Test Driven Development in the iOS World Part 1

Doug Sjoquist
http://www.sunetos.com
@dwsjoquist

What Is It?

Using automated software tests to drive the design and development of an application iteratively by writing your tests **BEFORE** you write your code

Why?

WHAT'S IN IT FOR ME?

If our target is to produce professional level work

If our target is to produce professional level work

Then to hit our target...

We
MUST
test our work

Automated testing is the best choice for many kinds of tests

Code must be designed with automated testing in mind

TDD ensures our code is testable

TDD helps us to only write code we need

The hardest part is

not

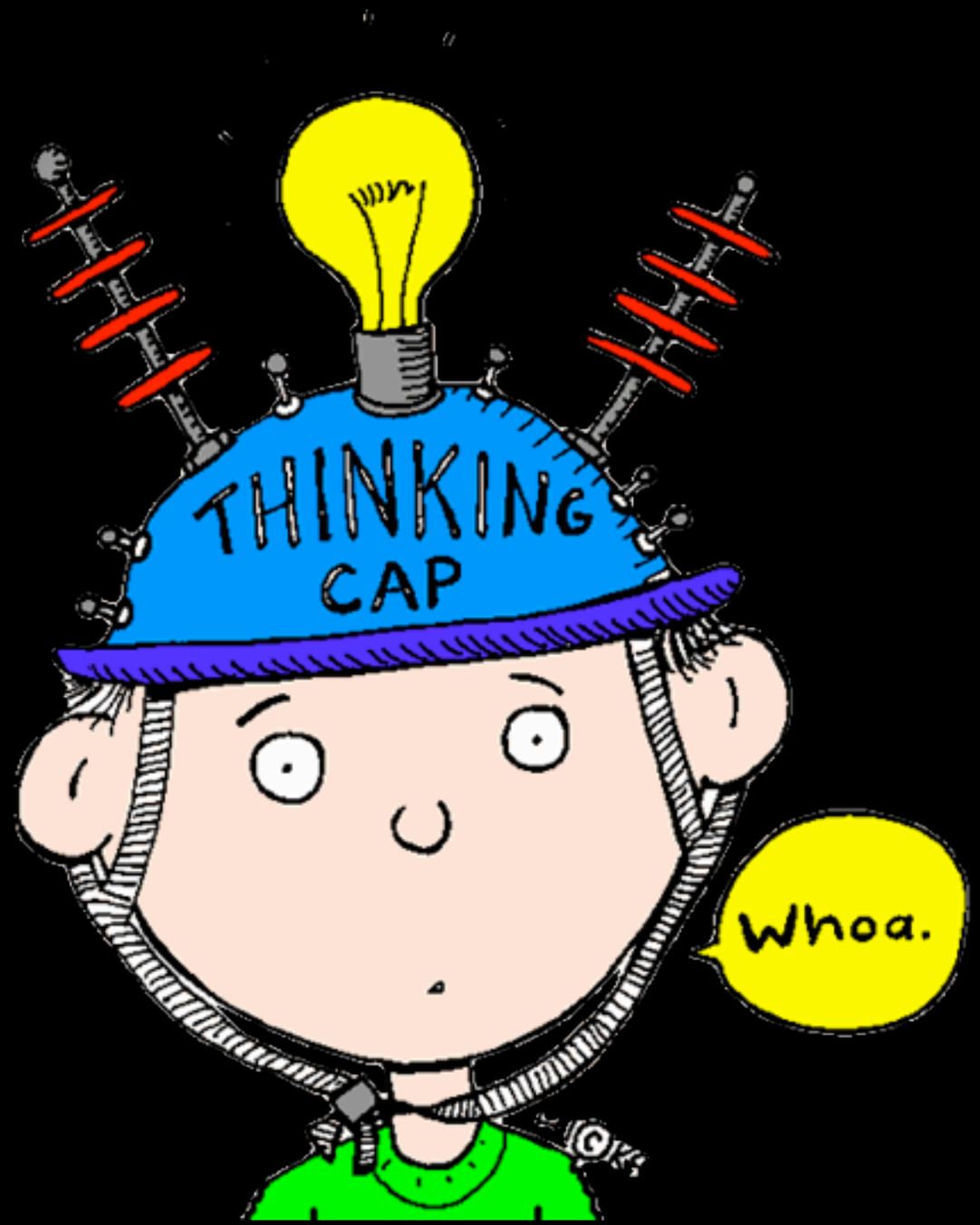
writing automated tests

The hardest part is

not

writing your tests first

The hardest part is deciding ***what*** to test and ***how*** to test it



Automated testing is ***THE*** key to ensure you will test your code often enough to make a difference

TDD is ***THE*** key to ensuring your code is testable

The Bottom Line

Why TDD?

It greatly enhances my ability to do professional level work

So... How?

Continuum of Components to be Tested

Continuum of Components to be Tested

- No dependencies

Continuum of Components to be Tested

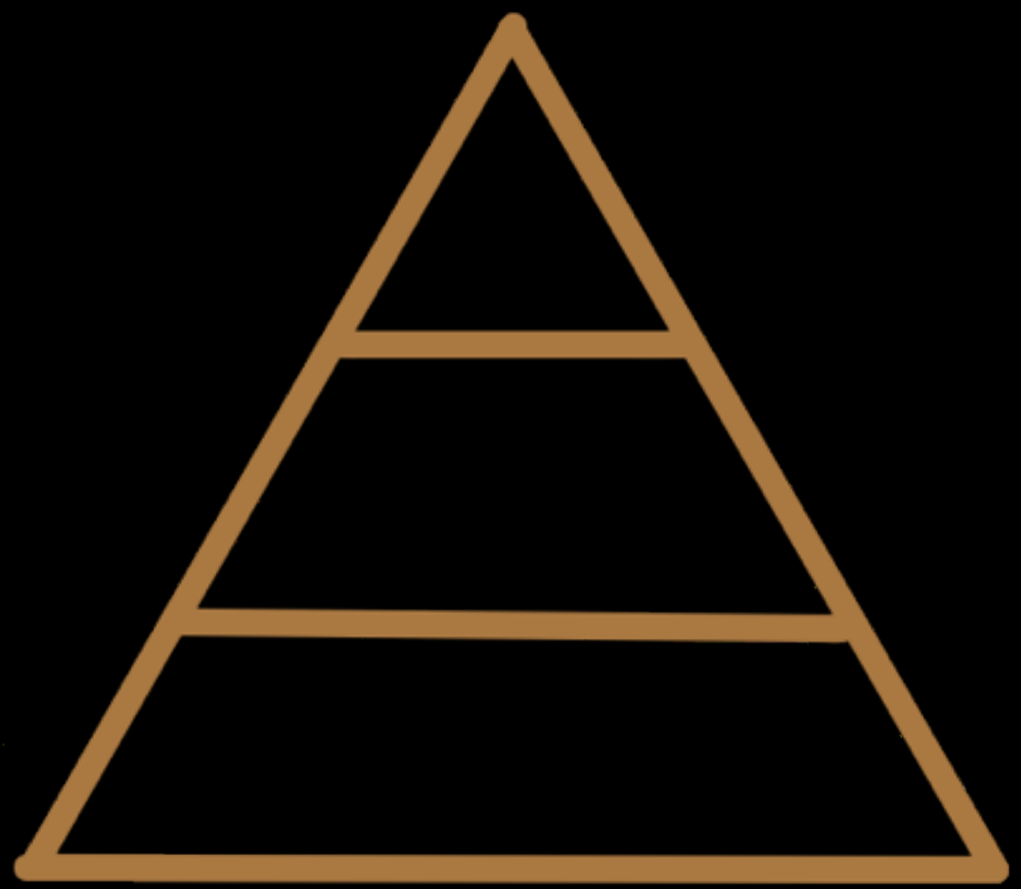
- No dependencies
- Internal dependencies

