CS121 iOS Tutorial 3

This tutorial provides a basic introduction insturments in Xcode. This tutorial is based on Ray Wnderlich's tutorial available here: http://www.raywenderlich.com/2696/how-to-debug-memory-leaks-with-xcode-and-instruments-tutorial

Purple bubbles give you information you'll need to know.

Yellow Bubbles tell you what to do.

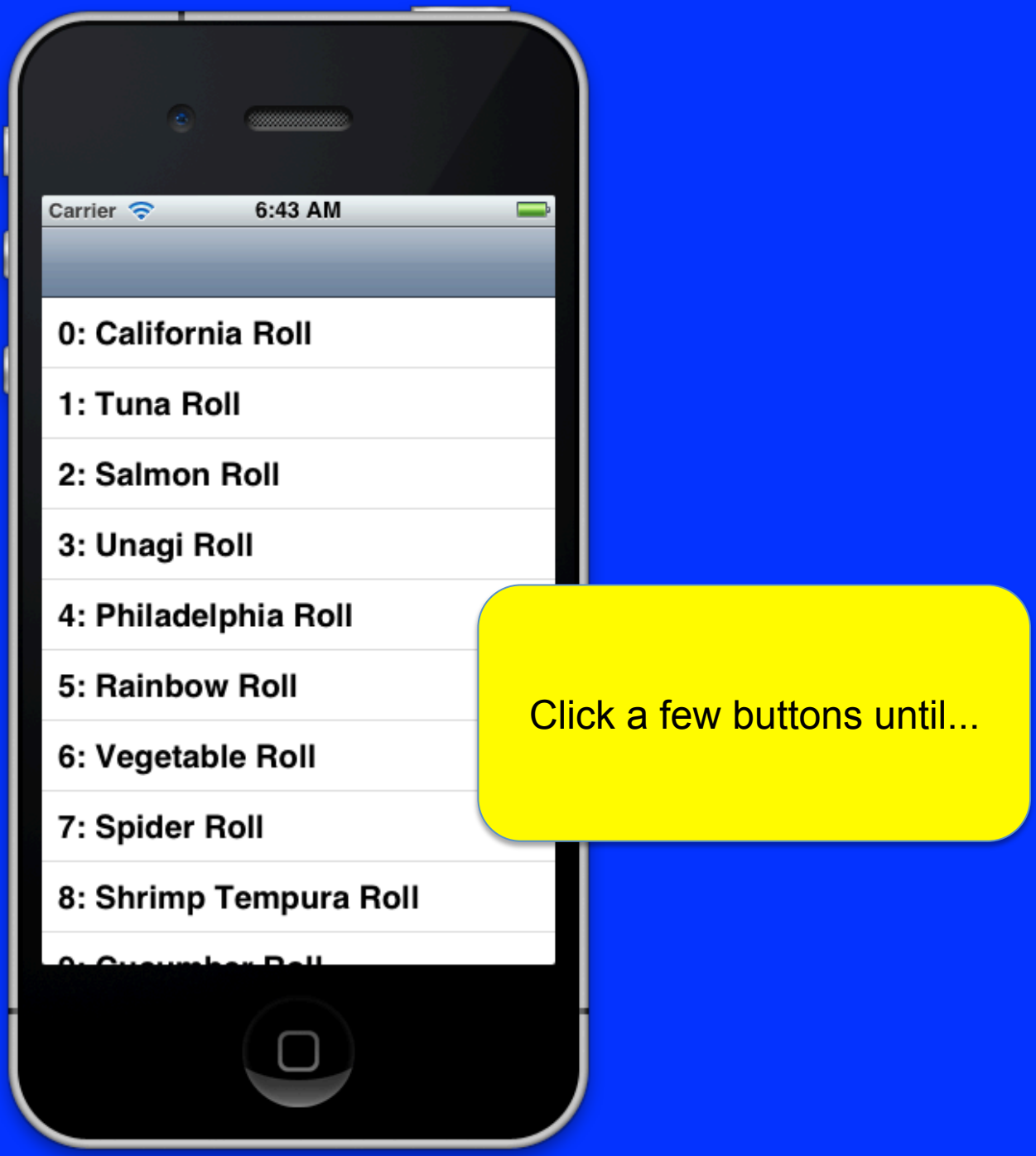
Orange bubbles tell you what you're not expected to understand yet. ☺

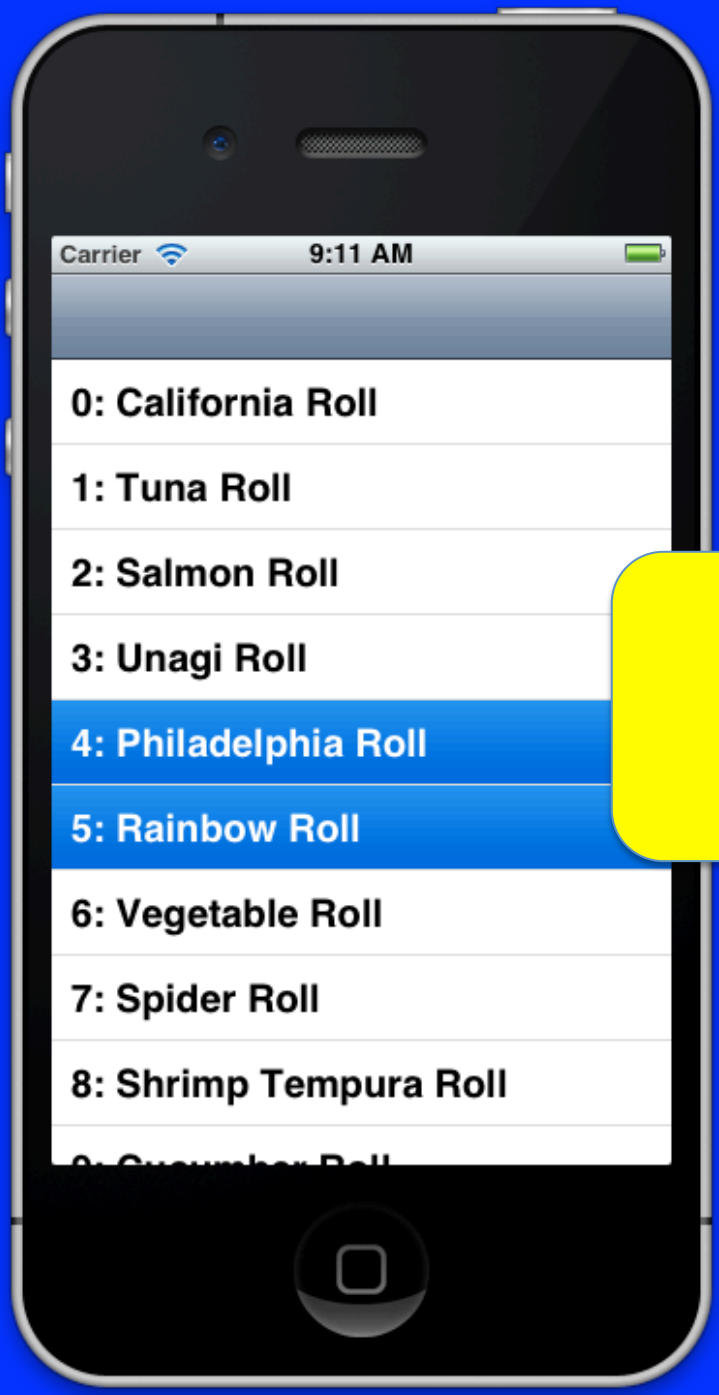
Download the test app from the course page

www.cs.hmc.edu/cs121/tutorials/leaky.zip

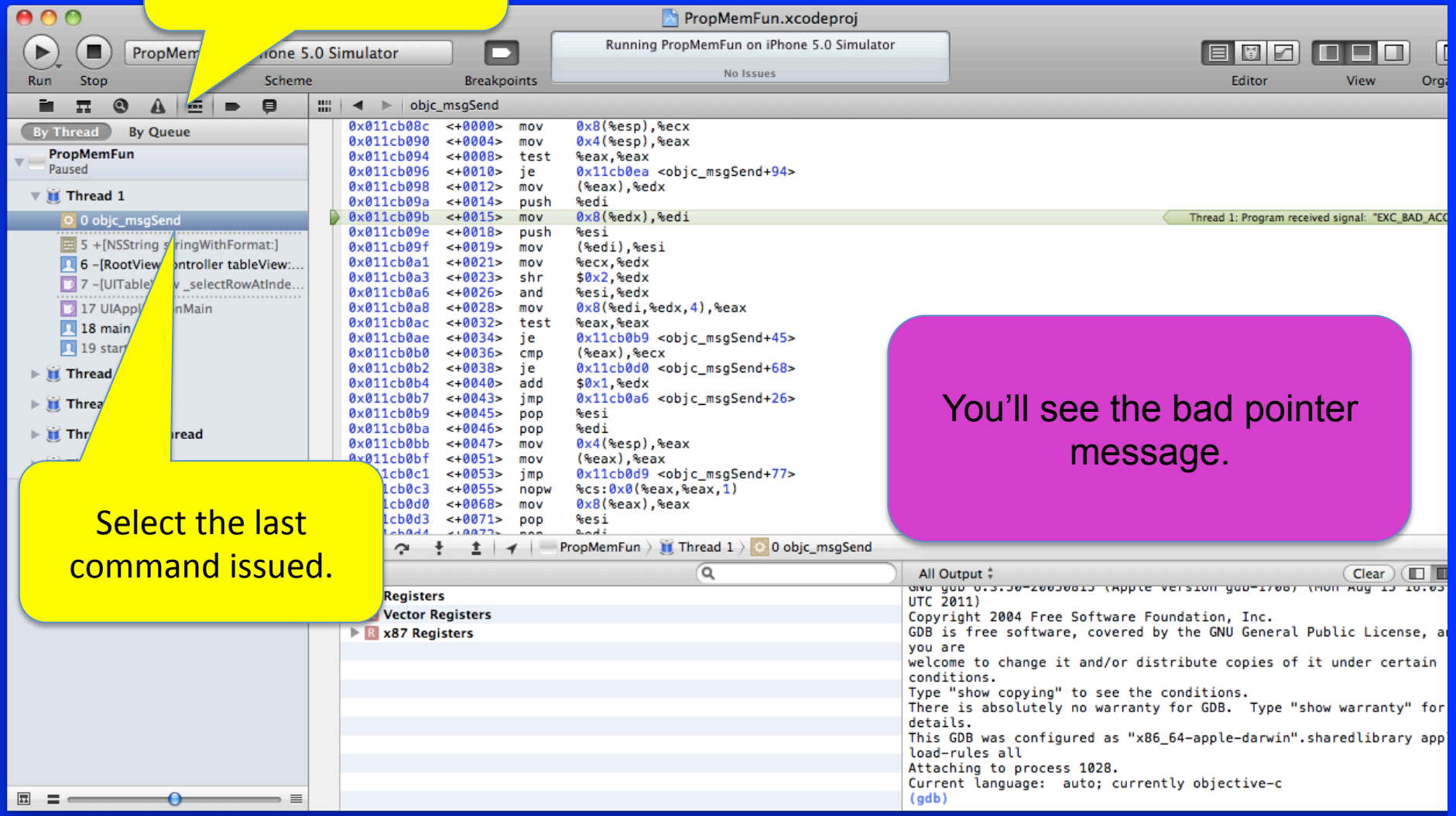
Unzip, it open the project folder.


