

Stanford CS193p

Developing Applications for iOS

Fall 2013-14



Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Main.storyboard Card Game View Controller View Automatic CardGameViewController.m No Selection

Matchismo 2 targets, iOS SDK 7.0

Matchismo

Main.storyboard

CardGameViewController.h

CardGameViewController.m

Model

Card.h

Card.m

PlayingCard.h

PlayingCard.m

Deck.h

Deck.m

PlayingCardDeck.h

PlayingCardDeck.m

Images.xcassets

Supporting Files

MatchismoTests

Frameworks

Products

A♣

Flips: 0

Okay, let's continue building our Card Matching Game!

```
//  
// CardGameViewController.m  
// Matchismo  
//  
// Created by CS193p Instructor.  
// Copyright (c) 2013 Stanford University.  
// All rights reserved.  
  
#import "CardGameViewController.h"  
@implementation CardGameViewController()  
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;  
@end  
  
@implementation CardGameViewController  
  
- (IBAction)touchCardButton:(UIButton *)sender  
{  
    if ([sender.currentTitle length]) {  
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"]  
            forState:UIControlStateNormal];  
        [sender setTitle:@"" forState:UIControlStateNormal];  
    } else {  
        [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"]  
            forState:UIControlStateNormal];  
        [sender setTitle:@"A♣" forState:UIControlStateNormal];  
    }  
    self.flipCount++;  
}  
  
- (void)setFlipCount:(int)flipCount  
{  
    _flipCount = flipCount;  
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.  
        flipCount];  
    NSLog(@"flipCount changed to %d", self.flipCount);  
}  
  
@end
```

Stanford CS193p
Fall 2013

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Card G... View Automatic CardGameViewController.m No Selection

Close Navigator.

```
//  
// CardGameViewController.m  
// Matchismo  
// Created by CS193p Instructor.  
// Copyright (c) 2013 Stanford University.  
// All rights reserved.  
  
#import "CardGameViewController.h"  
  
@interface CardGameViewController()  
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;  
@property (nonatomic) int flipCount;  
@end  
  
@implementation CardGameViewController  
  
- (IBAction)touchCardButton:(UIButton *)sender  
{  
    if ([sender.currentTitle length]) {  
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];  
        [sender setTitle:@"" forState:UIControlStateNormal];  
    } else {  
        [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];  
        [sender setTitle:@"A♦" forState:UIControlStateNormal];  
    }  
    self.flipCount++;  
}  
  
- (void)setFlipCount:(int)flipCount  
{  
    _flipCount = flipCount;  
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];  
    NSLog(@"flipCount changed to %d", self.flipCount);  
}  
  
@end
```

A♣

Flips: 0

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Card G... View Automatic CardGameViewController.m No Selection

#import "CardGameViewController.h"

@interface CardGameViewController ()

@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;

@property (nonatomic) int flipCount;

@end

@implementation CardGameViewController

- (IBAction)touchCardButton:(UIButton *)sender

{

if ([sender titleForState:UIControlStateNormal] == @"Front") {

[sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];

[sender setTitle:@"Back" forState:UIControlStateNormal];

} else {

[sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];

[sender setTitle:@"A♦" forState:UIControlStateNormal];

}

self.flipCount++;

}

- (void)setFlipCount:(int)flipCount

{

_flipCount = flipCount;

self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];

NSLog(@"flipCount changed to %d", self.flipCount);

}

@end

Scrolled down.

The first thing we're going to do is go over last week homework assignment.

This is the only time we will go over the homework assignment in lecture.

A♣

Flips: 0

Stanford CS193p
Fall 2013

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) 1 2

Automatic CardGameViewController.m @interface CardGameViewController()

```
#import "CardGameViewController.h"

@interface CardGameViewController ()  
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;  
@property (nonatomic) int flipCount;  
@property (strong, nonatomic) Deck *deck; ① Unknown type name 'Deck'  
@end

@implementation CardGameViewController  
- (IBAction)touchCardButton:(UIButton *)sender  
{  
    if ([sender.currentTitle length]) {  
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];  
        [sender setTitle:@"" forState:UIControlStateNormal];  
    } else {  
        [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];  
        [sender setTitle:@"A♦" forState:UIControlStateNormal];  
    }  
    self.flipCount++;  
}  
- (void)setFlipCount:(int)flipCount  
{  
    _flipCount = flipCount;  
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];  
    NSLog(@"flipCount changed to %d", self.flipCount);  
}  
@end
```

Minor problem.

A♣

Flips: 0

Here's the `@property` for the deck you were supposed to add in the first homework assignment.

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Card G... View Automatic CardGameViewController.m No Selection

```
#import "CardGameViewController.h"
#import "Deck.h"

@interface CardGameViewController ()  
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;  
@property (nonatomic) int flipCount;  
@property (strong, nonatomic) Deck *deck;  
@end

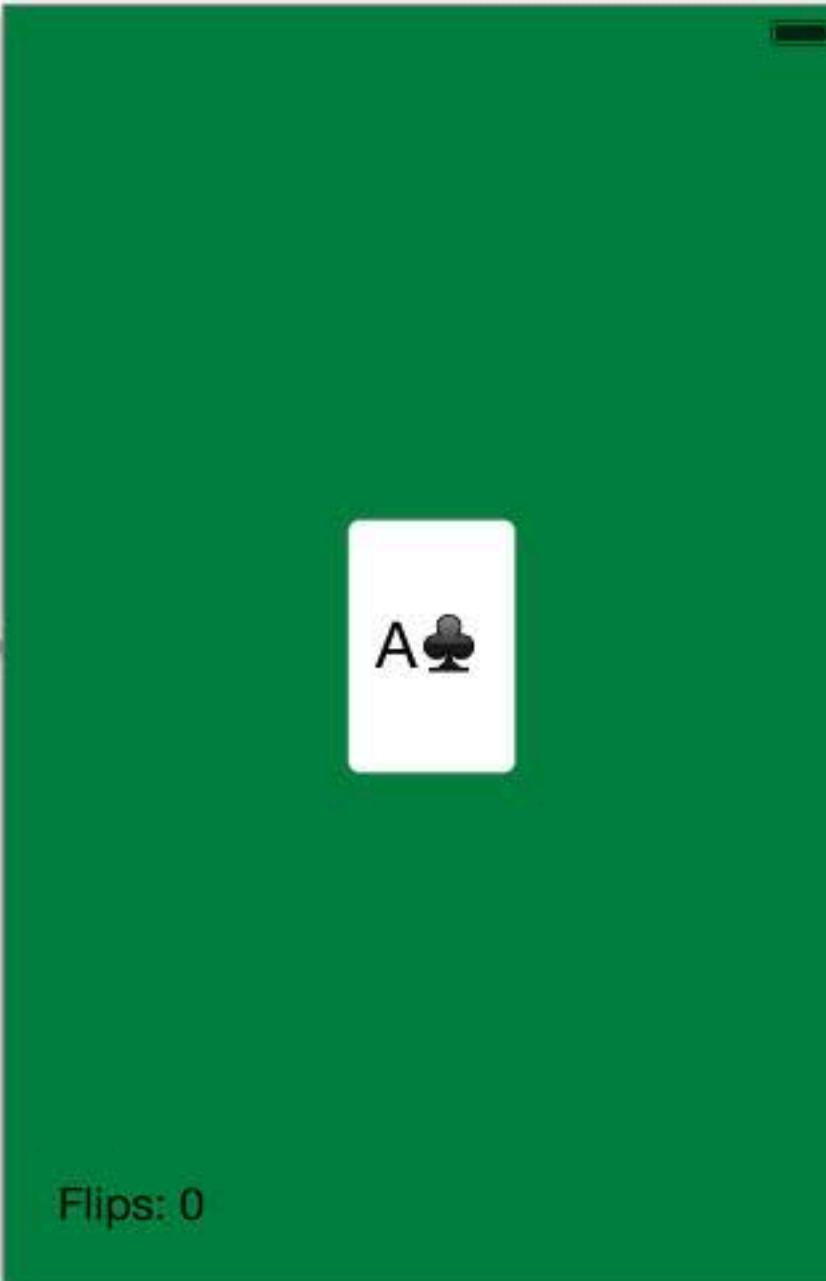
@implementation CardGameViewController

- (IBAction)touchCardButton:(UIButton *)sender  
{  
    if ([sender.currentTitle length]) {  
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];  
        [sender setTitle:@"" forState:UIControlStateNormal];  
    } else {  
        [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];  
        [sender setTitle:@"A♦" forState:UIControlStateNormal];  
    }  
    self.flipCount++;  
}

- (void)setFlipCount:(int)flipCount  
{  
    _flipCount = flipCount;  
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];  
    NSLog(@"flipCount changed to %d", self.flipCount);  
}

@end
```