Continuum of Components to be Tested

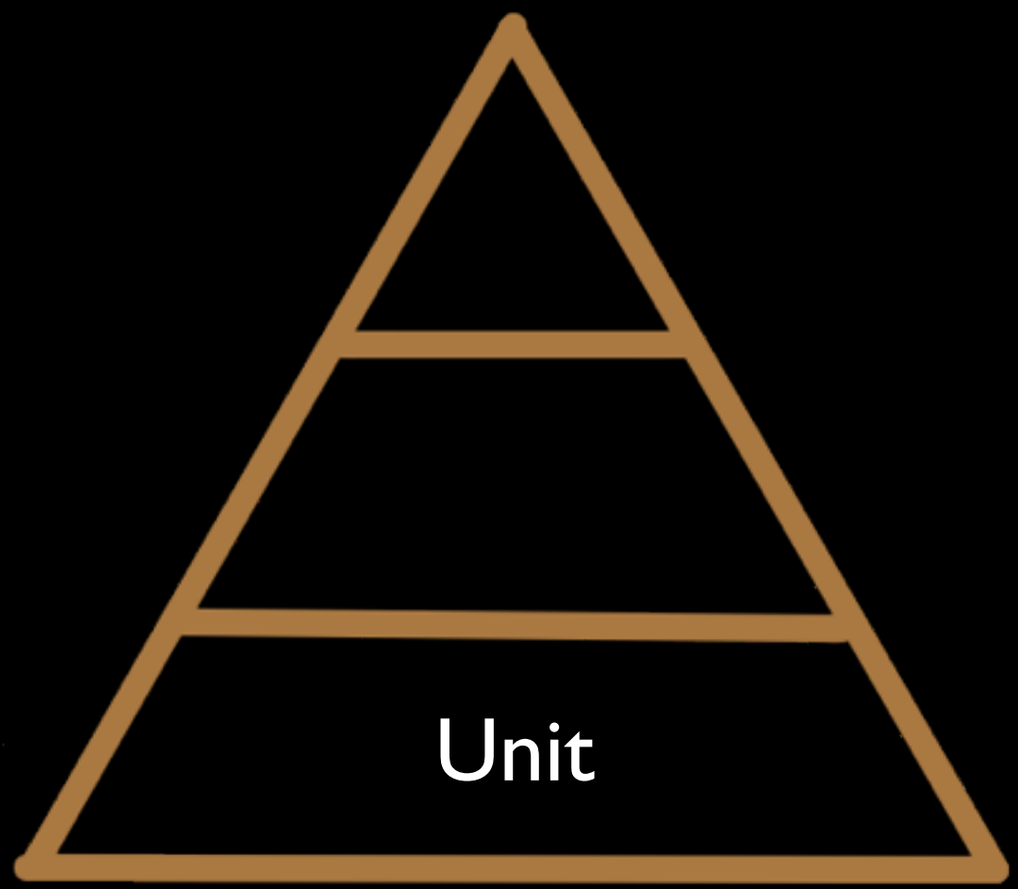
- No dependencies
- Internal dependencies
- External dependencies