PropMemFun.xcodeproj — RootViewController.m
Run
Stop
PropMemFun 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Running PropMemFun.app on iPhone 6.0 Simulator
Project 2
Editor
View
Organizer
PropMemFun
1 target, iOS SDK 6.0
Classes
Other Sources
Resources
Frameworks
Products
PropMemFun 〉 Classes 〉 RootViewController.m 〉 @implementation RootViewController
Set to run on the iPhone simulator then run.
All Output
Clear
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
This GDB was configured as "x86_64-apple-darwin".sharedlibrary apply-load-rules all
Attaching to process 19036.
2013-02-01 14:08:27.925 PropMemFun[19036:c07] Application windows are expected to have a root view controller at the end of application launch


Carrier
6:43 AM
0: California Roll
1: Tuna Roll
2: Salmon Roll
3: Unagi Roll
4: Philadelphia Roll
5: Rainbow Roll
6: Vegetable Roll
7: Spider Roll
8: Shrimp Tempura Roll
Click a few buttons until...


Carrier
9:11 AM
0: California Roll
1: Tuna Roll
2: Salmon Roll
3: Unagi Roll
4: Philadelphia Roll
5: Rainbow Roll
6: Vegetable Roll
7: Spider Roll
8: Shrimp Tempura Roll

… it crashes.



What is the problem?

PropMemFun.xcodeproj — RootViewController.m

PropMemFun › iPhone 5.0 Simulator

Running PropMemFun on iPhone 5.0 Simulator

No Issues

Run Stop Scheme Breakpoints Editor View

PropMemFun › Classes › RootViewController.m › -tableView:didSelectRowAtIndexPath:

By Thread By Queue

PropMemFun
Paused

Thread 1
0 ___forwarding___
6 +[NSString stringWithFormat:]
7 -[RootViewController tableView:...
8 -[UITableView _selectRowAtInde...
18 UIApplicationMain
19 main
20 start

Thread 2

Thread 4 WebThread

Thread 5

Thread 6

```
#pragma mark -
#pragma mark Table view delegate

- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath {

    NSString * sushiName = [_sushiTypes objectAtIndex:indexPath.row];
    NSString * sushiString = [NSString stringWithFormat:@"%d: %@", indexPath.row, sushiName];

    NSString * message = [NSString stringWithFormat:@"Last sushi: %@.  Cur sushi: %@",
        _lastSushiSelected, sushiString];
    UIAlertView *alertView = [[UIAlertView alloc] initWithTitle:@"Sushi Power!"
                                                        message:message
                                                       delegate:nil
                                              cancelButtonTitle:nil
                                              otherButtonTitles:@"OK", nil];

    [alertView show];

    _lastSushiSelected = sushiString;

}


#pragma mark -
#pragma mark Memory management
```

Thread 1

Why is this a zombie?

self = (RootViewController *) 0x6a10b20
UITableViewController = (UITableViewController) {...}
_sushiTypes = (__NSArrayI *) 0x6860690 14 objects
_lastSushiSelected = (_NSZombie_CFString *) 0x6a19740 <Zombie>
_cmd = (SEL) 0x822ead tableView:didSelectRowAtIndexPath:
tableView = (UITableView *) 0x704ce00

No Selection

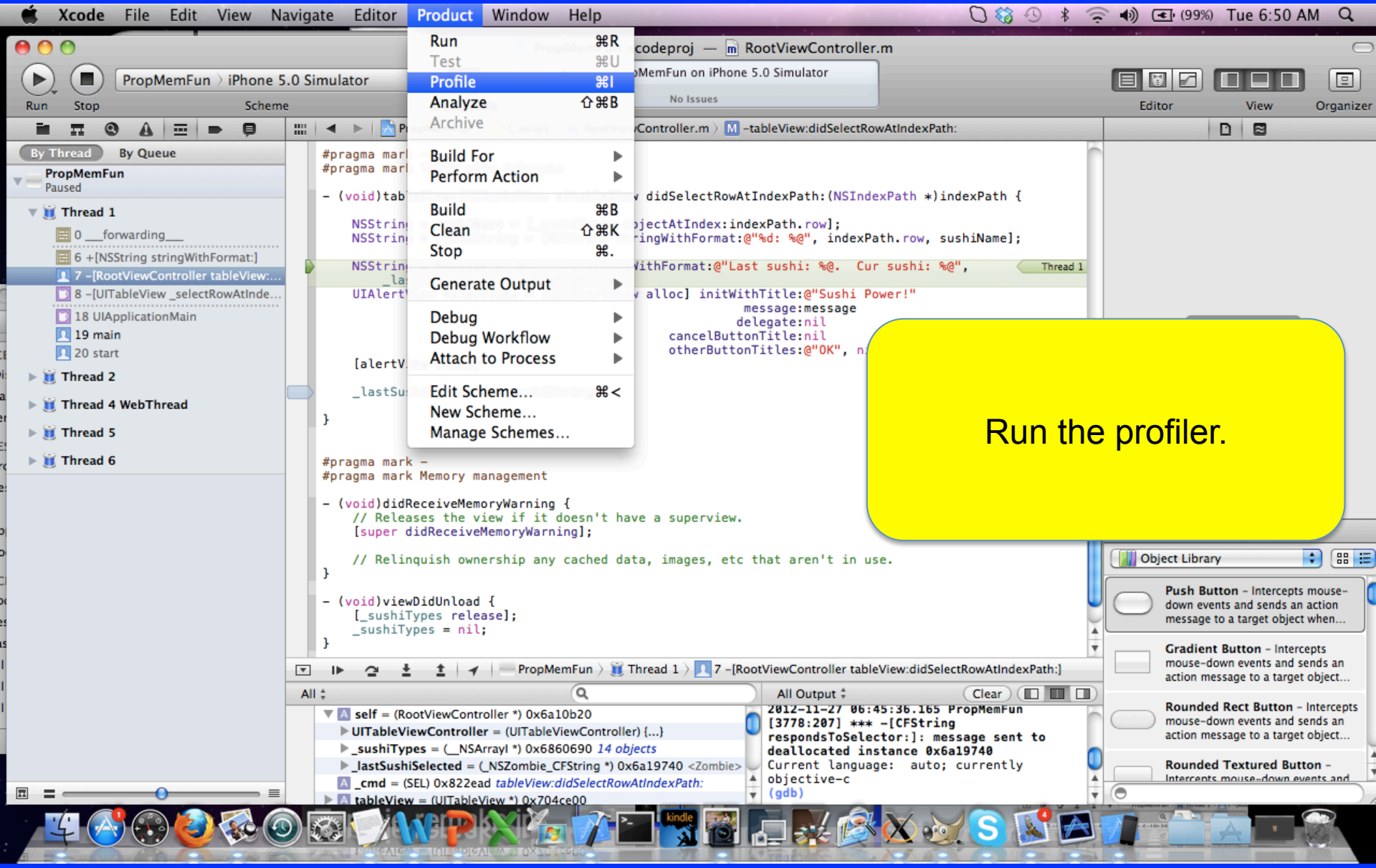
Object Library

Push Button – Intercepts mouse down events and sends an action message to a target object when.

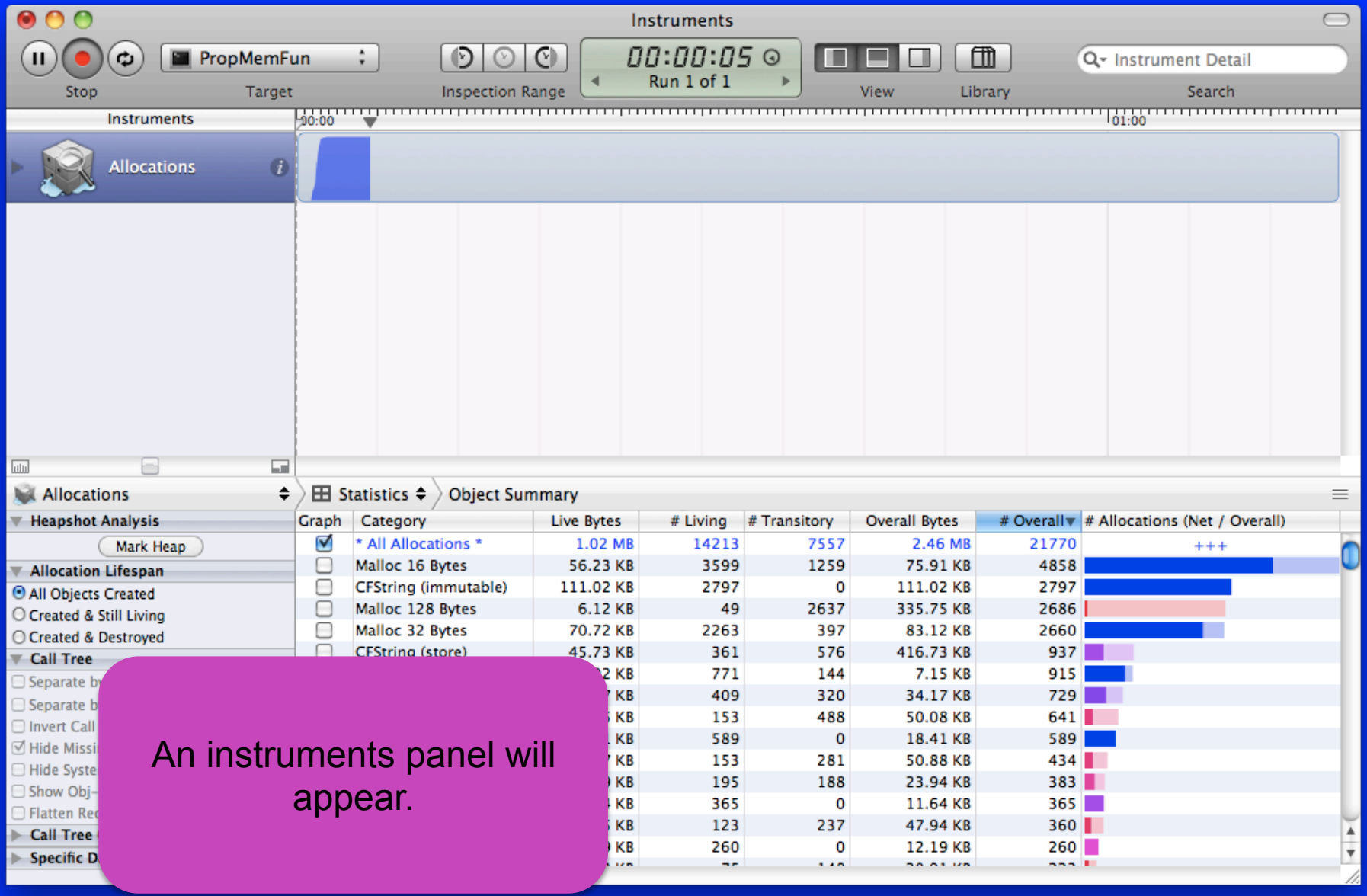
Gradient Button – Intercepts mouse-down events and sends a action message to a target object

Rounded Rect Button – Interce mouse-down events and sends a action message to a target object