We need to `#import Deck` if we want to use it in this class.



A♣

Flips: 0

The screenshot shows the Xcode IDE with the project "Matchismo" open. The main editor window displays the file `CardGameViewController.m`. The code implements a getter method for a `Deck` property using lazy instantiation:

```
#import "CardGameViewController.h"
#import "Deck.h"

@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@end

@implementation CardGameViewController

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
        [sender setTitle:@"A♣" forState:UIControlStateNormal];
    }
    self.flipCount++;
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}

@end
```

A callout bubble points to the `- (Deck *)createDeck` method with the text: "You'll see why we've factored out createDeck in a little while."

A callout bubble points to the `return [[PlayingCardDeck alloc] init];` line with the text: "We need to `#import PlayingCardDeck` if we want to use it in this class."

A callout bubble points to the `if (!_deck)` condition with the text: "Here's our getter method for lazy instantiation."

The Xcode interface includes a toolbar at the bottom with various icons, and the status bar at the top right shows "Stanford CS193p Fall 2013".

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Card G... View Automatic CardGameViewController.m No Selection

```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"

@interface CardGameViewController ()  
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;  
@property (nonatomic) int flipCount;  
@property (strong, nonatomic) Deck *deck;  
@end  
#import "PlayingCardDeck.h"  
@implementation CardGameViewController  
- (Deck *)deck  
{  
    if (!_deck) _deck = [self createDeck];  
    return _deck;  
}  
  
- (Deck *)createDeck  
{  
    return [[PlayingCardDeck alloc] init];  
}  
  
- (IBAction)touchCardButton:(UIButton *)sender  
{  
    if ([sender.currentTitle length]) {  
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];  
        [sender setTitle:@"" forState:UIControlStateNormal];  
    } else {  
        [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];  
        [sender setTitle:@"A♦" forState:UIControlStateNormal];  
    }  
    self.flipCount++;  
}  
  
- (void)setFlipCount:(int)flipCount  
{  
    _flipCount = flipCount;  
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];  
    NSLog(@"flipCount changed to %d", self.flipCount);  
}  
@end
```

A♣

Flips: 0

It is sort of unfortunate that we are importing PlayingCardDeck into this class since it is otherwise a generic card matching game Controller. In other words, there's really nothing that would prevent it from working with other Decks of other kinds of cards than PlayingCards. We'll use polymorphism next week to improve this state of affairs.

Even though this method returns a Deck, we are within our rights to return a PlayingCardDeck instance since PlayingCardDeck inherits from Deck (and thus "isa" Deck).

Stanford CS193p
Fall 2013

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch)

Automatic CardGameViewController.m -touchCardButton:

```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"

@interface CardGameViewController ()  
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;  
@property (nonatomic) int flipCount;  
@property (strong, nonatomic) Deck *deck;  
@end

@implementation CardGameViewController

- (Deck *)deck  
{  
    if (!_deck) _deck = [self createDeck];  
    return _deck;  
}

- (Deck *)createDeck  
{  
    return [[PlayingCardDeck alloc] init];  
}

- (IBAction)touchCardButton:(UIButton *)sender  
{  
    if ([sender.currentTitle length]) {  
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];  
        [sender setTitle:@"" forState:UIControlStateNormal];  
    } else {  
        Card *randomCard = [self.deck drawRandomCard];  
        [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];  
        [sender setTitle:@"A♦" forState:UIControlStateNormal];  
    }  
    self.flipCount++;  
}

- (void)setFlipCount:(int)flipCount  
{  
    _flipCount = flipCount;  
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];  
    NSLog(@"flipCount changed to %d", self.flipCount);  
}

@end
```

A♣

Flips: 0

Simply draw a random card ...