Continuum of Testing Types

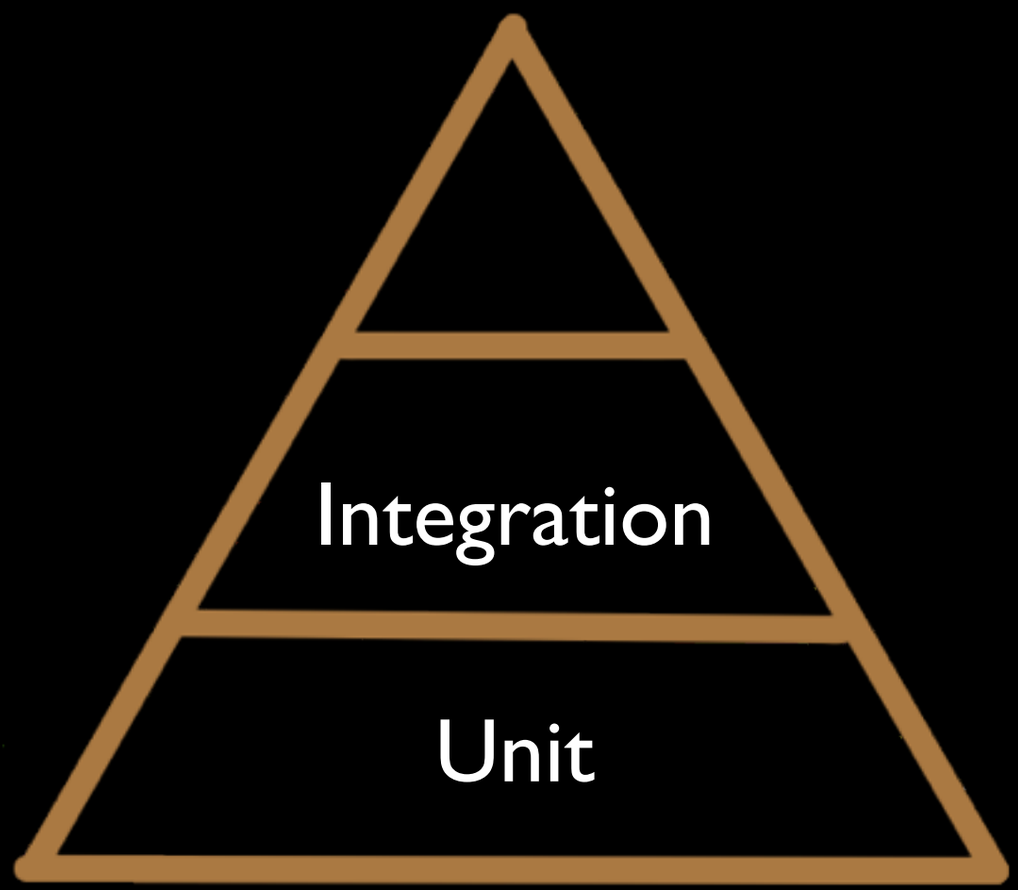
Continuum of Testing Types



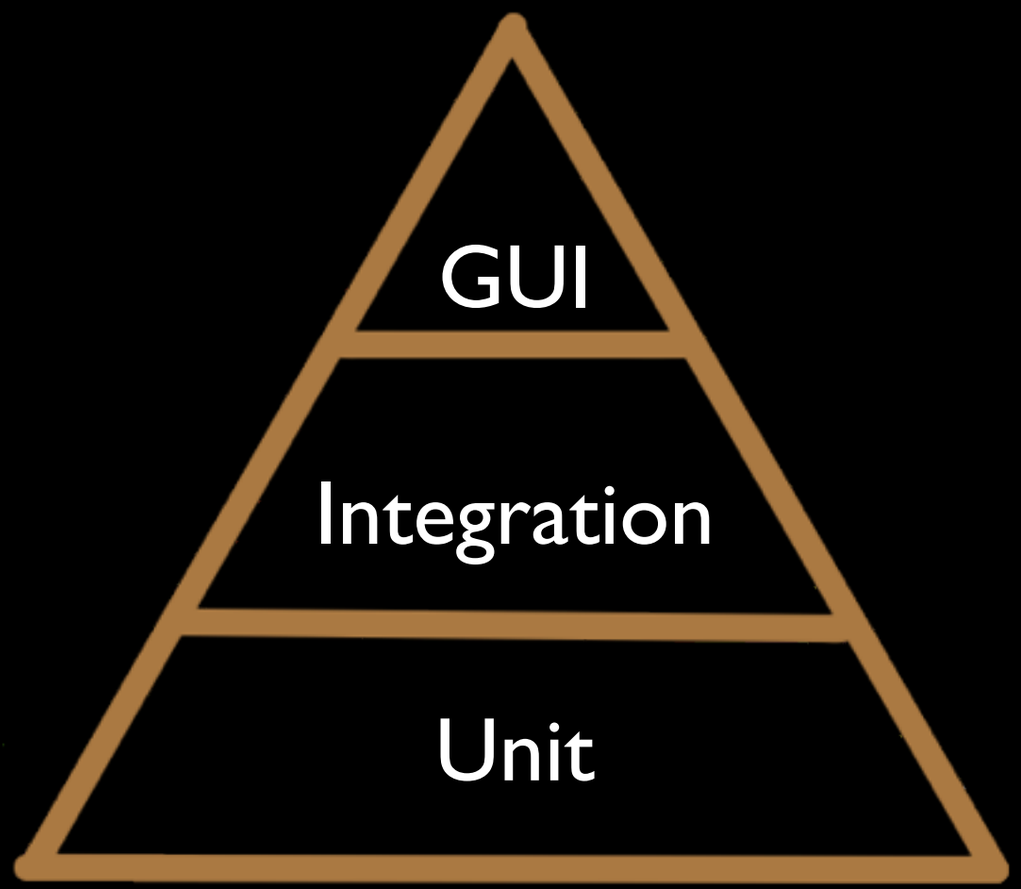
Continuum of Testing Types



Continuum of Testing Types

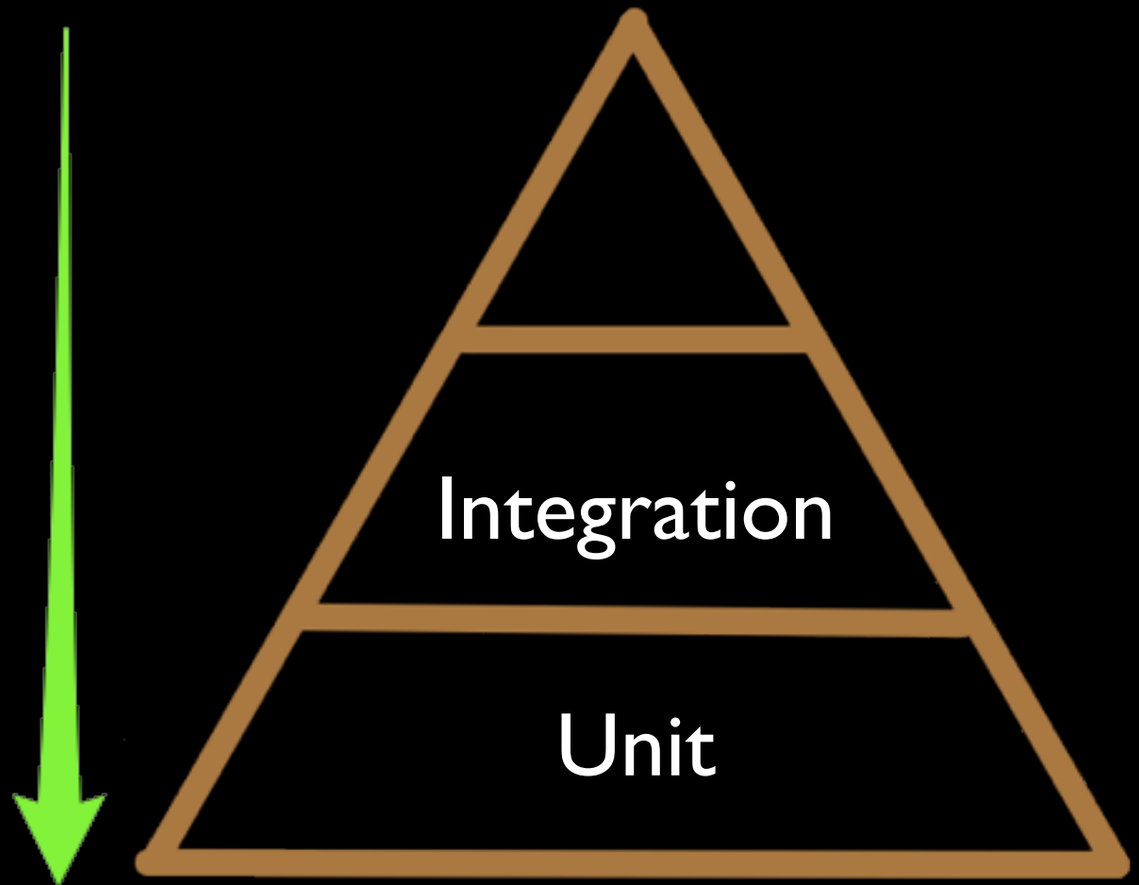


Continuum of Testing Types



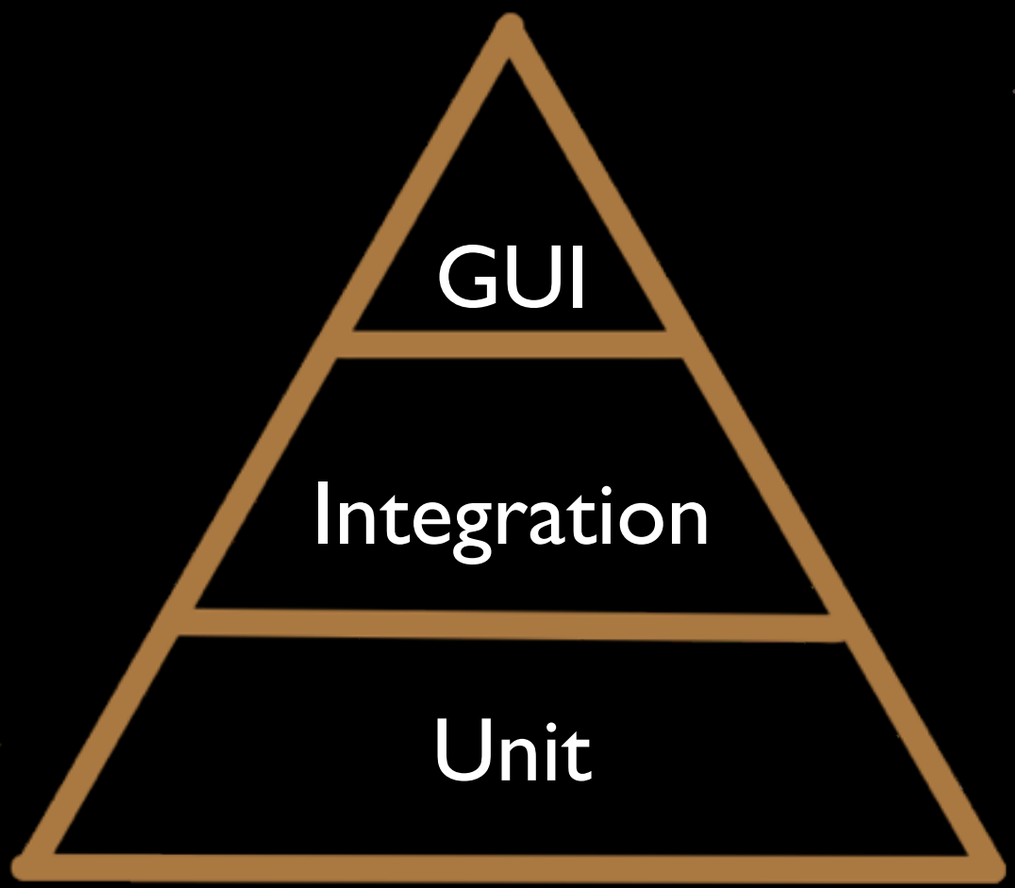
Continuum of Testing Types

Frequency
Ease of use
Immediacy



Continuum of Testing Types

Frequency
Ease of use
Immediacy



Brittleness
Complexity
Duration
Intrusiveness

Our Tools

Our Tools

Our Tools

GHUnit

Our Tools

GHUnit

OCMock

TDD Process

TDD Process

“Just Enough” Design

TDD Process

“Just Enough” Design

Development Iterations

TDD Process

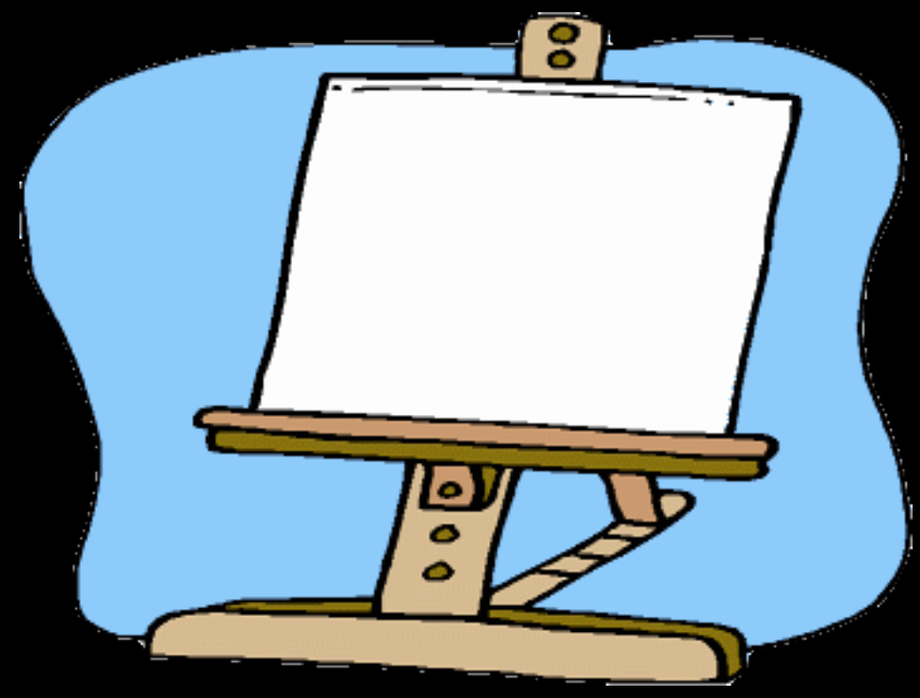
"Just Enough" Design

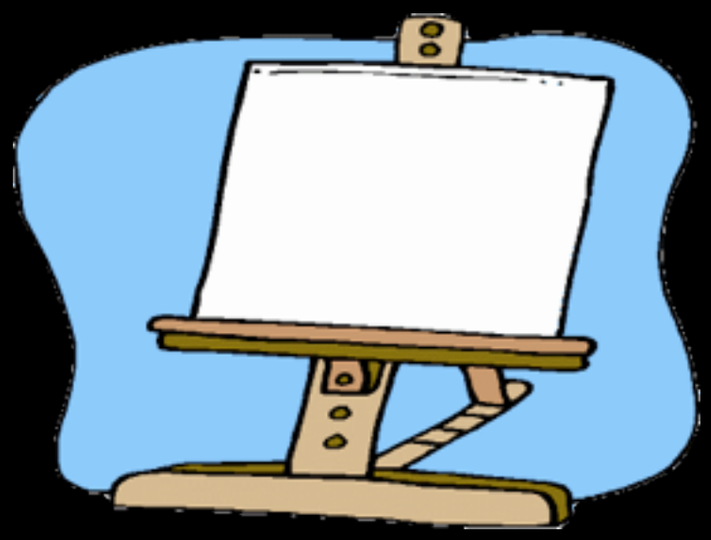
Development Iterations

Review & Refactoring

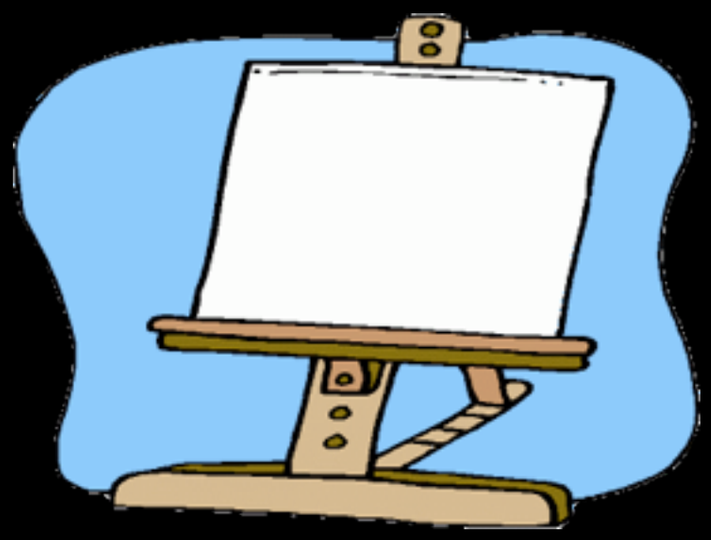
TDD Process

“Just Enough” Design

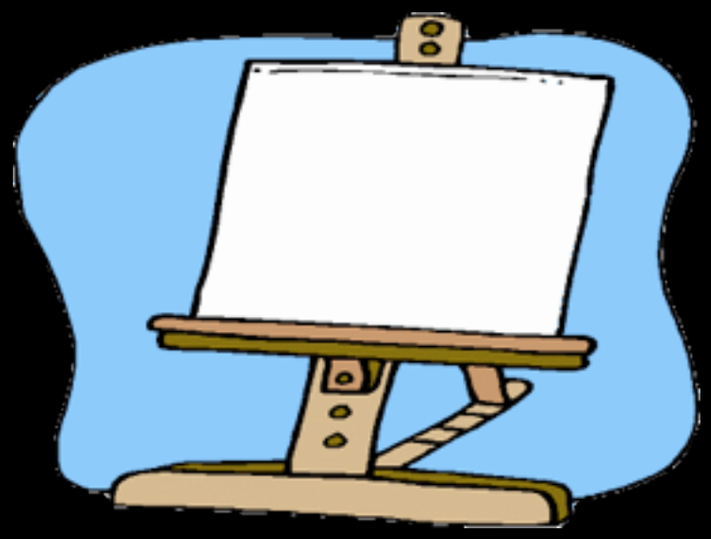