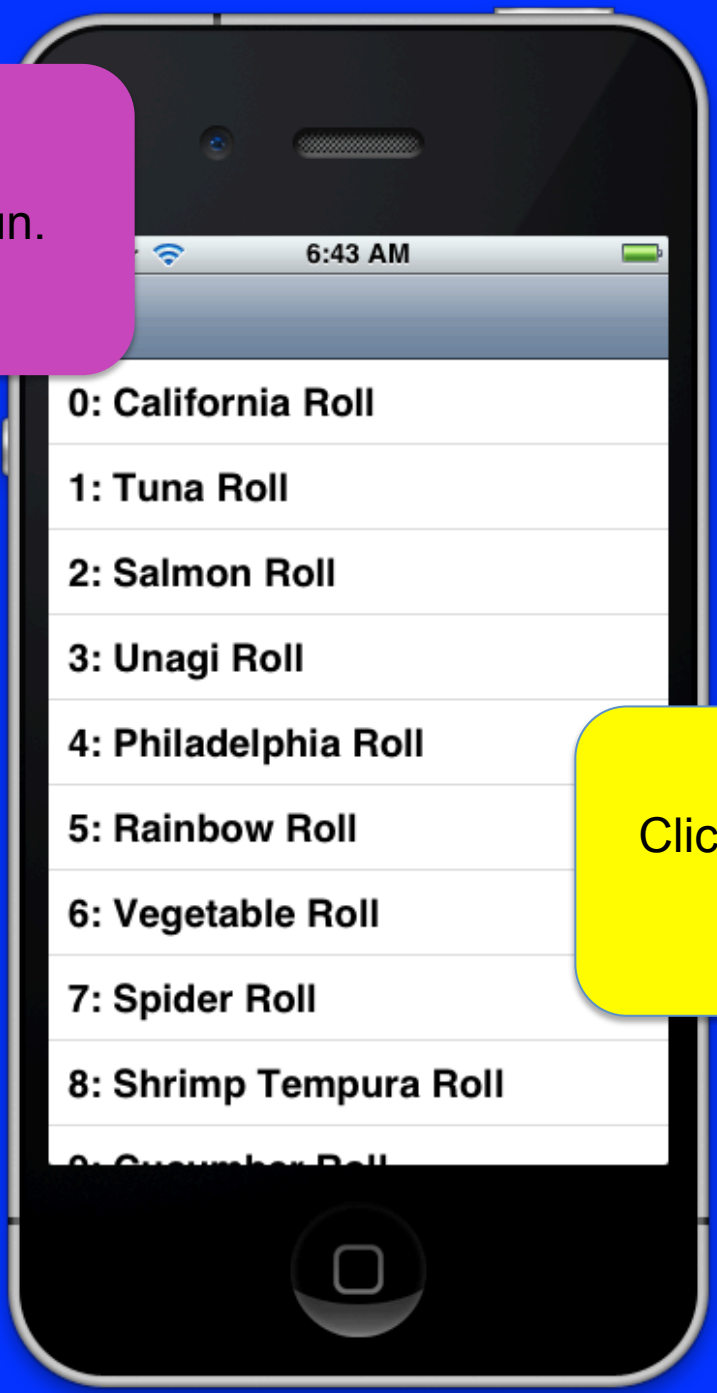
Rounded Textured Button –


Xcode File Edit View Navigate Editor Product Window Help
Run ⌘R
Test ⌘U
Profile ⌘I
Analyze ⇧⌘B
Archive
Build For
Perform Action
Build ⌘B
Clean ⇧⌘K
Stop ⌘.
Generate Output
Debug
Debug Workflow
Attach to Process
Edit Scheme... ⌘<
New Scheme...
Manage Schemes...
Run the profiler.

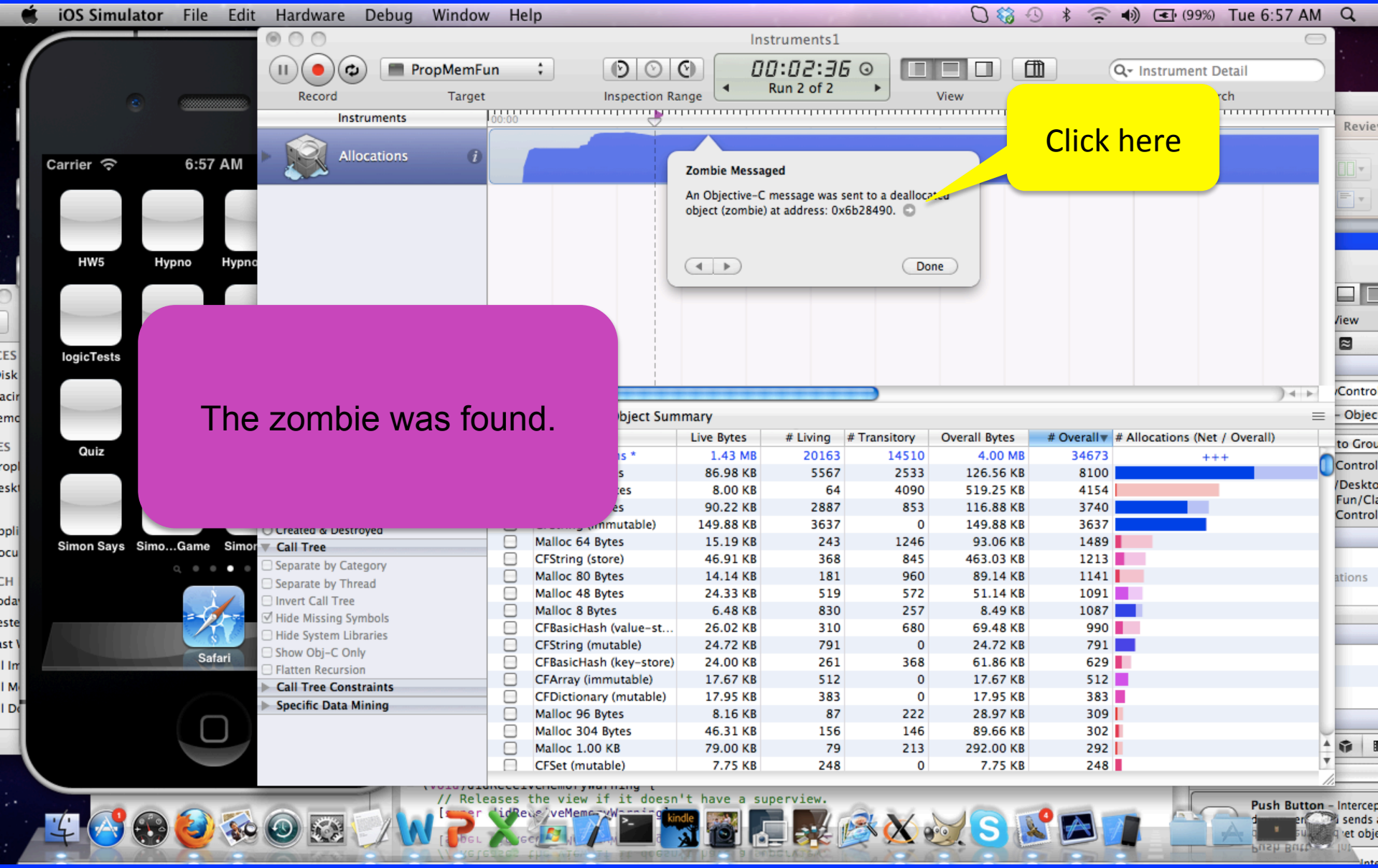
Select the Zombie tool.

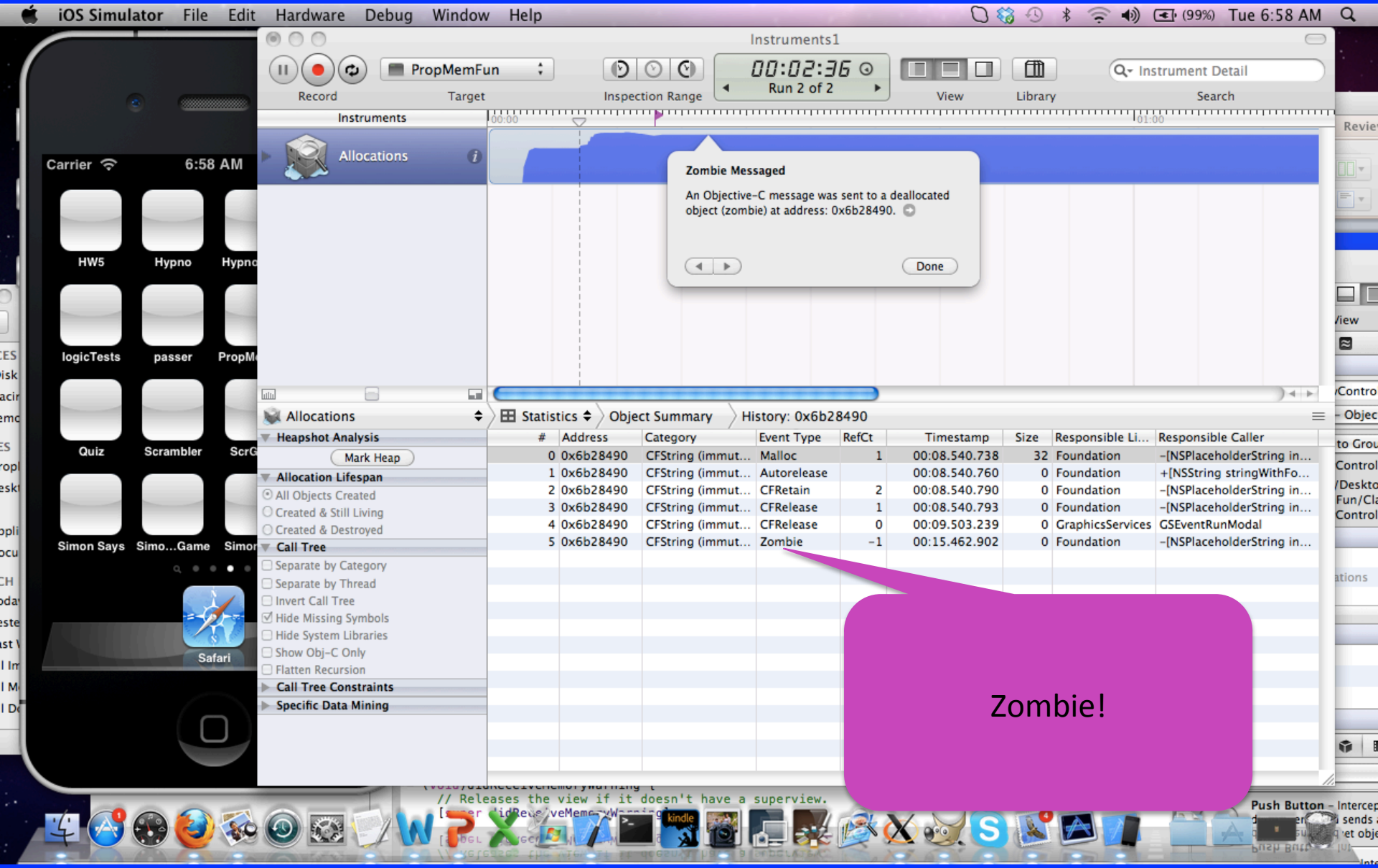

Instruments
Stop
PropMemFun
Target
Inspection Range
00:00:05
Run 1 of 1
View
Library
Instrument Detail
Search
Instruments
Allocations
Allocations
Statistics
Object Summary
Heapshot Analysis
Mark Heap
Allocation Lifespan
All Objects Created
Created & Still Living
Created & Destroyed
Call Tree
Graph
Category
Live Bytes
Living
Transitory
Overall Bytes
Overall
Allocations (Net / Overall)
* All Allocations * 1.02 MB 14213 7557 2.46 MB 21770 +++
Malloc 16 Bytes 56.23 KB 3599 1259 75.91 KB 4858
CFString (immutable) 111.02 KB 2797 0 111.02 KB 2797
Malloc 128 Bytes 6.12 KB 49 2637 335.75 KB 2686
Malloc 32 Bytes 70.72 KB 2263 397 83.12 KB 2660
CFString (store) 45.73 KB 361 576 416.73 KB 937
771 144 7.15 KB 915
409 320 34.17 KB 729
153 488 50.08 KB 641
589 0 18.41 KB 589
153 281 50.88 KB 434
195 188 23.94 KB 383
365 0 11.64 KB 365
123 237 47.94 KB 360
260 0 12.19 KB 260
An instruments panel will appear.