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Card G... View Automatic CardGameViewController.m -touchCardButton:

```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"

@interface CardGameViewController ()  
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;  
@property (nonatomic) int flipCount;  
@property (strong, nonatomic) Deck *deck;  
@end

@implementation CardGameViewController

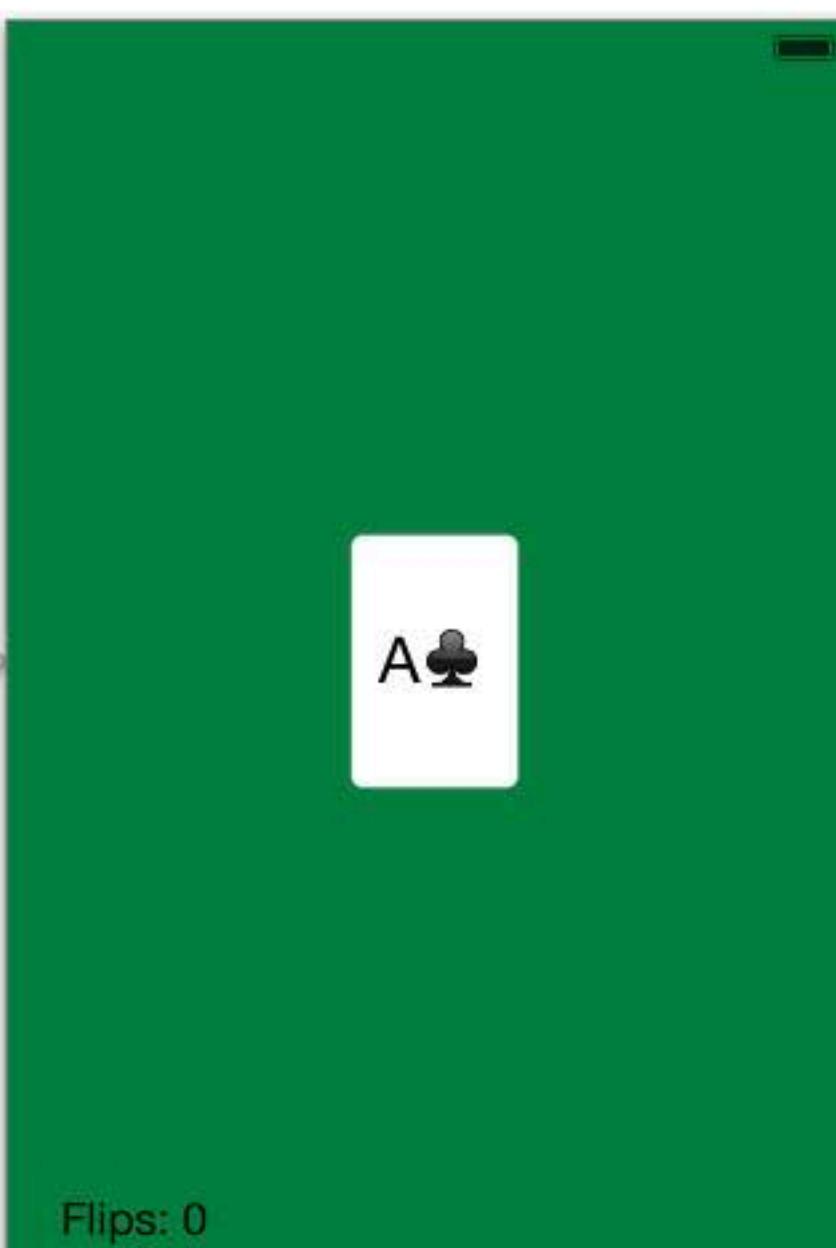
- (Deck *)deck  
{  
    if (!_deck) _deck = [self createDeck];  
    return _deck;  
}

- (Deck *)createDeck  
{  
    return [[PlayingCardDeck alloc] init];  
}

- (IBAction)touchCardButton:(UIButton *)sender  
{  
    if ([sender.currentTitle length]) {  
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];  
        [sender setTitle:@"" forState:UIControlStateNormal];  
    } else {  
        Card *randomCard = [self.deck drawRandomCard];  
        [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];  
        [sender setTitle:randomCard.contents forState:UIControlStateNormal];  
    }  
    self.flipCount++;  
}

- (void)setFlipCount:(int)flipCount  
{  
    _flipCount = flipCount;  
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];  
    NSLog(@"flipCount changed to %d", self.flipCount);  
}

@end
```



... and put its contents on the button.

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Automatic CardGameViewController.m -touchCardButton:

```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"

@interface CardGameViewController ()  
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;  
@property (nonatomic) int flipCount;  
@property (strong, nonatomic) Deck *deck;  
@end

@implementation CardGameViewController

- (Deck *)deck  
{  
    if (!_deck) _deck = [self createDeck];  
    return _deck;  
}  
  
- (Deck *)createDeck  
{  
    return [[PlayingCardDeck alloc] init];  
}  
  
- (IBAction)touchCardButton:(UIButton *)sender  
{  
    if ([sender.currentTitle length]) {  
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];  
        [sender setTitle:@"" forState:UIControlStateNormal];  
    } else {  
        Card *randomCard = [self.deck drawRandomCard];  
        if (randomCard) {  
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];  
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];  
        }  
    }  
    self.flipCount++;  
}  
  
- (void)setFlipCount:(int)flipCount  
{  
    _flipCount = flipCount;  
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];  
    NSLog(@"flipCount changed to %d", self.flipCount);  
}
```

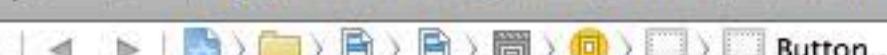
A♣

Flips: 0

But how do we handle running out of cards?
Easy.
Just don't flip to the front if the deck is empty
(which we know if drawRandomCard returns **nil**).

Protect against an empty deck.

Stanford CS193p
Fall 2013



```

@implementation CardGameViewController

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

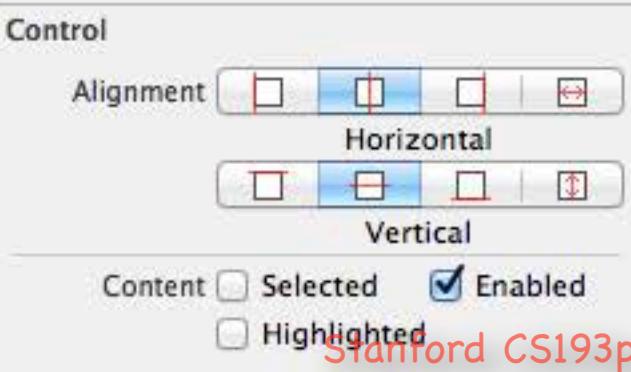
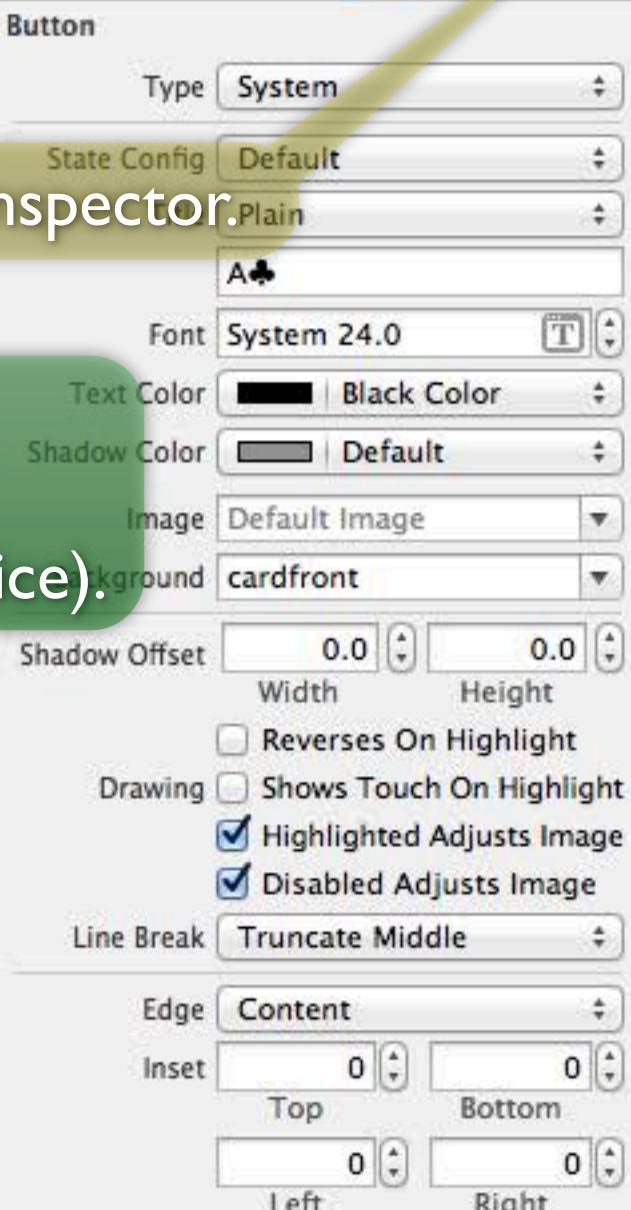
- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"]
                           forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"]
                           forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:
                         UIControlStateNormal];
        }
        self.flipCount++;
    }
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.
                           flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}
@end

```

Unhide the Attributes Inspector.

