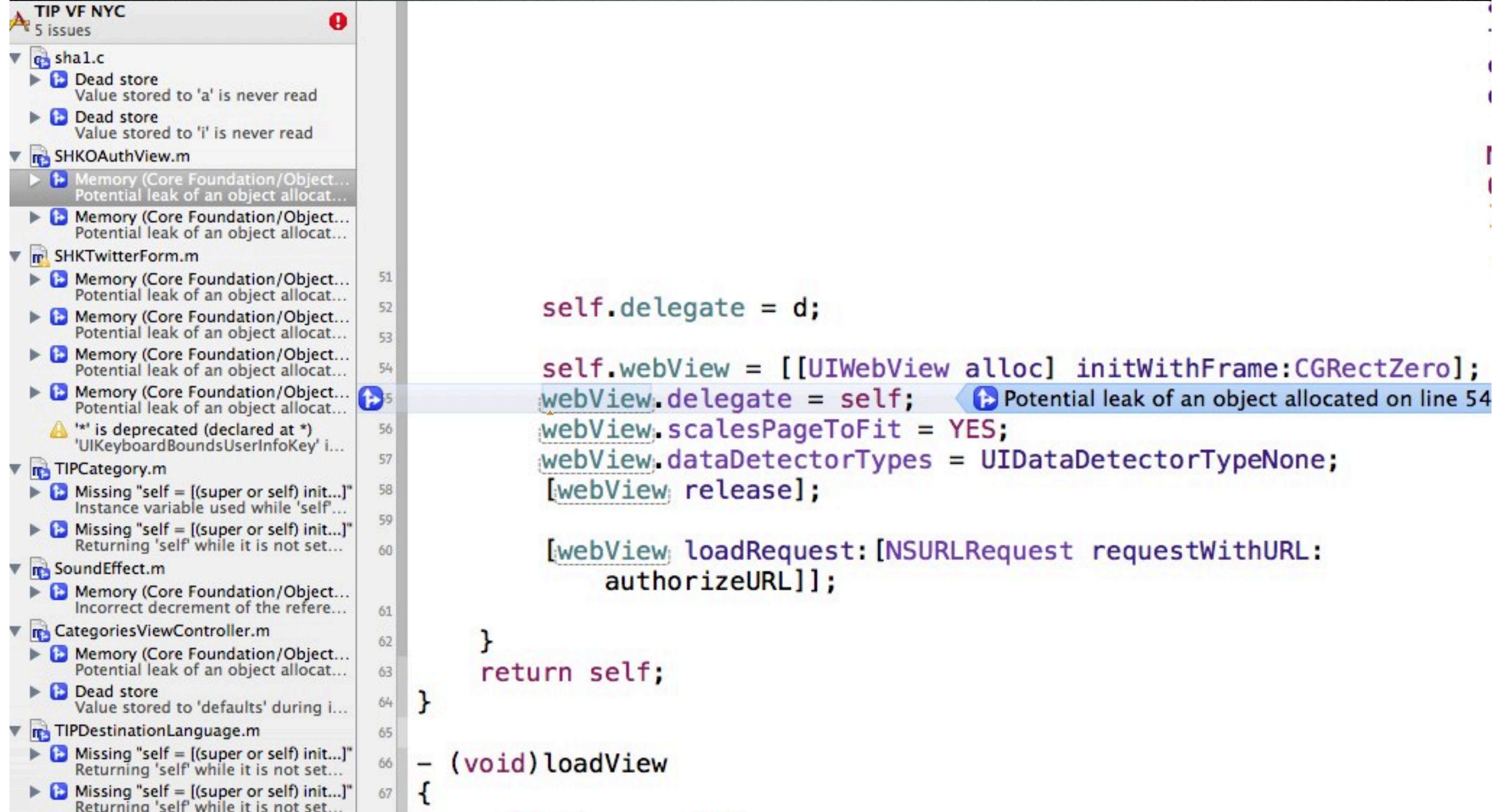
Design Spaghetti



Complexity

json-framework / Classes		
name	age	message
..		
NSObject+SBJson.h		Loadin
NSObject+SBJson.m		
SBJson.h		
SBJsonParser.h		
SBJsonParser.m		
SBJsonStreamParser.h		
SBJsonStreamParser.m		
SBJsonStreamParserAccumulator.h		
SBJsonStreamParserAccumulator.m		
SBJsonStreamParserAdapter.h		
SBJsonStreamParserAdapter.m		
SBJsonStreamParserState.h	June 18, 2011	Add m

Potential Bugs



The screenshot shows the Xcode IDE with the 'TIP VF NYC' project open. The left sidebar displays a list of files and their associated issues. The main editor area shows the code for `SHKOAuthView.m`, specifically the implementation of the `- (void)loadView` method.

TIP VF NYC
5 issues

- `sha1.c`
 - Dead store
Value stored to 'a' is never read
 - Dead store
Value stored to 'i' is never read
- `SHKOAuthView.m`
 - Memory (Core Foundation/Object... Potential leak of an object allocat...)
 - Memory (Core Foundation/Object... Potential leak of an object allocat...)
- `SHKTwitterForm.m`
 - Memory (Core Foundation/Object... Potential leak of an object allocat...)
 - Memory (Core Foundation/Object... Potential leak of an object allocat...)
 - Memory (Core Foundation/Object... Potential leak of an object allocat...)
 - Memory (Core Foundation/Object... Potential leak of an object allocat...)
 - Memory (Core Foundation/Object... Potential leak of an object allocat...)
 - Memory (Core Foundation/Object... Potential leak of an object allocat...)
 - /* is deprecated (declared at *) 'UIKeyboardBoundsUserInfoKey' i...
- `TIPCategory.m`
 - Missing "self = [(super or self) init...]" Instance variable used while 'self'...
 - Missing "self = [(super or self) init...]" Returning 'self' while it is not set...
- `SoundEffect.m`
 - Memory (Core Foundation/Object... Incorrect decrement of the refere...
- `CategoriesViewController.m`
 - Memory (Core Foundation/Object... Potential leak of an object allocat...)
 - Dead store
Value stored to 'defaults' during i...
- `TIPDestinationLanguage.m`
 - Missing "self = [(super or self) init...]" Returning 'self' while it is not set...
 - Missing "self = [(super or self) init...]" Returning 'self' while it is not set...

Code Snippet:

```
self.delegate = d;

self.webView = [[UIWebView alloc] initWithFrame:CGRectMakeZero];
webView.delegate = self; // Potential leak of an object allocated on line 54
webView.scalesPageToFit = YES;
webView.dataDetectorTypes = UIDataDetectorTypeNone;
[webView release];

[webView loadRequest:[NSURLRequest requestWithURL:
                     authorizeURL]];

}
return self;
}

- (void)loadView
{
```

Resources

- http://developer.apple.com/library/mac/#documentation/Unit_Testing_Applications/unit_testing_applications
- http://developer.apple.com/library/ios/#documentation/Unit_Testing_Applications/unit_testing_applications
- QUESTIONS?

Resources

- <http://developer.apple.com/library/mac/#documentation>
- http://developer.apple.com/library/ios/#documentation/Unit_Testing_Applications/unit_testing_applications.html
- QUESTIONS?