… and the app will run.



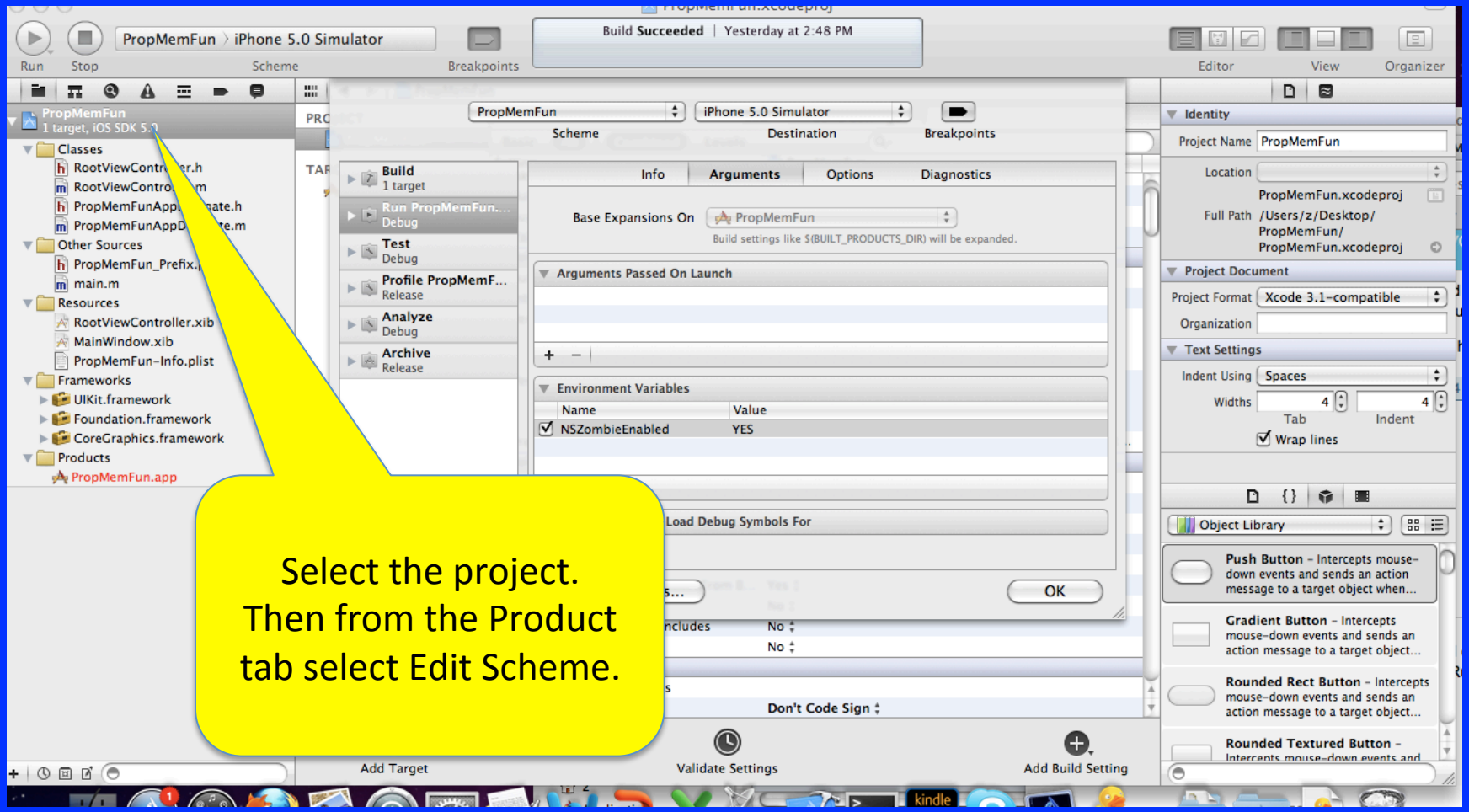
Click a few buttons until it crashes

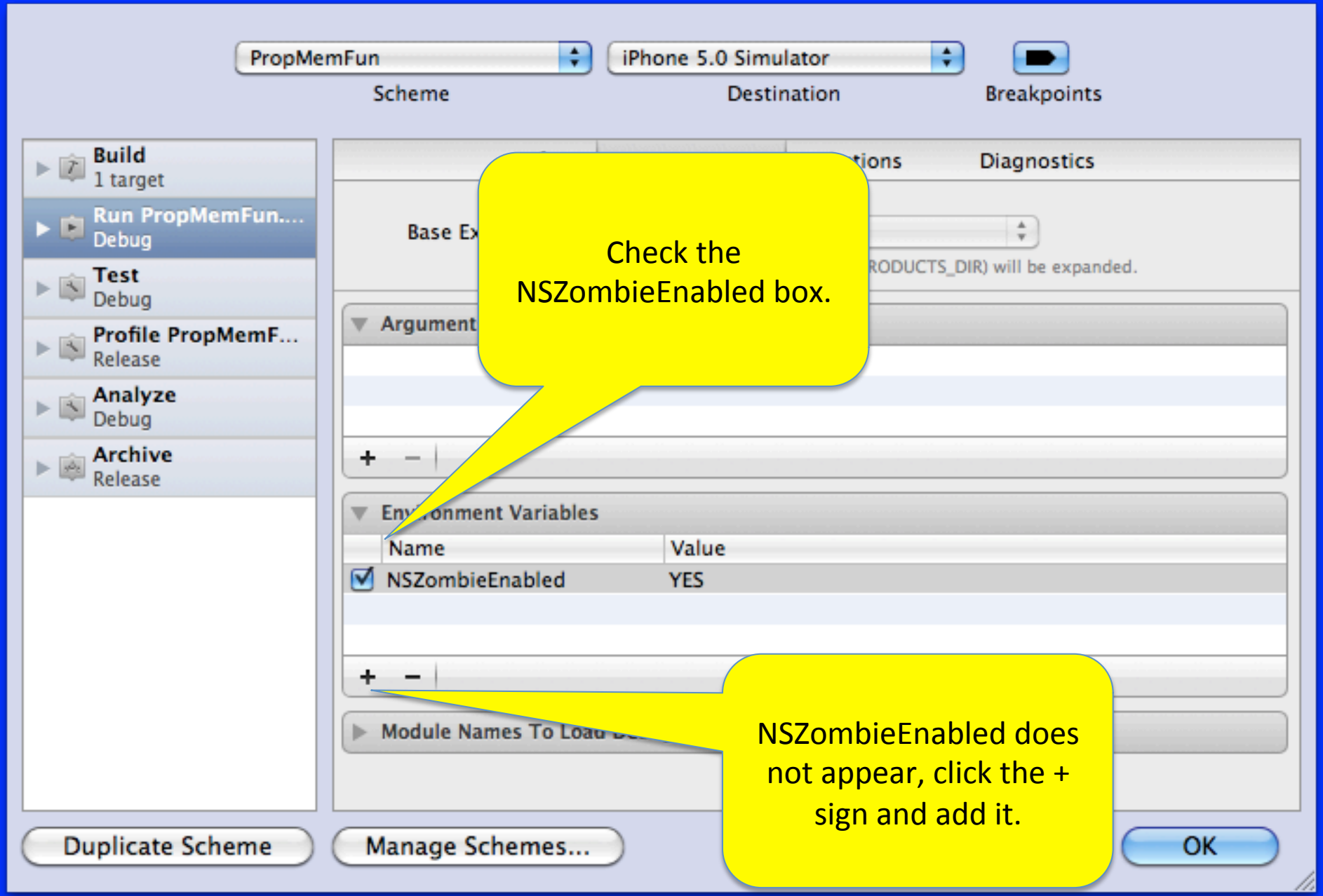

iOS Simulator File Edit Hardware Debug Window Help
Tue 6:57 AM
Instruments1
Record
PropMemFun
Target
Inspection Range
00:02:36
Run 2 of 2
View
Instrument Detail
Instruments
Allocations
Zombie Messaged
An Objective-C message was sent to a deallocated object (zombie) at address: 0x6b28490.
Done
Click here
The zombie was found.
Carrier
6:57 AM
HW5
Hypno
logicTests
Quiz
Simon Says
Safari
Call Tree
Separate by Category
Separate by Thread
Invert Call Tree
Hide Missing Symbols
Hide System Libraries
Show Obj-C Only
Flatten Recursion
Call Tree Constraints
Specific Data Mining
Live Bytes
Living
Transitory
Overall Bytes
Overall
Allocations (Net / Overall)
1.43 MB
20163
14510
4.00 MB
34673
86.98 KB
5567
2533
126.56 KB
8100
8.00 KB
64
4090
519.25 KB
4154
90.22 KB
2887
853
116.88 KB
3740
149.88 KB
3637
0
149.88 KB
3637
Malloc 64 Bytes
15.19 KB
243
1246
93.06 KB
1489
CFString (store)
46.91 KB
368
845
463.03 KB
1213
Malloc 80 Bytes
14.14 KB
181
960
89.14 KB
1141
Malloc 48 Bytes
24.33 KB
519
572
51.14 KB
1091
Malloc 8 Bytes
6.48 KB
830
257
8.49 KB
1087
CFBasicHash (value-st...
26.02 KB
310
680
69.48 KB
990
CFString (mutable)
24.72 KB
791
0
24.72 KB
791
CFBasicHash (key-store)
24.00 KB
261
368
61.86 KB
629
CFArray (immutable)
17.67 KB
512
0
17.67 KB
512
CFDictionary (mutable)
17.95 KB
383
0
17.95 KB
383
Malloc 96 Bytes
8.16 KB
87
222
28.97 KB
309
Malloc 304 Bytes
46.31 KB
156
146
89.66 KB
302
Malloc 1.00 KB
79.00 KB
79
213
292.00 KB
292
CFSet (mutable)
7.75 KB
248
0
7.75 KB
248
// Releases the view if it doesn't have a superview.
Push Button