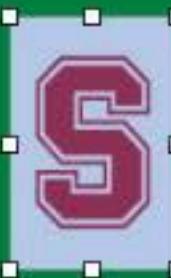
Finally, as Hinted at in the assignment,
let's start off with the card face down
(otherwise we'd end up showing the Ace of clubs twice).



```
@implementation CardGameViewController  
  
- (Deck *)deck  
{  
    if (!_deck) _deck = [self createDeck];  
    return _deck;  
}  
  
- (Deck *)createDeck  
{  
    return [[PlayingCardDeck alloc] init];  
}  
  
- (IBAction)touchCardButton:(UIButton *)sender  
{  
    if ([sender.currentTitle length]) {  
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"]  
            forState:UIControlStateNormal];  
        [sender setTitle:@"" forState:UIControlStateNormal];  
    } else {  
        Card *randomCard = [self.deck drawRandomCard];  
        if (randomCard) {  
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"]  
                forState:UIControlStateNormal];  
            [sender setTitle:randomCard.contents forState:  
                UIControlStateNormal];  
        }  
        self.flipCount++;  
    }  
}  
  
- (void)setFlipCount:(int)flipCount  
{  
    _flipCount = flipCount;  
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.  
        flipCount];  
    NSLog(@"flipCount changed to %d", self.flipCount);  
}  
  
@end
```

Just clear out the title ...

...and set the image to the back of the card.

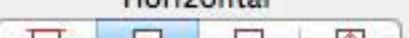


Flips: 0

Button

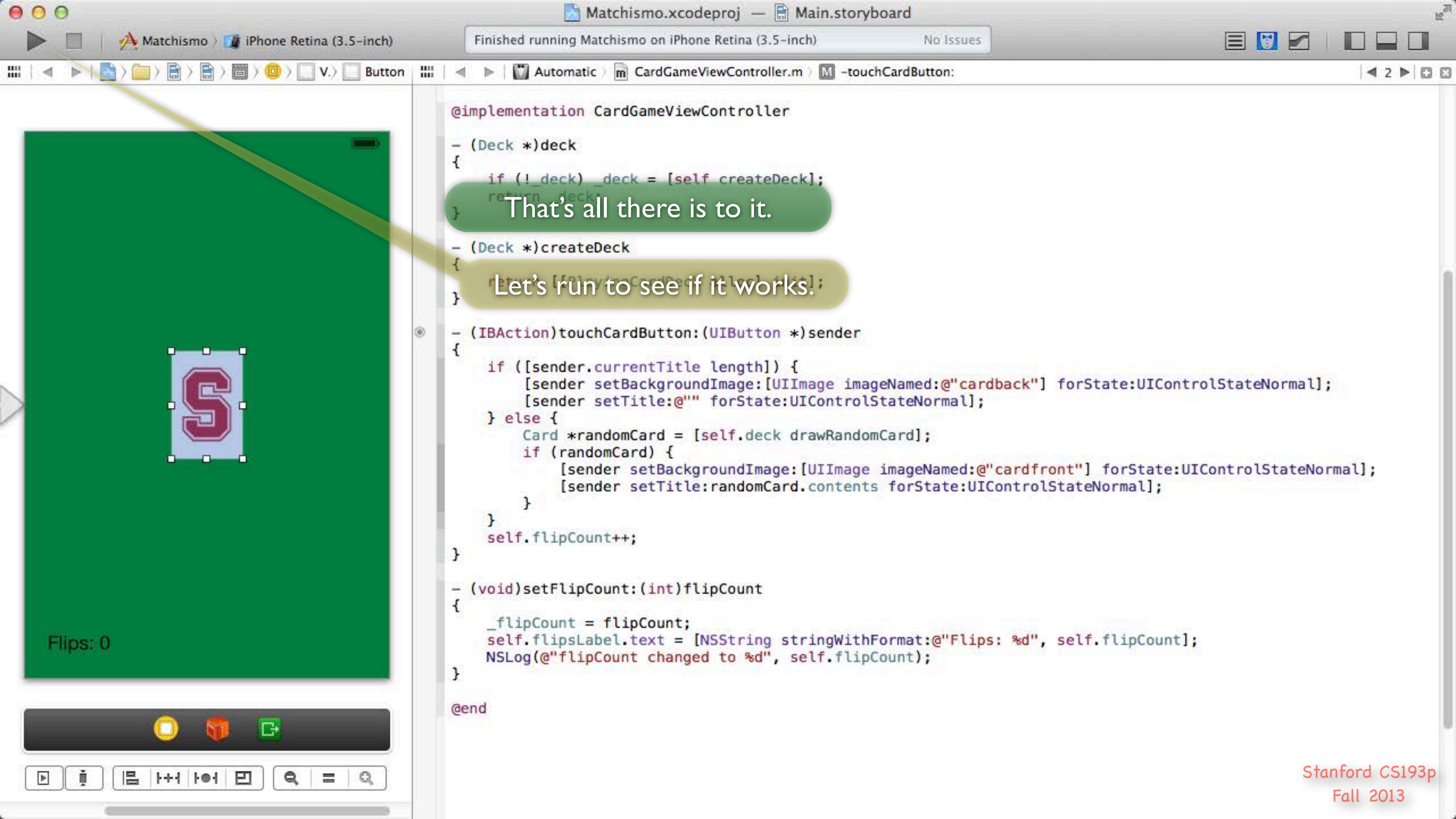
Type	System
State Config	Default
Title	Plain
Font	System 24.0
Text Color	Black Color
Shadow Color	Default
Image	Default Image
Background	cardback
Shadow Offset	0.0 0.0
Width	Height
<input type="checkbox"/> Reverses On Highlight	
Drawing	<input type="checkbox"/> Shows Touch On Highlight
<input checked="" type="checkbox"/> Highlighted Adjusts Image	
<input checked="" type="checkbox"/> Disabled Adjusts Image	
Line Break	Truncate Middle
Edge	Content
Inset	0 0
Top	Bottom
Left	Right