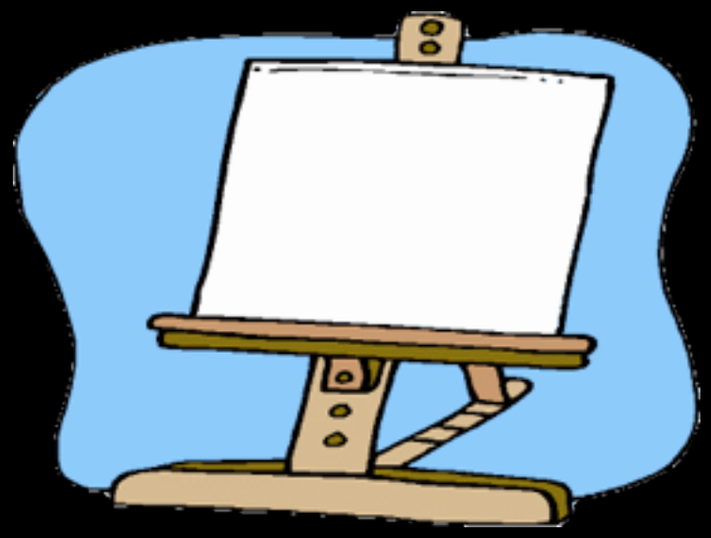
There is no single answer to how much is “Just Enough”



Decide: What is the purpose of this app?



Determine what constitutes “Done”



Do just enough up front design to get started

TDD Process

Development Iterations

Decide what behavior we want to add next

Think about a good way to test it

Write a small test that clearly expresses our expectations

Write
just enough
code that will let the test
compile, but still fails
because our expectations
were not met

Run the test
and watch it fail
so we know that
the test is being
exercised

Write or modify just enough code to make the new test pass

Ensure all existing tests still pass

TDD Process

Review & Refactoring

Decide if any code needs to be refactored

Handle each refactoring separately

Make the change

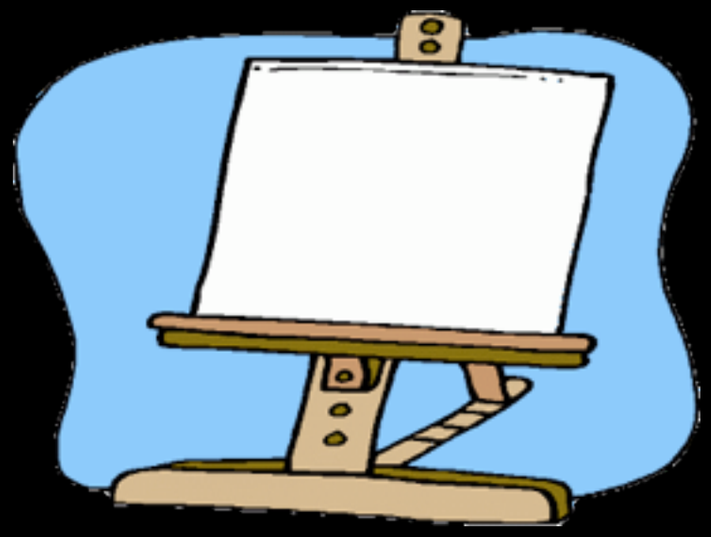
Run the tests

When we are satisfied with the changes, **AND** all the tests pass, the task is complete