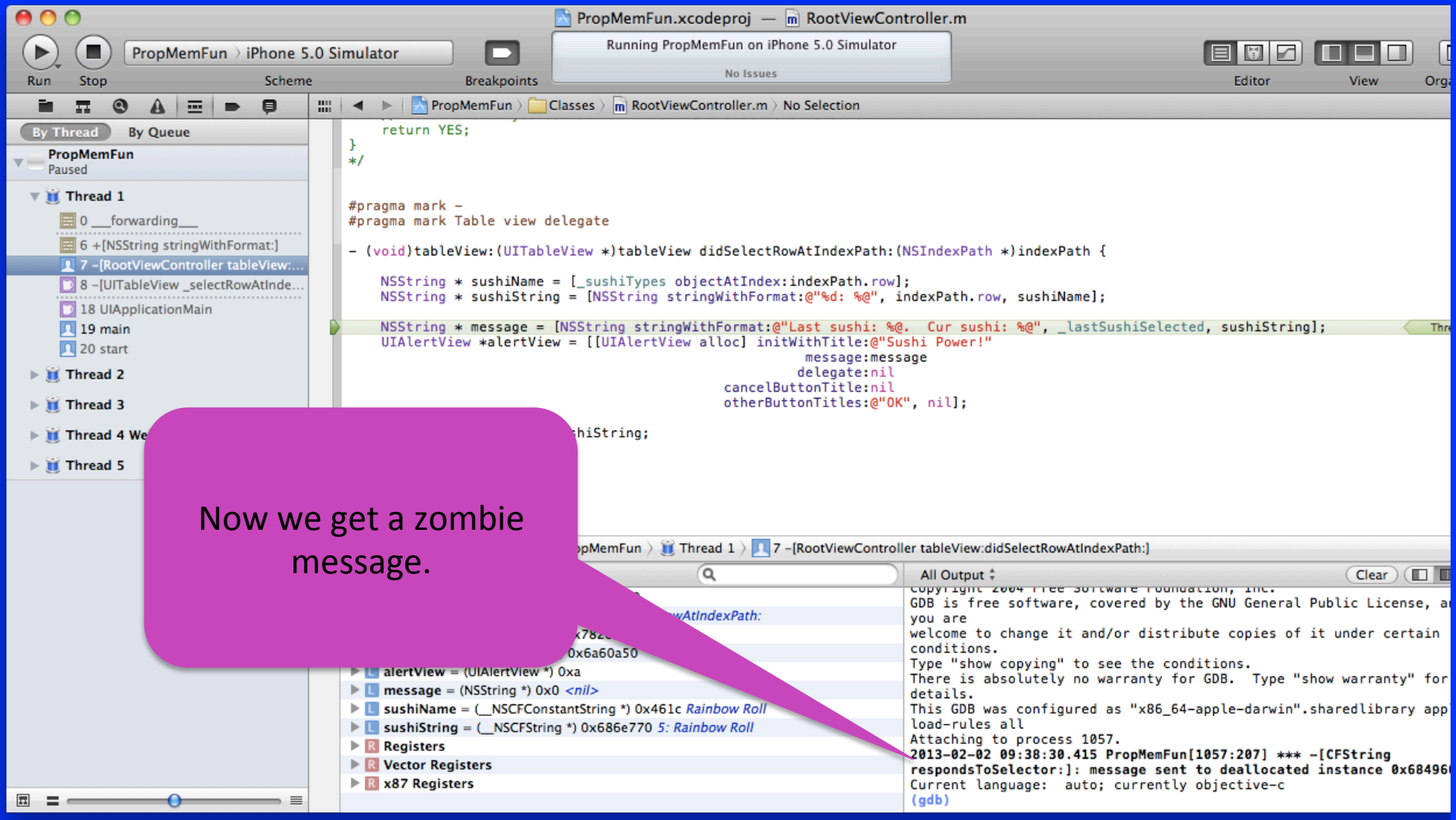

iOS Simulator File Edit Hardware Debug Window Help
Tue 6:58 AM
Instruments1
Record
PropMemFun
Target
Inspection Range
00:02:36
Run 2 of 2
View
Library
Instrument Detail
Search
Instruments
Allocations
Zombie Messaged
An Objective-C message was sent to a deallocated object (zombie) at address: 0x6b28490.
Done
Carrier
6:58 AM
HW5
Hypno
logicTests
passer
Quiz
Scrambler
Simon Says
Safari
Allocations
Statistics
Object Summary
History: 0x6b28490
Heapshot Analysis
Mark Heap
Allocation Lifespan
All Objects Created
Created & Still Living
Created & Destroyed
Call Tree
Separate by Category
Separate by Thread
Invert Call Tree
Hide Missing Symbols
Hide System Libraries
Show Obj-C Only
Flatten Recursion
Call Tree Constraints
Specific Data Mining
Address Category Event Type RefCt Timestamp Size Responsible Li... Responsible Caller
0 0x6b28490 CFString (immut... Malloc 1 00:08.540.738 32 Foundation -[NSPlaceholderString in...
1 0x6b28490 CFString (immut... Autorelease 00:08.540.760 0 Foundation +[NSString stringWithFo...
2 0x6b28490 CFString (immut... CFRetain 2 00:08.540.790 0 Foundation -[NSPlaceholderString in...
3 0x6b28490 CFString (immut... CFRelease 1 00:08.540.793 0 Foundation -[NSPlaceholderString in...
4 0x6b28490 CFString (immut... CFRelease 0 00:09.503.239 0 GraphicsServices GSEventRunModal
5 0x6b28490 CFString (immut... Zombie -1 00:15.462.902 0 Foundation -[NSPlaceholderString in...
Zombie!
// Releases the view if it doesn't have a superview.
Push Button

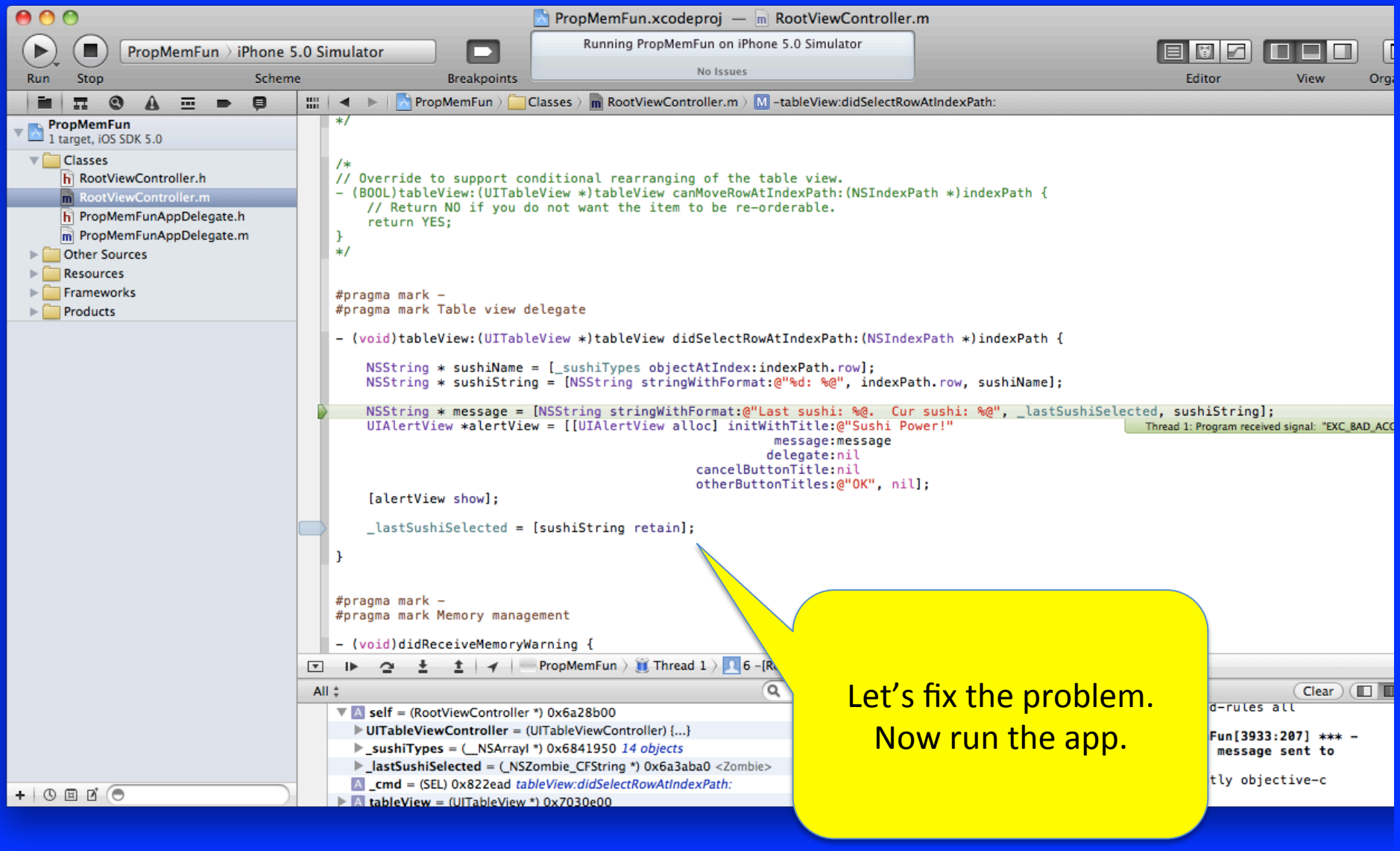
- We can fix it
- We can also tell xcode to watch out for zombies!

Watch out for Zombies!