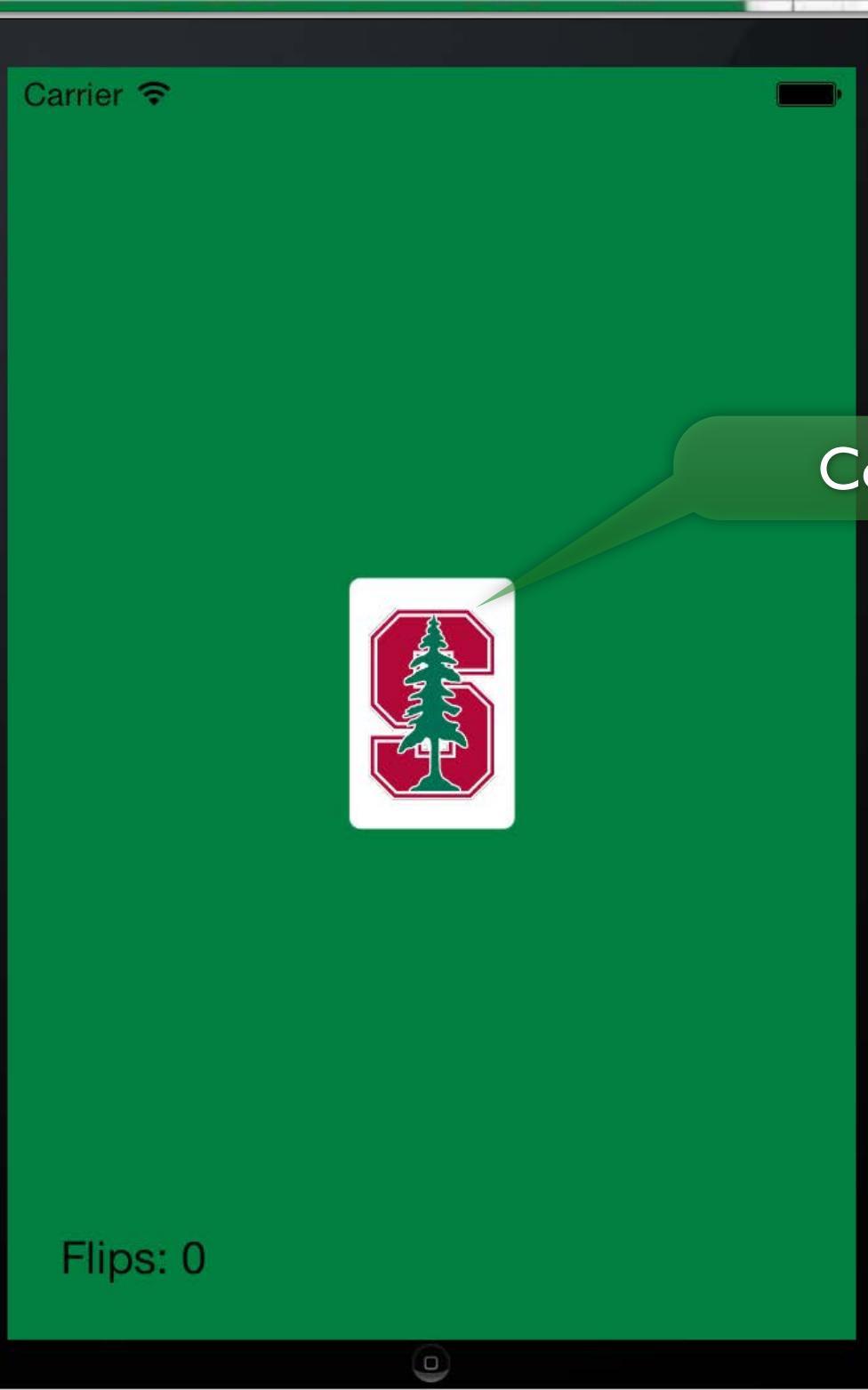
Control

Alignment	
	Horizontal
	
	Vertical
Content	<input type="checkbox"/> Selected
	<input checked="" type="checkbox"/> Enabled
	<input type="checkbox"/> Highlighted

Stanford CS193p

Fall 2013





```
implementation CardGameViewController  
  
(Deck *)deck  
  
    if (!_deck) _deck = [self createDeck];  
    return _deck;  
  
(Deck *)createDeck  
  
    return [[PlayingCardDeck alloc] init];  
  
    (IBAction)touchCardButton:(UIButton *)sender  
  
        if ([sender.currentTitle length]) {  
            [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];  
            [sender setTitle:@"" forState:UIControlStateNormal];  
        } else {  
            Card *randomCard = [self.deck drawRandomCard];  
            if (randomCard) {  
                [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];  
                [sender setTitle:randomCard.contents forState:UIControlStateNormal];  
            }  
        }  
        self.flipCount++;  
  
    (void)setFlipCount:(int)flipCount  
  
        _flipCount = flipCount;  
        self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];  
        NSLog(@"flipCount changed to %d", self.flipCount);
```

The screenshot shows the Xcode IDE with a running application on the left and its source code on the right.

Top Bar:

- Project: Matchismo
- Device: iPhone Retina (3.5-inch)
- File: Matchismo.xcodeproj — Main.storyboard
- Build: Running Matchismo on iPhone Retina (3.5-inch)
- Build Status: No Issues
- Navigation: Back, Forward, Home, Recent Projects, File, Editor, Run, Stop, Build, Scheme, Window, Help

Code View (CardGameViewController.m):

```
implementation CardGameViewController

(Deck *)deck

    if (!_deck) _deck = [self createDeck];
    return _deck;

(Deck *)createDeck

    return [[PlayingCardDeck alloc] init];

IBAction)touchCardButton:(UIButton *)sender

    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
    }
    self.flipCount++;

(void)setFlipCount:(int)flipCount

    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
```

Output View:

```
2013-00-00 00:00:00.154 Matchismo[11937:a0b] flipCount changed to 1
```

Bottom Status Bar:

- Auto
- Eye icon
- Info icon
- All Output

Matchismo.xcodeproj — Main.storyboard

Running Matchismo on iPhone Retina (3.5-inch) No Issues

Automatic CardGameViewController.m -touchCardButton:

Carrier

Touch

Back to face down. Yay!

Flips: 2

```
implementation CardGameViewController

(Deck *)deck

    if (!_deck) _deck = [self createDeck];
    return _deck;

(Deck *)createDeck

    return [[PlayingCardDeck alloc] init];

IBAction)touchCardButton:(UIButton *)sender

    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
    }
    self.flipCount++;

(void)setFlipCount:(int)flipCount

    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);

2013-00-00 00:00:00.154 Matchismo[11937:a0b] flipCount changed to 1
2013-00-00 00:00:00.318 Matchismo[11937:a0b] flipCount changed to 2
```

All Output

Matchismo.xcodeproj — Main.storyboard

Running Matchismo on iPhone Retina (3.5-inch) No Issues

Automatic CardGameViewController.m -touchCardButton:

Carrier

Touch

J♥

A different card! It's working!

Flips: 3

```
implementation CardGameViewController

(Deck *)deck

    if (!_deck) _deck = [self createDeck];
    return _deck;

(Deck *)createDeck

    return [[PlayingCardDeck alloc] init];

IBAction)touchCardButton:(UIButton *)sender

    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
        self.flipCount++;
    }

(void)setFlipCount:(int)flipCount

    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);

2013-00-00 00:00:00.154 Matchismo[11937:a0b] flipCount changed to 1
2013-00-00 00:00:00.318 Matchismo[11937:a0b] flipCount changed to 2
2013-00-00 00:00:00.865 Matchismo[11937:a0b] flipCount changed to 3
```

All Output

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Automatic CardGameViewController.m -touchCardButton:

```
@implementation CardGameViewController

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCard:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
    }
    self.flipCount++;
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}

@end
```

This game would be a lot more interesting with more than just one card!

Flips: 0

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Card G... View Automatic CardGameViewController.m -touchCardButton:

```
@implementation CardGameViewController

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    }
    Card *randomCard = [self.deck drawRandomCard];
    if (randomCard) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
        [sender setTitle:randomCard.contents forState:UIControlStateNormal];
    }
    self.flipCount++;
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}

@end
```

Let's start by moving our one card into the upper left corner.