The Bottom Line

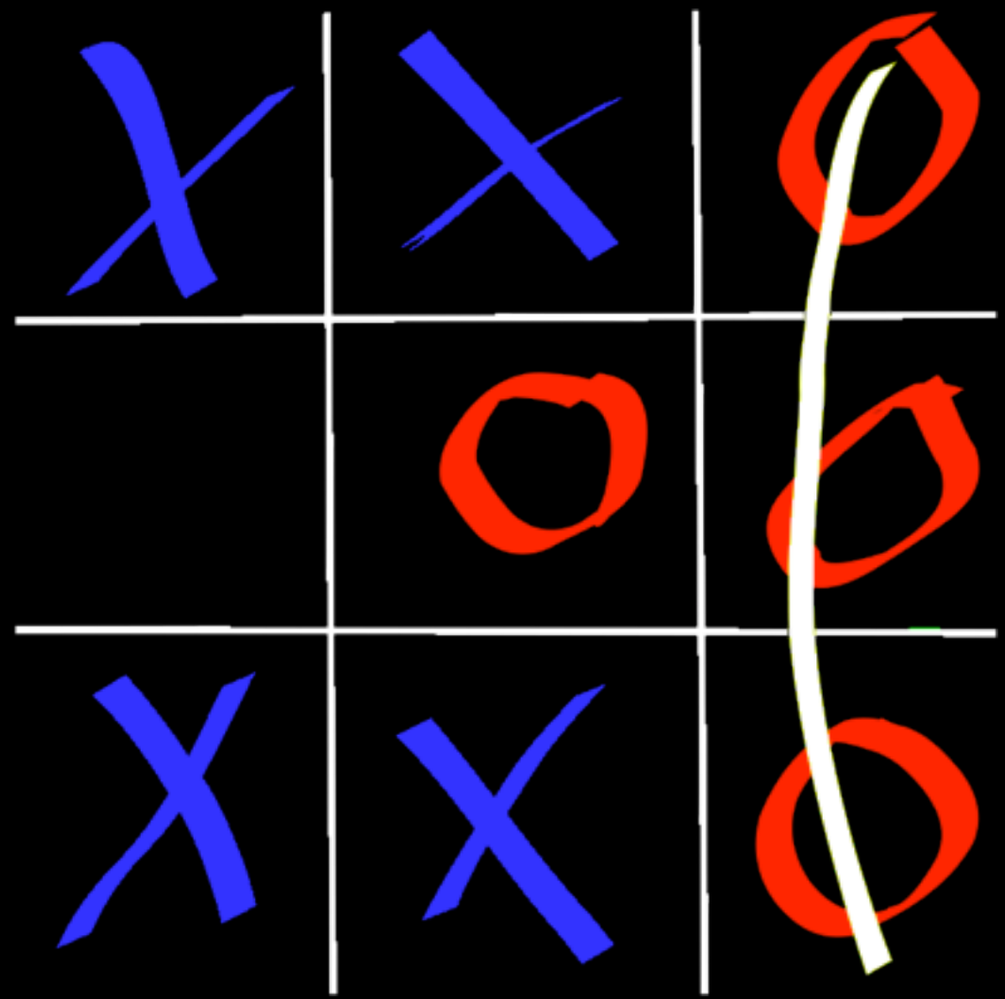
TDD Advantage

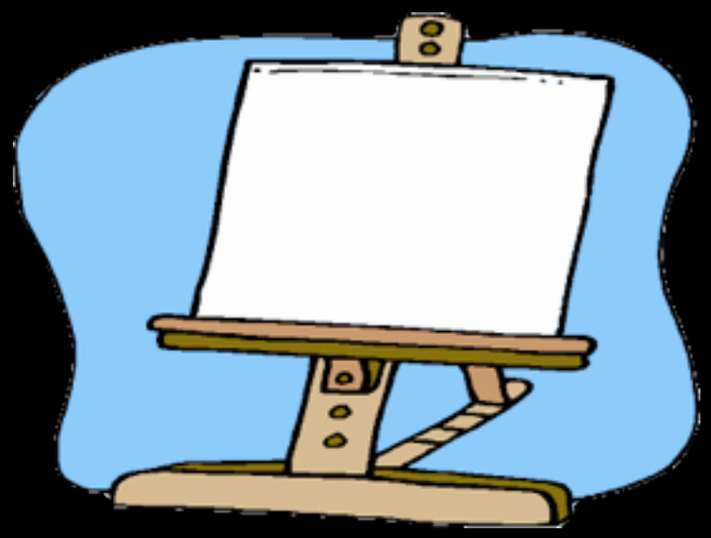
We are never very far from having working code, however incomplete it is.