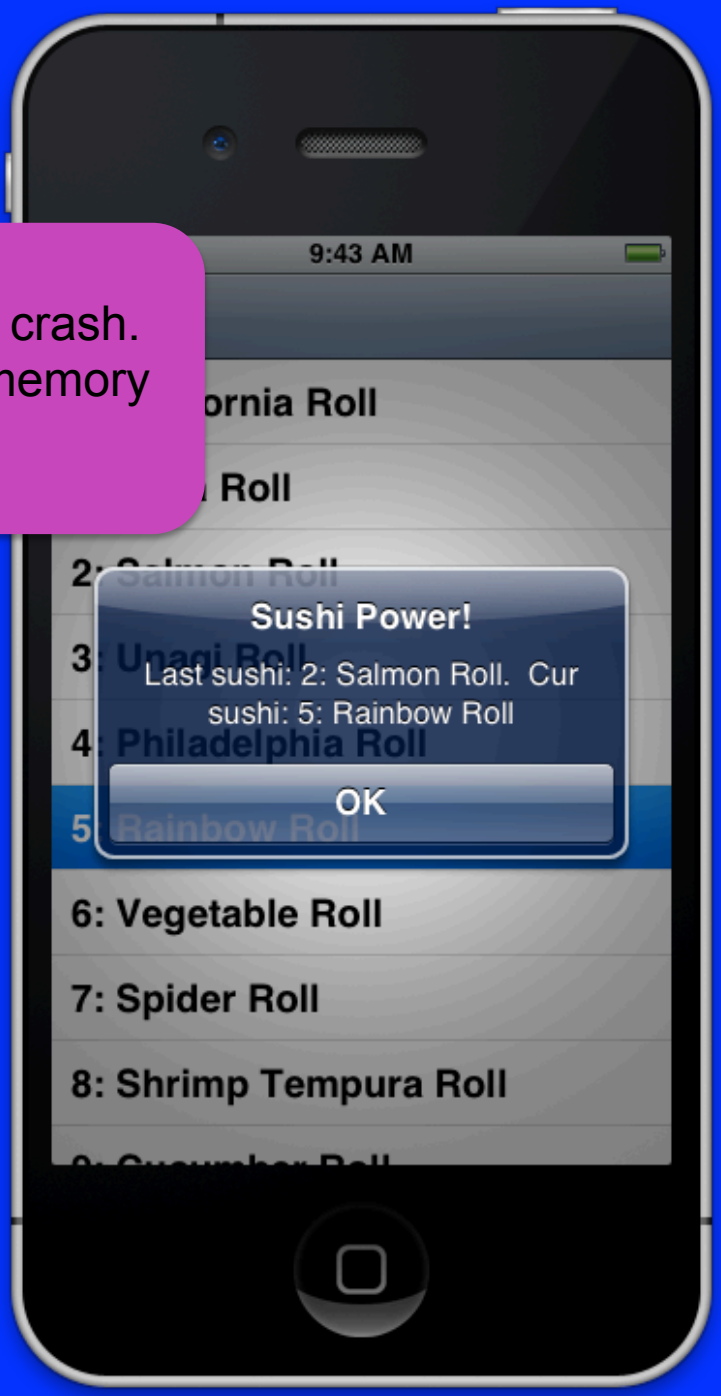


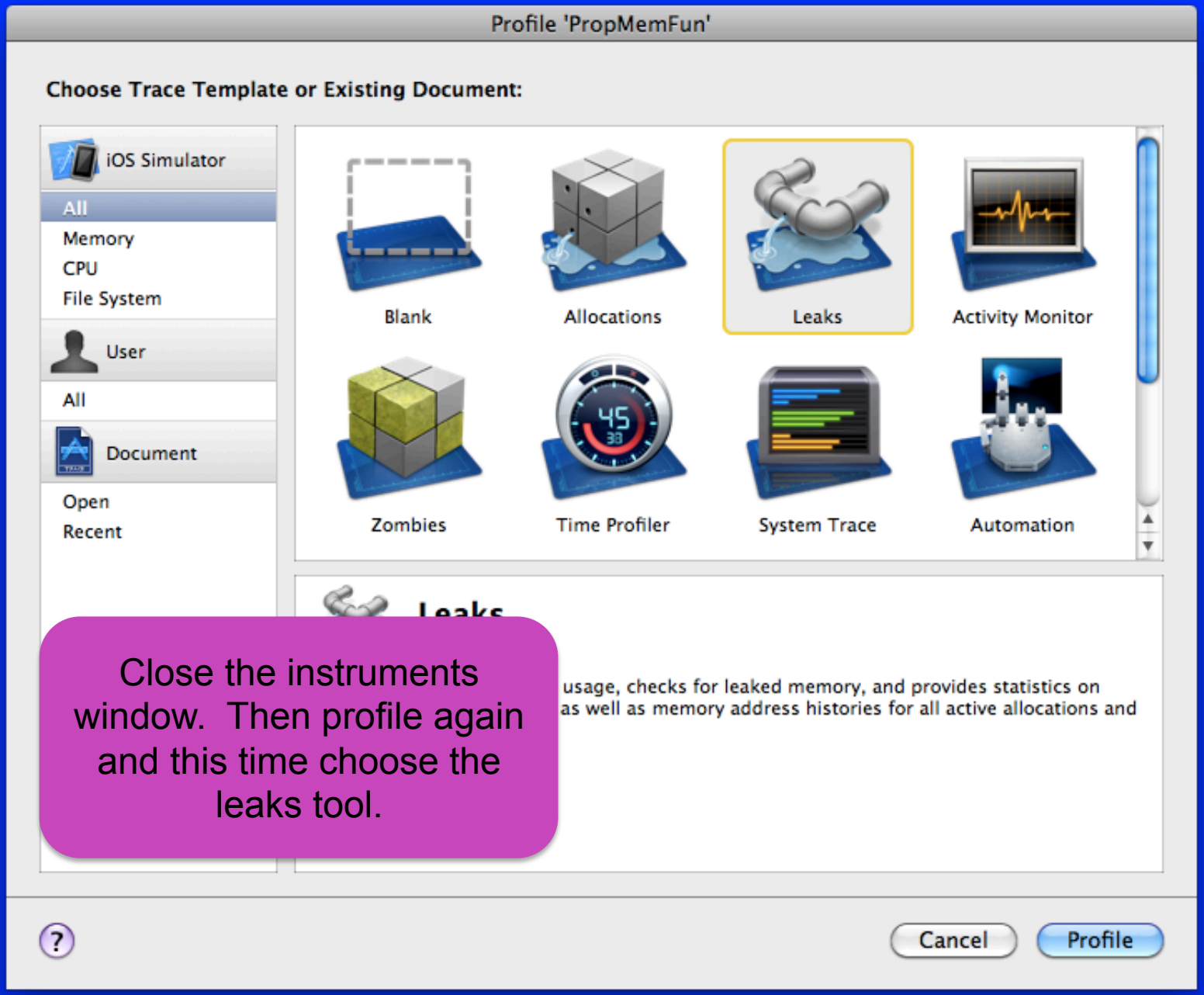

PropMemFun
Scheme
iPhone 5.0 Simulator
Destination
Breakpoints
Build
1 target
Run PropMemFun....
Debug
Test
Debug
Profile PropMemF...
Release
Analyze
Debug
Archive
Release
Diagnostics
Base Ex
RODUCTS_DIR) will be expanded.
Argument
Environment Variables
Name
Value
NSZombieEnabled
YES
Module Names To Load
Check the NSZombieEnabled box.
NSZombieEnabled does not appear, click the + sign and add it.
Duplicate Scheme
Manage Schemes...
OK

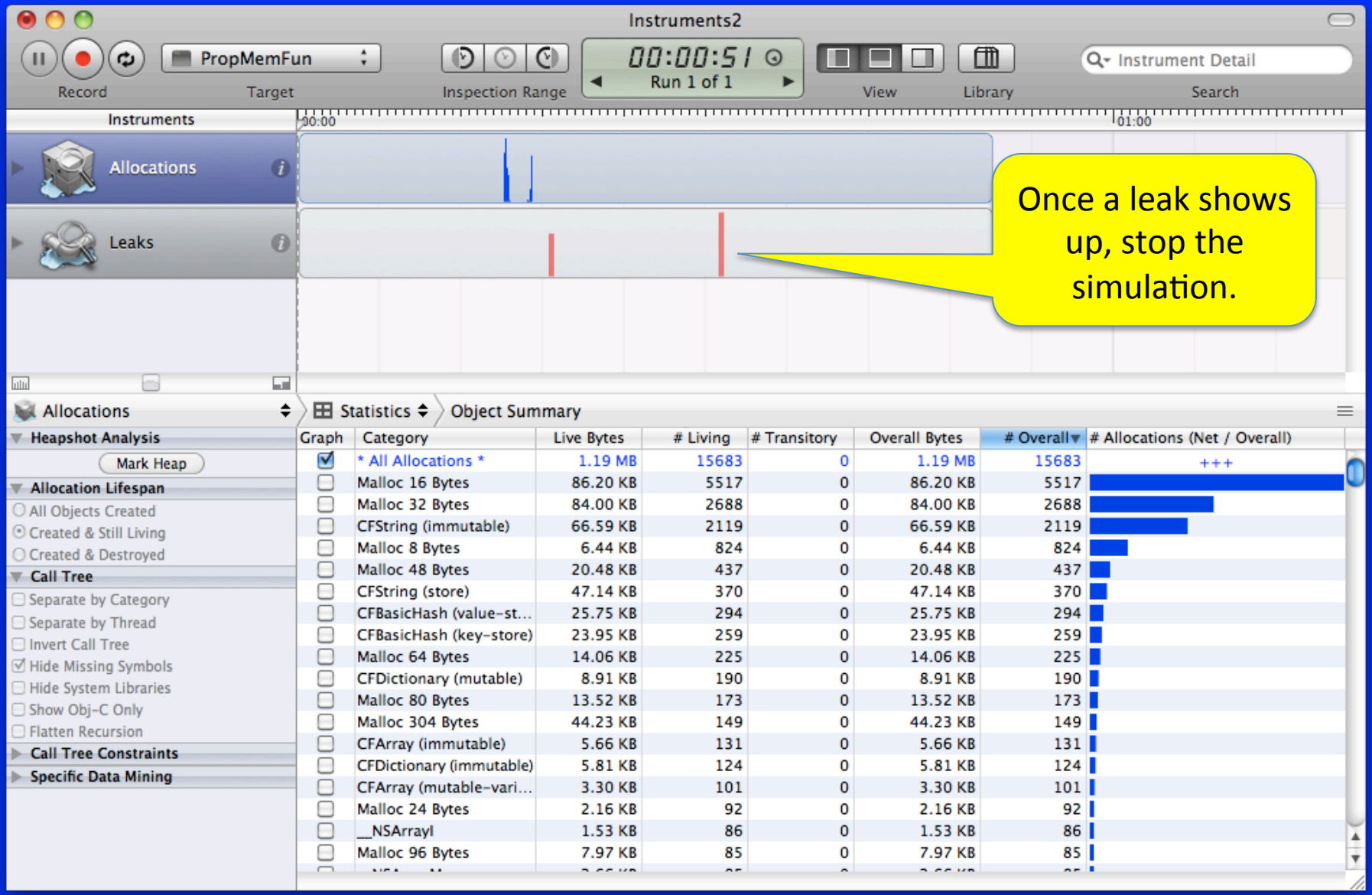

PropMemFun.xcodeproj — RootViewController.m
Run
Stop
PropMemFun › iPhone 5.0 Simulator
Scheme
Breakpoints
Running PropMemFun on iPhone 5.0 Simulator
No Issues
Editor
View
PropMemFun › Classes › RootViewController.m › No Selection
By Thread
By Queue
PropMemFun
Paused
Thread 1
0 __forwarding__
6 +[NSString stringWithFormat:]
7 -[RootViewController tableView:...
8 -[UITableView _selectRowAtInde...
18 UIApplicationMain
19 main
20 start
Thread 2
Thread 3
Thread 4
Thread 5
return YES;
}
*/
#pragma mark -
#pragma mark Table view delegate
- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath {
NSString * sushiName = [_sushiTypes objectAtIndex:indexPath.row];
NSString * sushiString = [NSString stringWithFormat:@"%d: %@", indexPath.row, sushiName];
NSString * message = [NSString stringWithFormat:@"Last sushi: %@. Cur sushi: %@", _lastSushiSelected, sushiString];
UIAlertView *alertView = [[UIAlertView alloc] initWithTitle:@"Sushi Power!"
message:message
delegate:nil
cancelButtonTitle:nil
otherButtonTitles:@"OK", nil];
hiString;
Now we get a zombie message.
pMemFun › Thread 1 › 7 -[RootViewController tableView:didSelectRowAtIndexPath:]
wAtIndexPath:
0x6a60a50
alertView = (UIAlertView *) 0xa
message = (NSString *) 0x0 <nil>
sushiName = (__NSCFConstantString *) 0x461c Rainbow Roll
sushiString = (__NSCFString *) 0x686e770 5: Rainbow Roll
Registers
Vector Registers
x87 Registers
All Output
Clear
GDB is free software, covered by the GNU General Public License, a
you are
welcome to change it and/or distribute copies of it under certain
conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for
details.
This GDB was configured as "x86_64-apple-darwin".sharedlibrary app
load-rules all
Attaching to process 1057.
2013-02-02 09:38:30.415 PropMemFun[1057:207] *** -[CFString
respondsToSelector:]: message sent to deallocated instance 0x68496
Current language: auto; currently objective-c
(gdb)