Flips: 0

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Automatic CardGameViewController.m -touchCardButton:

```
@implementation CardGameViewController

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
        self.flipCount++;
    }
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}

@end
```

Copy and paste it
to make a second card.

Flips: 0

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Card G... View Automatic CardGameViewController.m -touchCardButton:

```
@implementation CardGameViewController

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
    }
    self.flipCount++;
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}

@end
```

Reposition it using blue guidelines.

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Card G... View Automatic CardGameViewController.m -touchCardButton:

```
@implementation CardGameViewController

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
    }
    self.flipCount++;
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}

@end
```

Flips: 0

Make a total of 12 cards.

Matchismo.xcodeproj — Main.storyboard

Running Matchismo on iPhone Retina (3.5-inch)

No Issues

Card Game View Controller

Automatic

CardGameViewController.m

-touchCardButton:

```
implementation CardGameViewController

(Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

(Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
    }
    self.flipCount++;
}

(void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}
```

Carrier

Flips: 0

Let's Run and see what we've got so far.

Matchismo.xcodeproj — Main.storyboard

Running Matchismo on iPhone Retina (3.5-inch) No Issues

Card GameViewController.m -touchCardButton:

implementation CardGameViewController

```
(Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

(Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

(void)touchCard:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
    }
    self.flipCount++;
}

(void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}
```

Carrier

Flips: 4

Tapping should flip over any of the cards.

2013-00-00 00:00:00.544 Matchismo[13384:a0b] flipCount changed to 2
2013-00-00 00:00:00.880 Matchismo[13384:a0b] flipCount changed to 3
2013-00-00 00:00:01.544 Matchismo[13384:a0b] flipCount changed to 4

Stanford CS193p
Fall 2013

Matchismo.xcodeproj — Main.storyboard

Running Matchismo on iPhone Retina (3.5-inch) No Issues

Card Game View Controller Automatic CardGameViewController.m -touchCardButton:

Carrier

Tapping again should flip it back down.

```
implementation CardGameViewController

(Deck *)deck

    if (!_deck) _deck = [self createDeck];
    return _deck;

(Deck *)createDeck

    return [[PlayingCardDeck alloc] init];

if ([sender.currentTitle length]) {
    [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
    [sender setTitle:@"" forState:UIControlStateNormal];
} else {
    Card *randomCard = [self.deck drawRandomCard];
    if (randomCard) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
        [sender setTitle:randomCard.contents forState:UIControlStateNormal];
    }
}
self.flipCount++;

(void)setFlipCount:(int)flipCount

    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);

2013-00-00 00:00:00.000 Matchismo[13384:a0b] flipCount changed to 5
2013-00-00 00:00:01.544 Matchismo[13384:a0b] flipCount changed to 4
2013-00-00 00:00:02.137 Matchismo[13384:a0b] flipCount changed to 5
```

Stanford CS193p
Fall 2013

Matchismo.xcodeproj — Main.storyboard

Running Matchismo on iPhone Retina (3.5-inch) No Issues

Card Game View Controller Automatic CardGameViewController.m -touchCardButton:

Carrier

As we might expect, tapping yet again brings up a different, random card.

implementation CardGameViewController

```
(Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

(Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

-(void)-touchCardButton:(UIButton *)sender
{
    if ([sender isKindOfClass:[UIButton class]]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
    }
    self.flipCount++;
}

(void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}
```

2013-00-00 00:00:01.544 Matchismo[13384:a0b] flipCount changed to 4
2013-00-00 00:00:02.137 Matchismo[13384:a0b] flipCount changed to 5
2013-00-00 00:00:03.366 Matchismo[13384:a0b] flipCount changed to 6

Stanford CS193p
Fall 2013

The screenshot shows the Xcode IDE with a project named "Matchismo". The main window displays a storyboard scene for an iPhone Retina (3.5-inch) device. The scene contains a green background with a 3x4 grid of cards. Each card features a large red letter 'S' on a white background. In the bottom-left corner of the green area, there is a label "Flips: 0". A yellow speech bubble in the top-left corner says "Stop.". On the right side of the screen, the code for the CardGameViewController.m file is visible:

```
@implementation CardGameViewController

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

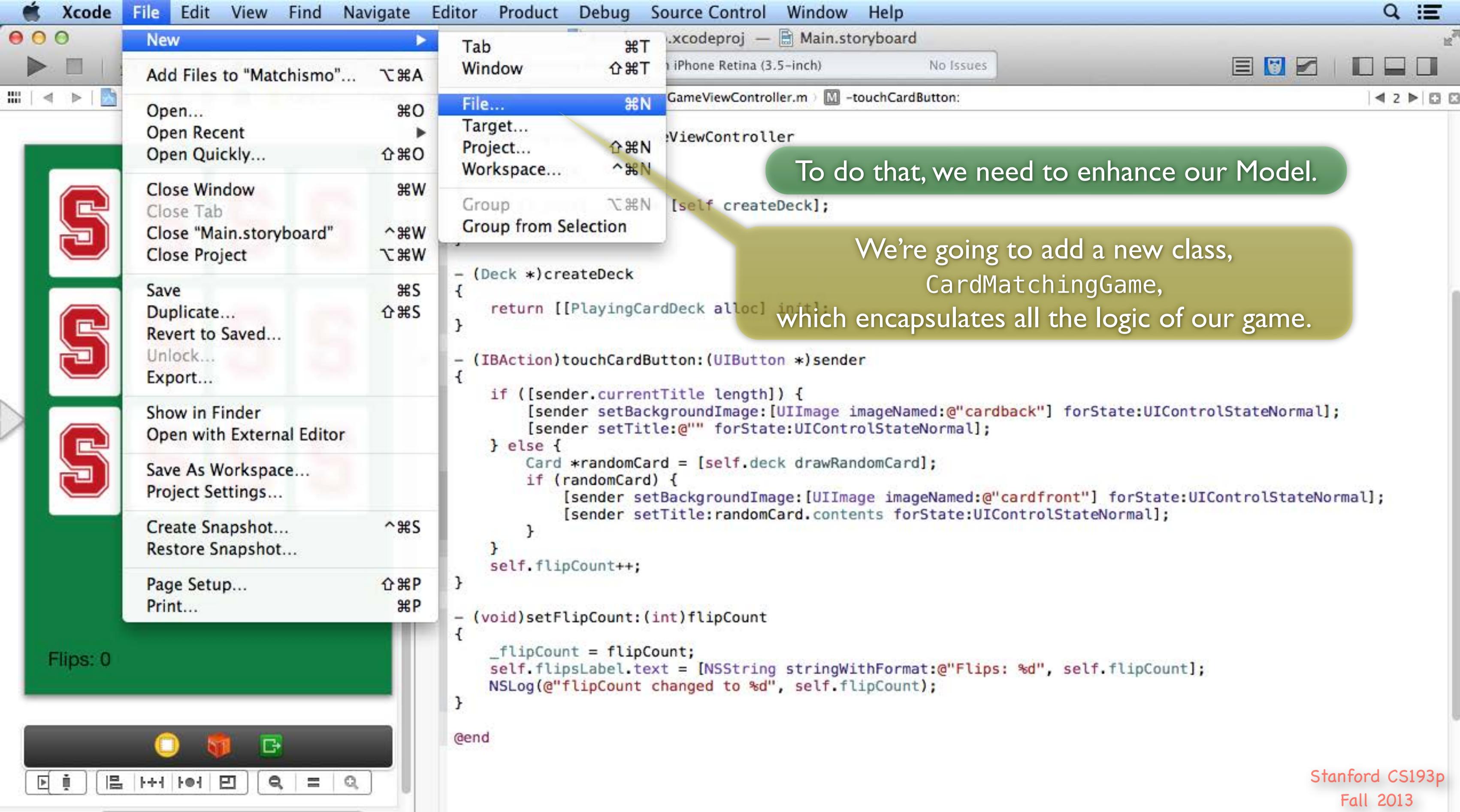
- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
    }
    self.flipCount++;
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}

@end
```

A green callout box points to the code line `- (Deck *)createDeck` with the text: "Now it's time to make a card matching game that actually matches cards!".