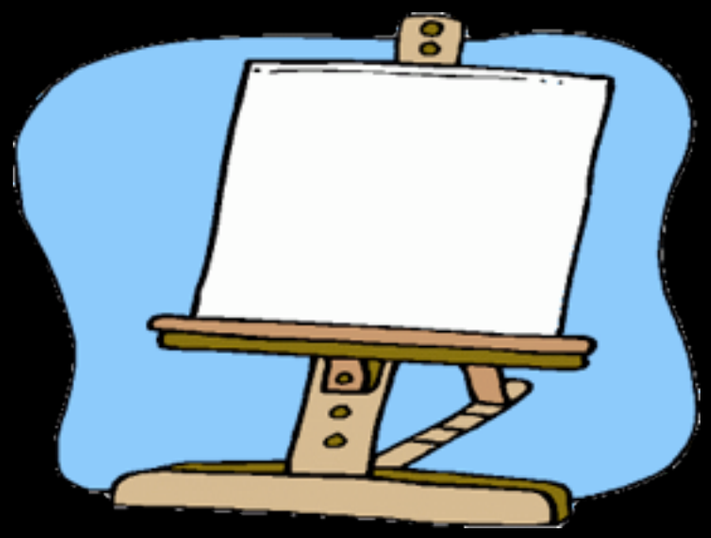
App's Purpose

Simple Tic-Tac-Toe Game

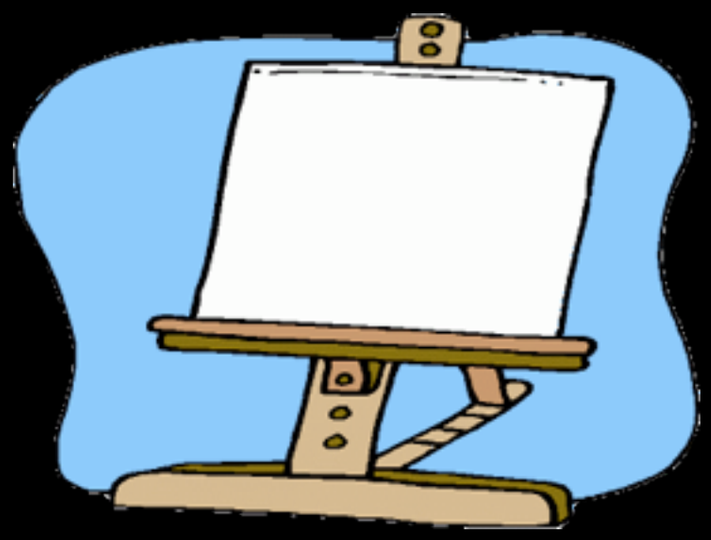




Think about our minimum feature set

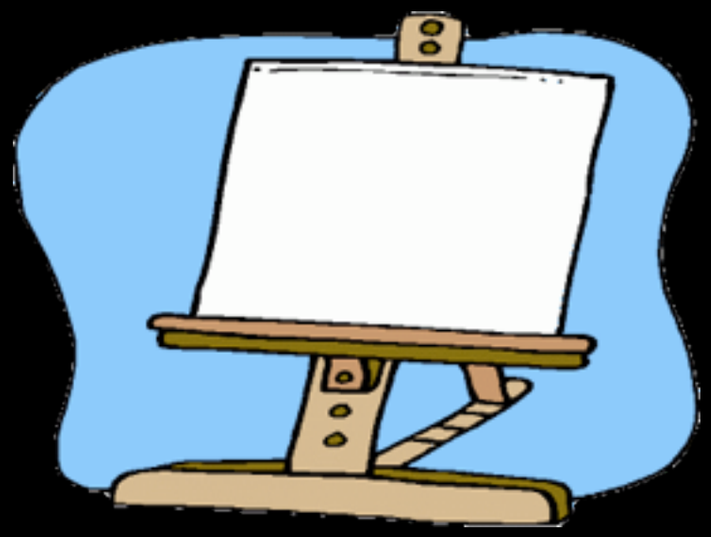


Simple Screen Mockup



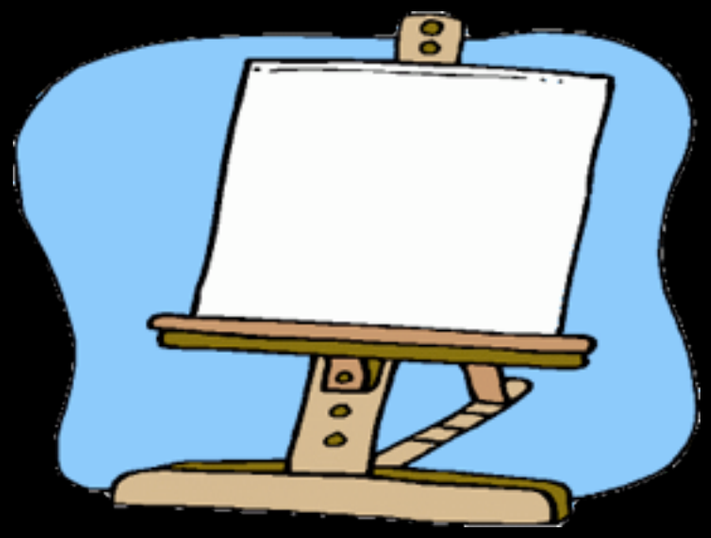
Simple model

- Players
- GameBoard
- GameManager
- GameView (UIKit based)



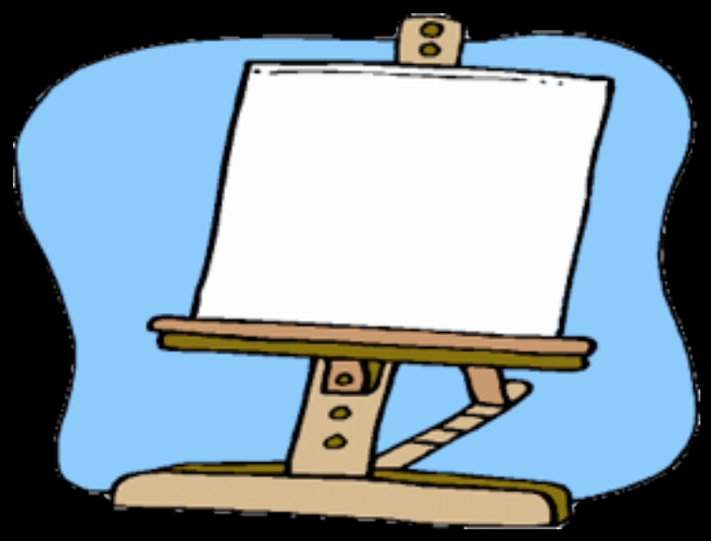
Players

Just a String



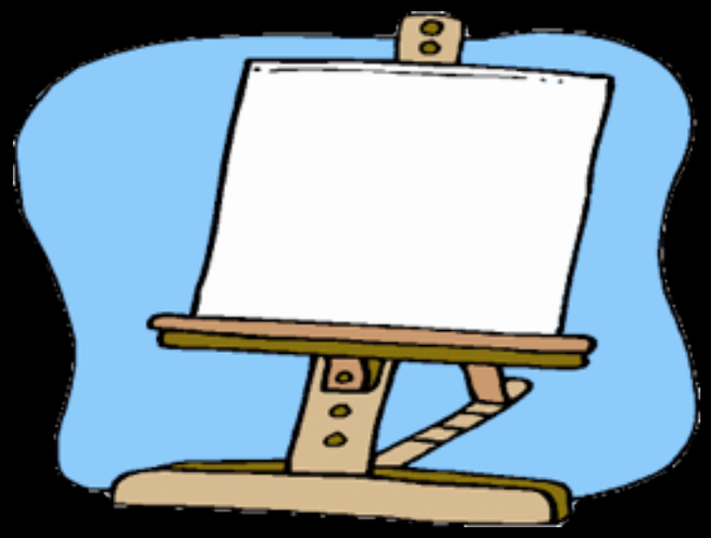
GameBoard

Keep track of positions
Validate moves
Check for winner or draw



GameManager

Manage starting a game
Track the players and turns
Make computer's move
Act as game controller



GameView

UIView with UIButton and UILabel instances

Time to write some tests!

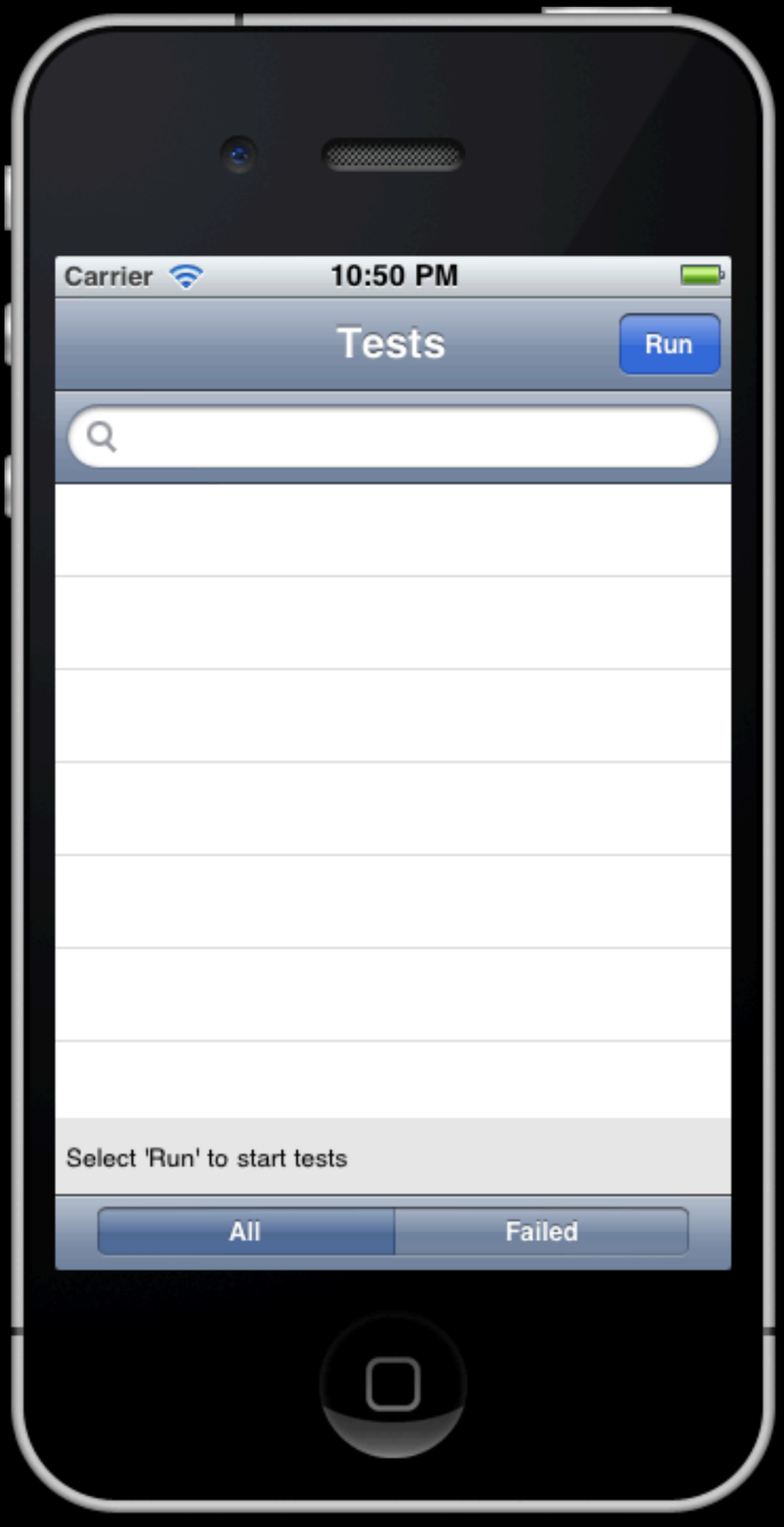
{cc000}

Where to begin testing?