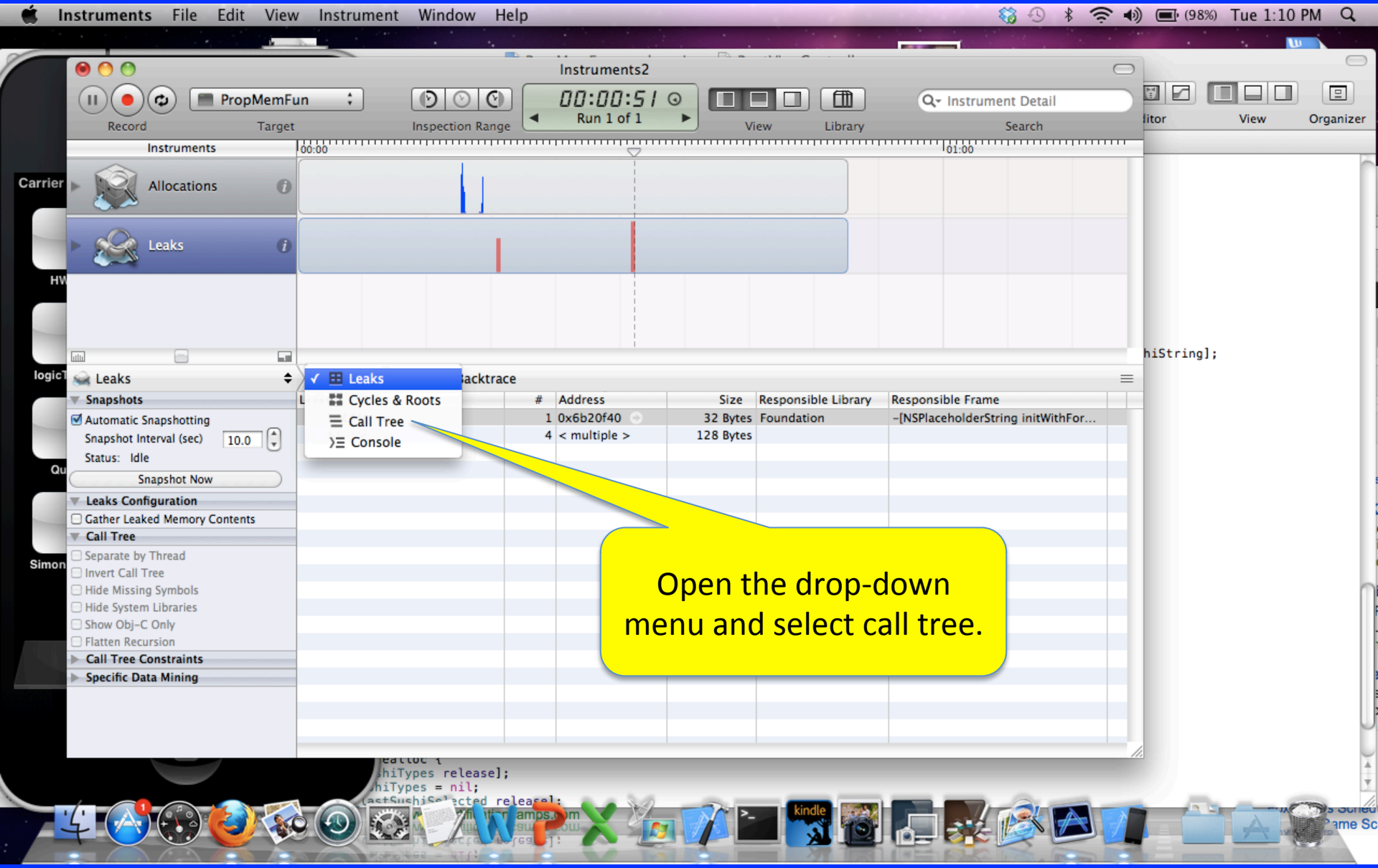

PropMemFun.xcodeproj — RootViewController.m
Run
Stop
PropMemFun › iPhone 5.0 Simulator
Scheme
Breakpoints
Running PropMemFun on iPhone 5.0 Simulator
No Issues
Editor
View
PropMemFun
1 target, iOS SDK 5.0
Classes
RootViewController.h
RootViewController.m
PropMemFunAppDelegate.h
PropMemFunAppDelegate.m
Other Sources
Resources
Frameworks
Products
PropMemFun › Classes › RootViewController.m › -tableView:didSelectRowAtIndexPath:
*/
/*
// Override to support conditional rearranging of the table view.
- (BOOL)tableView:(UITableView *)tableView canMoveRowAtIndexPath:(NSIndexPath *)indexPath {
// Return NO if you do not want the item to be re-orderable.
return YES;
}
*/
#pragma mark -
#pragma mark Table view delegate
- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath {
NSString * sushiName = [_sushiTypes objectAtIndex:indexPath.row];
NSString * sushiString = [NSString stringWithFormat:@"%d: %@", indexPath.row, sushiName];
NSString * message = [NSString stringWithFormat:@"Last sushi: %@. Cur sushi: %@", _lastSushiSelected, sushiString];
UIAlertView *alertView = [[UIAlertView alloc] initWithTitle:@"Sushi Power!"
message:message
delegate:nil
cancelButtonTitle:nil
otherButtonTitles:@"OK", nil];
[alertView show];
_lastSushiSelected = [sushiString retain];
}
#pragma mark -
#pragma mark Memory management
- (void)didReceiveMemoryWarning {
Thread 1: Program received signal: "EXC_BAD_ACC
PropMemFun › Thread 1 › 6 -[R
All
Clear
self = (RootViewController *) 0x6a28b00
UITableViewController = (UITableViewController) {...}
_sushiTypes = (__NSArrayI *) 0x6841950 14 objects
_lastSushiSelected = (_NSZombie_CFString *) 0x6a3aba0 <Zombie>
_cmd = (SEL) 0x822ead tableView:didSelectRowAtIndexPath:
tableView = (UITableView *) 0x7030e00
d-rules all
Fun[3933:207] *** -
message sent to
tly objective-c
Let’s fix the problem.
Now run the app.

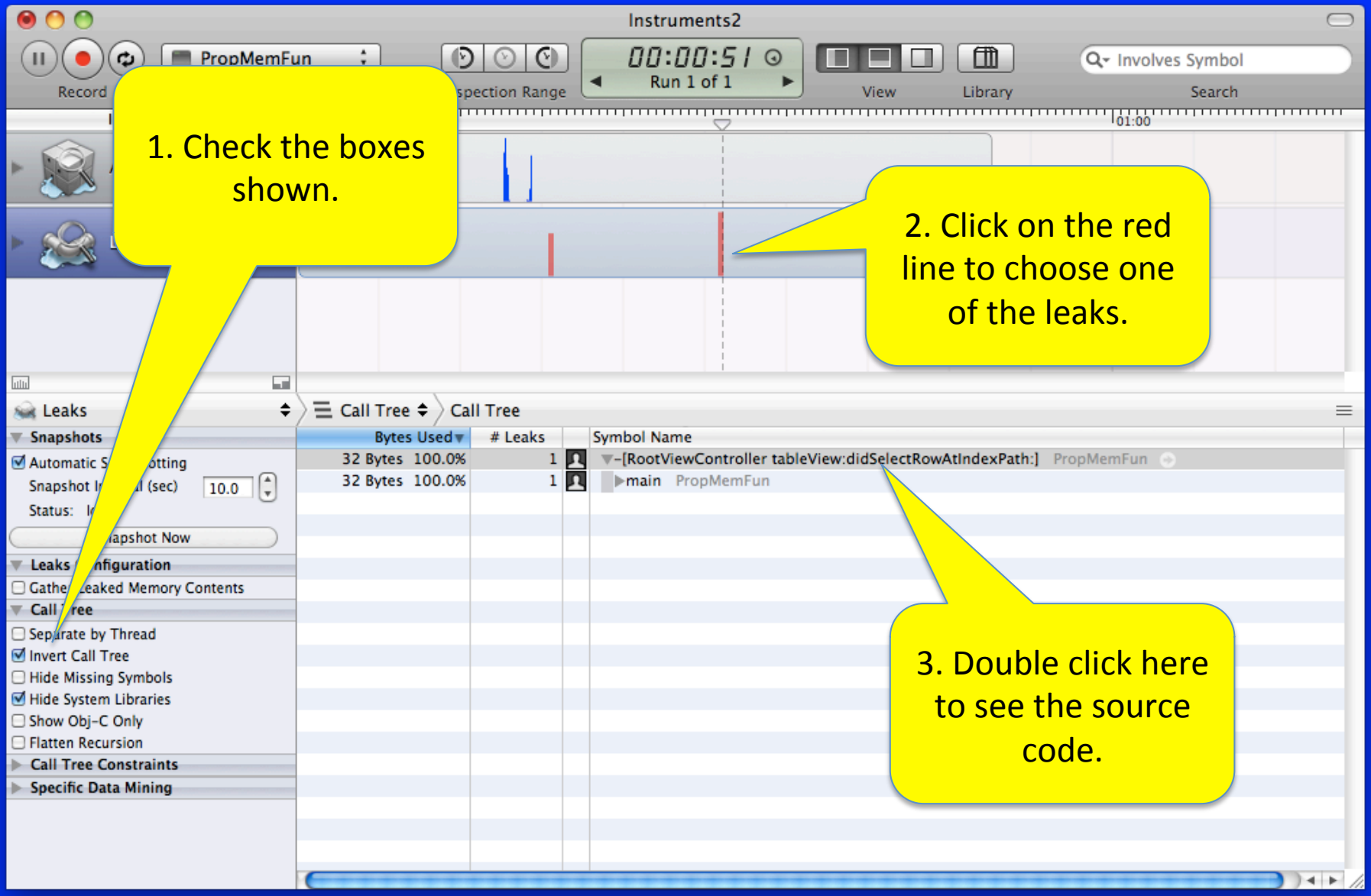
It should no longer crash. But we do have a memory leak.