Choose a template for your new file:

iOS

Cocoa Touch

C and C++

User Interface

Core Data

Resource

Other

OS X

Cocoa

C and C++

User Interface

Core Data

Resource

Other



Objective-C class



Objective-C category



Objective-C class extension



Objective-C protocol



Objective-C test case class



Objective-C class

An Objective-C class, with implementation and header files.

Objective-C class, of course.

Cancel

Previous

Next

Flips: 0

```
_flipCount = flipCount;
self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
NSLog(@"flipCount changed to %d", self.flipCount);

}

@end
```

Choose options for your new file:

Call it CardMatchingGame.

Class **CardMatchingGame**Subclass of **NSObject**

- Targeted for iPad
- With XIB for user interface

Subclass of **NSObject**.

```
state:UIControlStateNormal];  
state:UIControlStateNormal];  
];
```

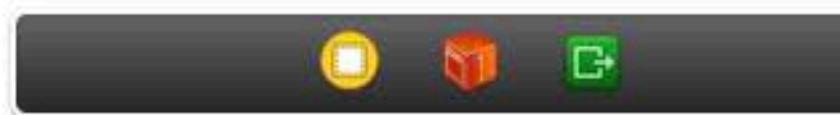
Cancel

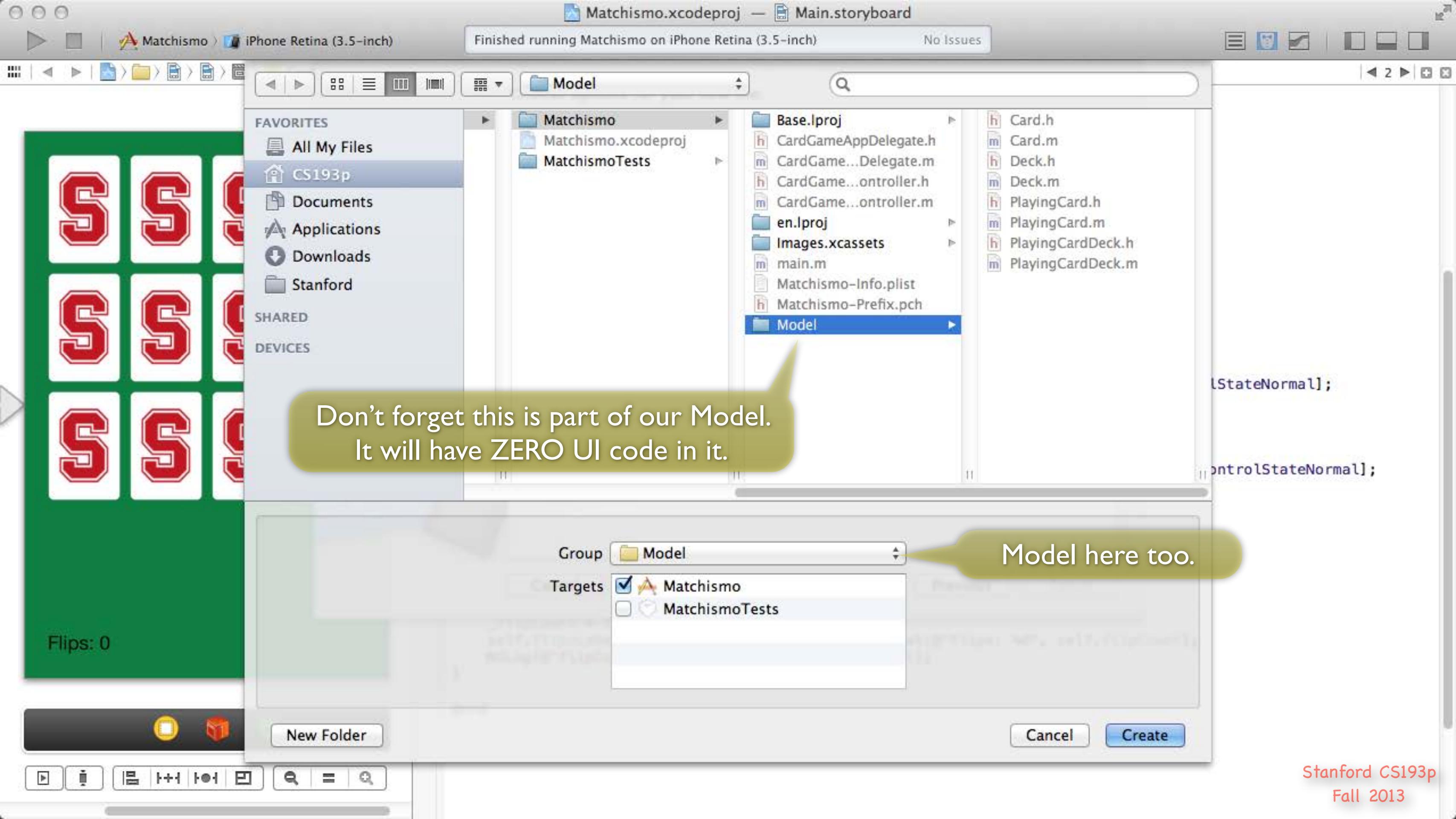
Previous

Next

Flips: 0

```
_flipCount = flipCount;  
self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];  
NSLog(@"flipCount changed to %d", self.flipCount);  
}  
  
@end
```





```
// CardMatchingGame.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import <Foundation/Foundation.h>

@interface CardMatchingGame : NSObject

@end
```

```
// CardMatchingGame.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

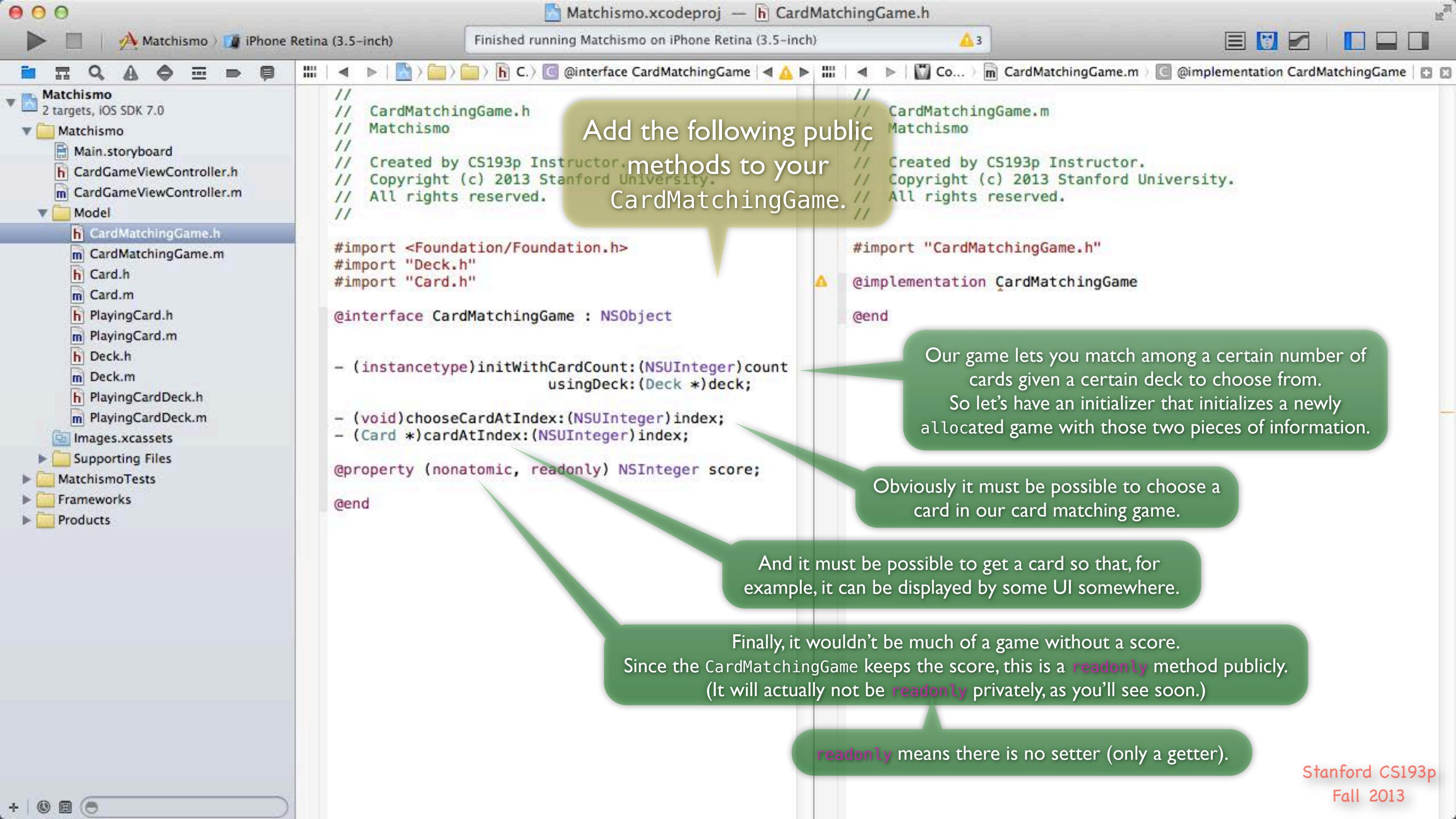
#import "CardMatchingGame.h"

@implementation CardMatchingGame

@end
```

Hopefully your CardMatchingGame appeared here in the Model group. If not, you can simply drag it there.

Often the best way to start designing a new class is to fill out its `@interface` since that delineates what it is this class is responsible for doing.



```
// CardMatchingGame.h
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

- (instancetype)initWithCardCount:(NSUInteger)count
    usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end
```

Uh oh! Warning!

```
// CardMatchingGame.m
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import "CardMatchingGame.h"

@implementation CardMatchingGame
    // Method definitions for chooseCardAtIndex:, initWithCardCount:usingDeck:, and cardAtIndex: are missing here.
@end
```

Method definition for 'chooseCardAtIndex:' not found
Method definition for 'initWithCardCount:usingDeck:' not found
Method definition for 'cardAtIndex:' not found

Click here to see what this warning is all about.

No problem.
Incomplete implementation.
That makes sense because we haven't implemented any of these public methods yet.

You should never submit code in this course that has any warnings or errors.

Notice that none of these methods has anything to do with user-interface.
It is up to the Controller to interpret the Model into something presented to the user via the View.

This area where we put private properties is called a “Class Extension”.

```

// CardMatchingGame.h
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

- (instancetype)initWithCardCount:(NSUInteger)count
    usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;
@end

```

Note the ()

```

// CardMatchingGame.m