Create TestGameBoard.m

```
// TestGameBoard.m

#import <GHUnitIOS/GHUnitIOS.h>

@interface TestGameBoard : GHTestCase { }
@end

@implementation TestGameBoard

@end
```

{cc007}


Carrier
10:50 PM
Tests
Run
Select 'Run' to start tests
All
Failed

Our First Test

```
@implementation TestGameBoard

- (void) testValidMove_row0_col0 {
    GameBoard *gameBoard =
      [[GameBoard alloc] init];

    [gameBoard movePlayer:@"playerA"
                      row:0 col:0];

    GHAssertEqualStrings(@"playerA",
      [gameBoard playerAtRow:0 col:0],
      @"playerAt should return 'playerA'");

    [gameBoard release];
}
```

{cc002}

```
// Gameboard.h

#import <Foundation/Foundation.h>

@interface GameBoard : NSObject {
}

@end
```

{cc003}

```
// GameBoard.m

#import "GameBoard.h"

@implementation GameBoard

@end
```

{cc003}

```
// TestGameBoard.m

#import <GHUnitIOS/GHUnitIOS.h>
#import "GameBoard.h"

@interface TestGameBoard : GHTestCase { }
@end

@implementation TestGameBoard

...
```

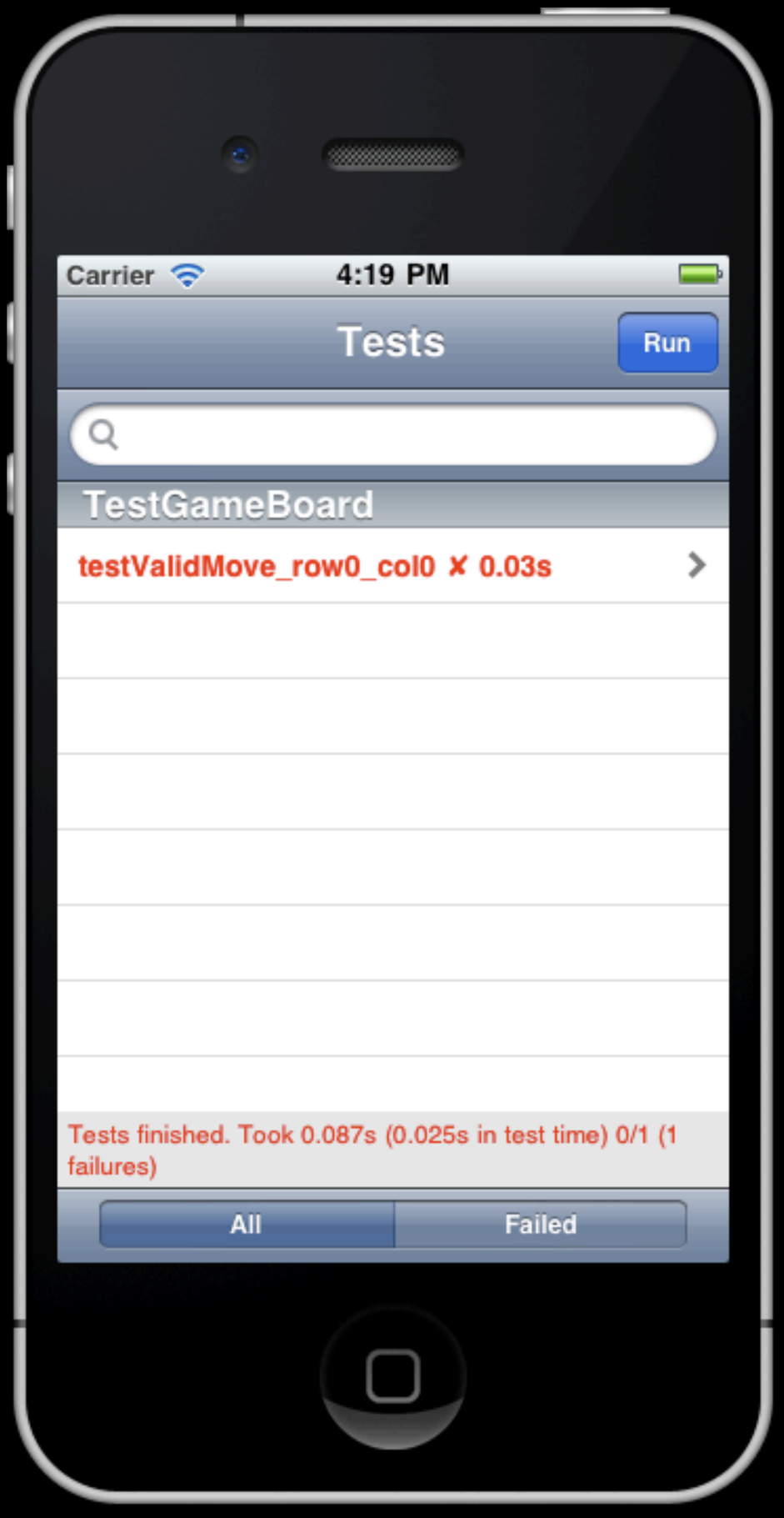
{cc004}

```
// Gameboard.h

#import <Foundation/Foundation.h>

@interface GameBoard : NSObject {
}

- (void) movePlayer:(NSString *) player
                row:(int) row
                col:(int) col;
- (NSString *) playerAtRow:(int) row
                       col:(int) col;

@end
```

{cc005}

```
// GameBoard.m
#import "GameBoard.h"

@implementation GameBoard

- (void) movePlayer:(NSString *) player
                row:(int) row
                col:(int) col {
}

- (NSString *) playerAtRow:(int) row
                       col:(int) col {
    return nil;
}
@end
```