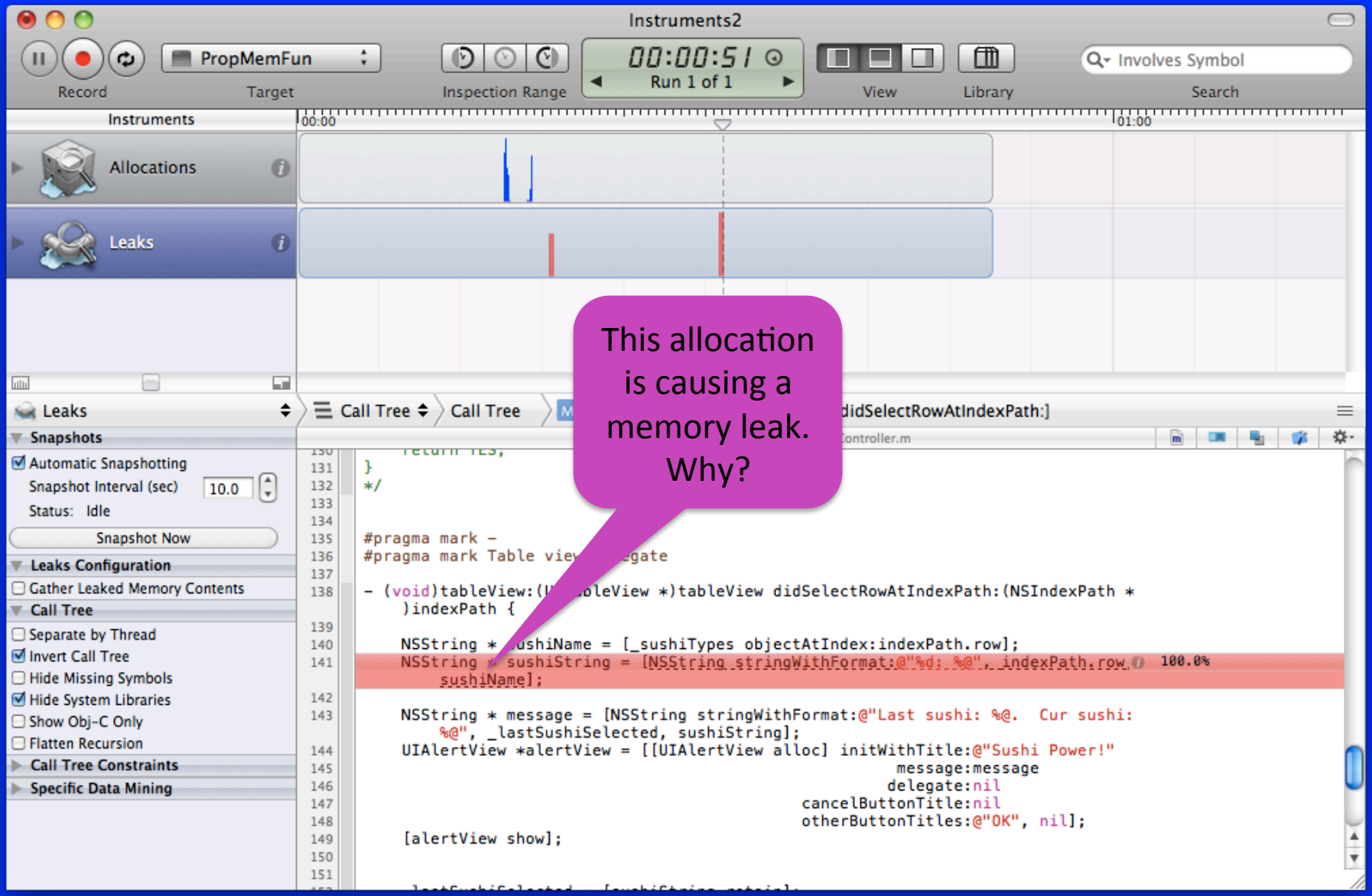


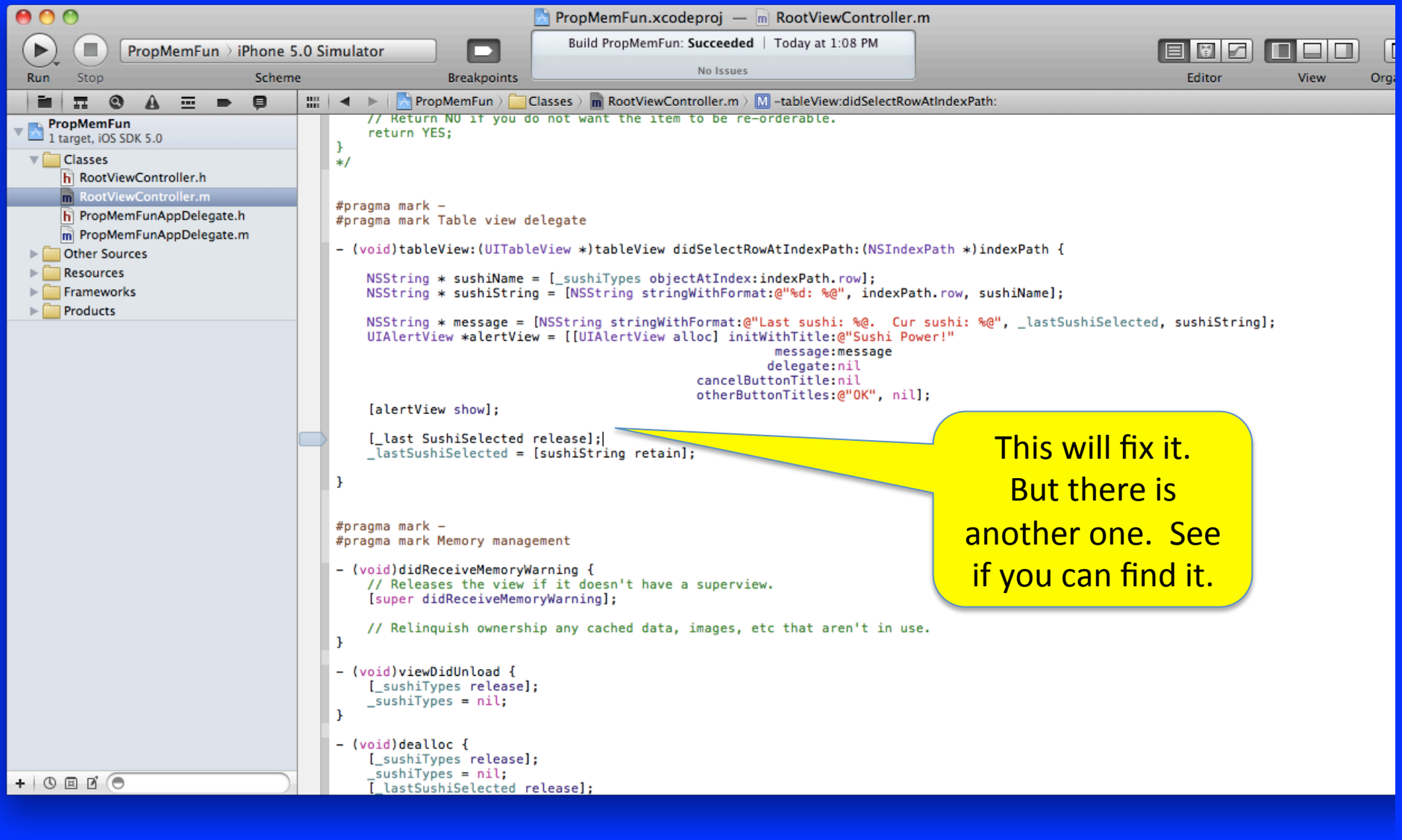

Profile 'PropMemFun'
Choose Trace Template or Existing Document:
iOS Simulator
All
Memory
CPU
File System
User
All
Document
Open
Recent
Blank
Allocations
Leaks
Activity Monitor
Zombies
Time Profiler
System Trace
Automation
usage, checks for leaked memory, and provides statistics on
as well as memory address histories for all active allocations and
Close the instruments window. Then profile again and this time choose the leaks tool.
Cancel
Profile


Instruments2
Record
PropMemFun
Target
Inspection Range
00:00:51
Run 1 of 1
View
Library
Instrument Detail
Search
Instruments
00:00
01:00
Allocations
Leaks
Once a leak shows up, stop the simulation.
Allocations
Statistics
Object Summary
Heapshot Analysis
Mark Heap
Allocation Lifespan
All Objects Created
Created & Still Living
Created & Destroyed
Call Tree
Separate by Category
Separate by Thread
Invert Call Tree
Hide Missing Symbols
Hide System Libraries
Show Obj-C Only
Flatten Recursion
Call Tree Constraints
Specific Data Mining
Graph
Category
Live Bytes
Living
Transitory
Overall Bytes
Overall
Allocations (Net / Overall)
* All Allocations * 1.19 MB 15683 0 1.19 MB 15683 +++
Malloc 16 Bytes 86.20 KB 5517 0 86.20 KB 5517
Malloc 32 Bytes 84.00 KB 2688 0 84.00 KB 2688
CFString (immutable) 66.59 KB 2119 0 66.59 KB 2119
Malloc 8 Bytes 6.44 KB 824 0 6.44 KB 824
Malloc 48 Bytes 20.48 KB 437 0 20.48 KB 437
CFString (store) 47.14 KB 370 0 47.14 KB 370
CFBasicHash (value-st... 25.75 KB 294 0 25.75 KB 294
CFBasicHash (key-store) 23.95 KB 259 0 23.95 KB 259
Malloc 64 Bytes 14.06 KB 225 0 14.06 KB 225
CFDictionary (mutable) 8.91 KB 190 0 8.91 KB 190
Malloc 80 Bytes 13.52 KB 173 0 13.52 KB 173
Malloc 304 Bytes 44.23 KB 149 0 44.23 KB 149
CFArray (immutable) 5.66 KB 131 0 5.66 KB 131
CFDictionary (immutable) 5.81 KB 124 0 5.81 KB 124
CFArray (mutable-vari... 3.30 KB 101 0 3.30 KB 101
Malloc 24 Bytes 2.16 KB 92 0 2.16 KB 92
__NSArrayI 1.53 KB 86 0 1.53 KB 86
Malloc 96 Bytes 7.97 KB 85 0 7.97 KB 85


Instruments File Edit View Instrument Window Help
Tue 1:10 PM
Instruments2
Record
PropMemFun
Target
Inspection Range
00:00:51
Run 1 of 1
View
Library
Instrument Detail
Search
Organizer
Instruments
Allocations
Leaks
Leaks
Cycles & Roots
Call Tree
Console
Snapshots
Automatic Snapshotting
Snapshot Interval (sec)
10.0
Status: Idle
Snapshot Now
Leaks Configuration
Gather Leaked Memory Contents
Call Tree
Separate by Thread
Invert Call Tree
Hide Missing Symbols
Hide System Libraries
Show Obj-C Only
Flatten Recursion
Call Tree Constraints
Specific Data Mining
Address Size Responsible Library Responsible Frame
1 0x6b20f40 32 Bytes Foundation -[NSPlaceholderString initWithFor...
4 < multiple > 128 Bytes
Open the drop-down menu and select call tree.


Instruments2
PropMemFun
Record
00:00:51
Run 1 of 1
View
Library
Involves Symbol
Search
01:00
1. Check the boxes shown.
2. Click on the red line to choose one of the leaks.
Leaks
Call Tree
Call Tree
Snapshots
Automatic S otting
Snapshot I l (sec)
10.0
Status:
napshot Now
Leaks nfiguration
Gathe eaked Memory Contents
Call ree
Separate by Thread
Invert Call Tree
Hide Missing Symbols
Hide System Libraries
Show Obj-C Only
Flatten Recursion
Call Tree Constraints
Specific Data Mining
Bytes Used
Leaks
Symbol Name
32 Bytes 100.0% 1 -[RootViewController tableView:didSelectRowAtIndexPath:] PropMemFun
32 Bytes 100.0% 1 main PropMemFun
3. Double click here to see the source code.


Instruments2
Record
PropMemFun
Target
Inspection Range
00:00:51
Run 1 of 1
View
Library
Involves Symbol
Search
Instruments
00:00
01:00
Allocations
Leaks
This allocation is causing a memory leak. Why?
Leaks
Call Tree
Call Tree
didSelectRowAtIndexPath:]
Snapshots
Automatic Snapshotting
Snapshot Interval (sec)
10.0
Status: Idle
Snapshot Now
Leaks Configuration
Gather Leaked Memory Contents
Call Tree
Separate by Thread
Invert Call Tree
Hide Missing Symbols
Hide System Libraries
Show Obj-C Only
Flatten Recursion
Call Tree Constraints
Specific Data Mining
100.0%
}
*/
#pragma mark -
#pragma mark Table view delegate
- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *
)indexPath {
NSString * sushiName = [_sushiTypes objectAtIndex:indexPath.row];
NSString * sushiString = [NSString stringWithFormat:@"%d: %@", indexPath.row,
sushiName];
NSString * message = [NSString stringWithFormat:@"Last sushi: %@. Cur sushi:
%@", _lastSushiSelected, sushiString];
UIAlertView *alertView = [[UIAlertView alloc] initWithTitle:@"Sushi Power!"
message:message
delegate:nil
cancelButtonTitle:nil
otherButtonTitles:@"OK", nil];
[alertView show];


PropMemFun.xcodeproj — RootViewController.m
Build PropMemFun: Succeeded | Today at 1:08 PM
No Issues
Run
Stop
PropMemFun › iPhone 5.0 Simulator
Scheme
Breakpoints
Editor
View
PropMemFun › Classes › RootViewController.m › -tableView:didSelectRowAtIndexPath:
PropMemFun
1 target, iOS SDK 5.0
Classes
RootViewController.h
RootViewController.m
PropMemFunAppDelegate.h
PropMemFunAppDelegate.m
Other Sources
Resources
Frameworks
Products
// Return NO if you do not want the item to be re-orderable.
return YES;
}
*/
#pragma mark -
#pragma mark Table view delegate
- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath {
NSString * sushiName = [_sushiTypes objectAtIndex:indexPath.row];
NSString * sushiString = [NSString stringWithFormat:@"%d: %@", indexPath.row, sushiName];
NSString * message = [NSString stringWithFormat:@"Last sushi: %@. Cur sushi: %@", _lastSushiSelected, sushiString];
UIAlertView *alertView = [[UIAlertView alloc] initWithTitle:@"Sushi Power!"
message:message
delegate:nil
cancelButtonTitle:nil
otherButtonTitles:@"OK", nil];
[alertView show];
[_last SushiSelected release];
_lastSushiSelected = [sushiString retain];
}
#pragma mark -
#pragma mark Memory management
- (void)didReceiveMemoryWarning {
// Releases the view if it doesn't have a superview.
[super didReceiveMemoryWarning];
// Relinquish ownership any cached data, images, etc that aren't in use.
}
- (void)viewDidUnload {
[_sushiTypes release];
_sushiTypes = nil;
}
- (void)dealloc {
[_sushiTypes release];
_sushiTypes = nil;
[_lastSushiSelected release];
This will fix it. But there is another one. See if you can find it.