{cc005}

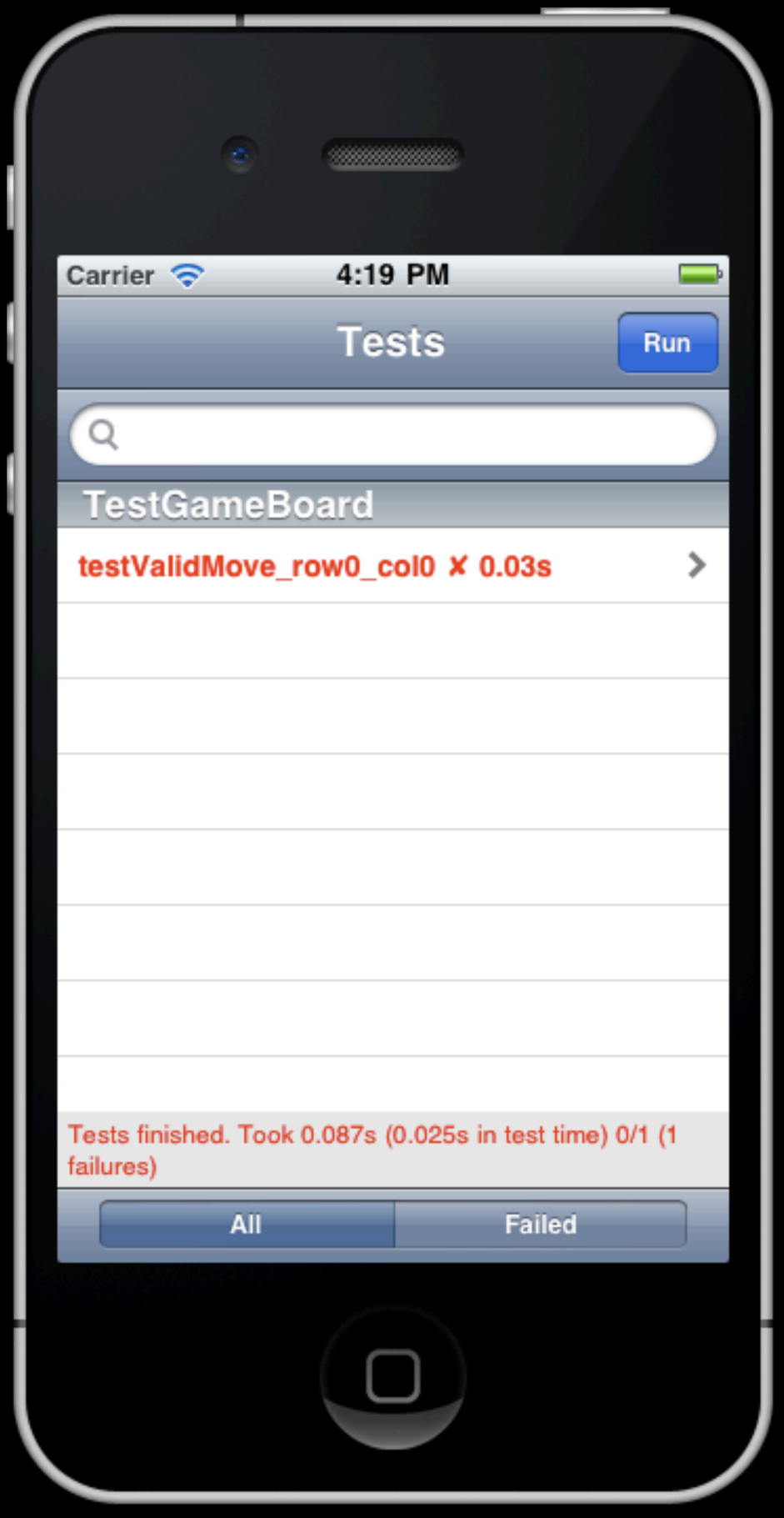

Carrier
4:19 PM
Tests
Run
TestGameBoard
testValidMove_row0_col0 ✘ 0.03s
Tests finished. Took 0.087s (0.025s in test time) 0/1 (1 failures)
All
Failed


Carrier
4:24 PM
Tests
testValidMove_r...
Re-run
TestGameBoard/testValidMove_row0_col0 ✘ 0.00s
Name: GHTestFailureException
File: /Users/dwsjoquist/ivygulch/presentations/CocoaConf/tddPart1/TDD_TicTacToe/TDD_TicTacToeTests/TestGameBoard.m
Line: 22
Reason: 'playerA' should be equal to '(null)'. playerAt... should return 'playerA'
#0 0xe0558c __exceptionPreprocess() (/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator4.3.sdk/System/Library/Frameworks/CoreFoundation.framework/CoreFoundation)
#1 0xf59313 objc_exception_throw() (/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator4.3.sdk/usr/lib/libobjc.A.dylib)
#2 0xe054e1 +[NSException raise] (/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator4.3.sdk/System/Library/Frameworks/CoreFoundation.framework/CoreFoundation)
#3 0x002d9b +[TestGameBoard testValidMove_row0_col0] (/Users/dwsjoquist/Library/Application Support/iPhone Simulator/4.3.2/Applications/8D13C02E-017C-4A72-

Not Just Casual Reassurance

Confidence in our ability to make changes

Allows us to focus on one thing at a time


Carrier
4:19 PM
Tests
Run
TestGameBoard
testValidMove_row0_col0 ✘ 0.03s
Tests finished. Took 0.087s (0.025s in test time) 0/1 (1 failures)
All
Failed

Resist the temptation to do more than is necessary!


DANGER

You will be writing code that does not yet have a test to validate it

You will be tempted
to skip writing
the test for it later

You might split your focus between too many things

You will probably write more code than you need


DANGER

K.I.S.S.

```
// Gameboard.h

#import <Foundation/Foundation.h>

@interface GameBoard : NSObject {
    NSString *player_;
}

...
```

{cc006}

```
// GameBoard.m
...

- (void) movePlayer:(NSString *) player
                row:(int) row
                col:(int) col {
    player_ = player;
}
```

```
- (NSString *) playerAtRow:(int) row
                       col:(int) col {
    return player_;
}

...
```

{cc006}


Carrier
4:27 PM
Tests
Run
TestGameBoard
testValidMove_row0_col0 ✓ 0.00s
Tests finished. Took 0.054s (0.000s in test time) 1/1 (0 failures)
All
Failed

Our Second Test

```
- (void) testTwoValidMoves {
    GameBoard *gameBoard =
      [[GameBoard alloc] init];
    [gameBoard movePlayer:@"playerA"
                      row:0 col:0];
    [gameBoard movePlayer:@"playerB"
                      row:1 col:1];

    GHAssertEqualStrings(@"playerA",
      [gameBoard playerAtRow:0 col:0],
      @"playerAt should return 'playerA'");
    GHAssertEqualStrings(@"playerB",
      [gameBoard playerAtRow:1 col:1],
      @"playerAt should return 'playerB'");

    [gameBoard release];
}
```

{cc007}


Carrier
4:33 PM
Tests
testTwoValidMo...
Re-run
TestGameBoard/testTwoValidMoves ✘ 0.00s
Name: GHTestFailureException
File: /Users/dwsjoquist/ivygulch/presentations/CocoaConf/tddPart1/TDD_TicTacToe/TDD_TicTacToeTests/TestGameBoard.m
Line: 34
Reason: 'playerA' should be equal to 'playerB'. playerAt 0 0 should return 'playerA'
#0 0xe0558c __exceptionPreprocess() (/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator4.3.sdk/System/Library/Frameworks/CoreFoundation.framework/CoreFoundation)
#1 0xf59313 objc_exception_throw() (/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator4.3.sdk/usr/lib/libobjc.A.dylib)
#2 0xe054e1 +[NSException raise] (/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator4.3.sdk/System/Library/Frameworks/CoreFoundation.framework/CoreFoundation)
#3 0x00278c +[TestGameBoard testTwoValidMoves] (/Users/dwsjoquist/Library/Application Support/iPhone Simulator/4.3.2/Applications/8D13C02E-017C-4A72-9D56-

```
// GameBoard.h

@interface GameBoard : NSObject {
    NSString * board_[3][3];
}

...
```

{cc008}

```
// GameBoard.m

...

- (void) movePlayer:player
                 row:(int) row
                 col:(int) col {
    board_[row][col] = player;
}

- (NSString *) playerAtRow:(int) row
                       col:(int) col {
    return board_[row][col];
}

...
```

{cc008}


Carrier
4:38 PM
Tests
Run
TestGameBoard
testTwoValidMoves ✓ 0.00s
testValidMove_row0_col0 ✓ 0.00s
Tests finished. Took 0.038s (0.000s in test time) 2/2 (0 failures)
All
Failed

Let's take a step back and review where we are

```
// TestGameBoard.m

@interface TestGameBoard : GHTestCase { }
GameBoard *gameBoard_;
@end

...
```

{cc009}

```
...

@implementation TestGameBoard

- (void) setUp {
    [super setUp];

    gameBoard_ = [[GameBoard alloc] init];
}

- (void) tearDown {
    [gameBoard_ release];

    [super tearDown];
}
...
```

{cc009}

```
...

- (void) testValidMove_row0_col0 {
    [gameBoard movePlayer:@"playerA"
                      row:0 col:0];

    GHAssertEqualStrings(@"playerA",
      [gameBoard playerAtRow:0 col:0],
      @"playerAt should return 'playerA'");
}

...
```

{cc009}

```
...

- (void) testTwoValidMoves {
    [gameBoard movePlayer:@"playerA"
                      row:0 col:0];
    [gameBoard movePlayer:@"playerB"
                      row:1 col:1];

    GHAssertEqualStrings(@"playerA",
      [gameBoard playerAtRow:0 col:0],
      @"playerAt should return 'playerA'");
    GHAssertEqualStrings(@"playerB",
      [gameBoard playerAtRow:1 col:1],
      @"playerAt should return 'playerB'");
}
@end
```

{cc009}


Carrier
4:41 PM
Tests
Run
TestGameBoard
testTwoValidMoves ✓ 0.00s
testValidMove_row0_col0 ✓ 0.00s
Tests finished. Took 0.083s (0.000s in test time) 2/2 (0 failures)
All
Failed

Squirrel!

```
// todo.txt

Tests to add:
- test that GameBoard detects moves outside
valid range

- test that GameBoard detects when a makes
an invalid move (selects a move already
made by a player)

- test that GameBoard to make sure only two
players can be used for a given game
```

{cc010}

The Bottom Line

It is critical to Stay on Target!

What's In Part 2?

- Deeper tests
- Components with dependencies
- OCMock usage

TDD in iOS

Coming soon: TDD/iOS tutorial series on

http://www.raywenderlich.com

Doug Sjoquist
http://www.sunetos.com
@dwsjoquist