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import "CardMatchingGame.h"

@interface CardMatchingGame()
@property (nonatomic, readwrite) NSInteger score;
@end

@implementation CardMatchingGame

@end

```

Let's start by redeclaring the **score** to not be **readonly** in our implementation (only). We usually don't use this **readwrite** directive unless publicly we made the **@property readonly** (since **readwrite** is the default).

It's perfectly fine for a **@property** to be **readonly** both publicly and privately.

```

Matchismo
  2 targets, iOS SDK 7.0
    Matchismo
      Main.storyboard
      CardGameViewController.h
      CardGameViewController.m
    Model
      CardMatchingGame.h
      CardMatchingGame.m
      Card.h
      Card.m
      PlayingCard.h
      PlayingCard.m
      Deck.h
      Deck.m
      PlayingCardDeck.h
      PlayingCardDeck.m
    Images.xcassets
    Supporting Files
    MatchismoTests
    Frameworks
    Products

```

```

// CardMatchingGame.h
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

- (instancetype)initWithCardCount:(NSUInteger)count
                           usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;
@end

```

```

// CardMatchingGame.m
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import "CardMatchingGame.h"

@interface CardMatchingGame()
@property (nonatomic, readwrite) NSInteger score;
@property (nonatomic, strong) NSMutableArray *cards; // of Card
@end

@implementation CardMatchingGame
@end

```

Our game needs to keep track of the cards,
so we need a private `@property` to do that.

Indeed there is no way to express in Objective-C that
this array should only have Card objects in it.
One might argue that that is a shortcoming.
All we can do is be sure to comment what we intend.

```

Matchismo
  2 targets, iOS SDK 7.0
    Matchismo
      Main.storyboard
      CardGameViewController.h
      CardGameViewController.m
    Model
      CardMatchingGame.h
      CardMatchingGame.m
      Card.h
      Card.m
      PlayingCard.h
      PlayingCard.m
      Deck.h
      Deck.m
      PlayingCardDeck.h
      PlayingCardDeck.m
    Images.xcassets
    Supporting Files
    MatchismoTests
    Frameworks
    Products

```

```

// CardMatchingGame.h
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

- (instancetype)initWithCardCount:(NSUInteger)count
                           usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end

```

```

// CardMatchingGame.m
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import "CardMatchingGame.h"

@interface CardMatchingGame()
@property (nonatomic, readwrite) NSInteger score;
@property (nonatomic, strong) NSMutableArray *cards; // of Card
@end

@implementation CardMatchingGame

- (NSMutableArray *)cards
{
    if (!_cards) _cards = [[NSMutableArray alloc] init];
    return _cards;
}

@end

```

Lazy instantiation!
Hopefully this is quite familiar
to you by now.

```
// CardMatchingGame.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

- (instancetype)initWithCardCount:(NSUInteger)count
                           usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end
```

```
// CardMatchingGame.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import "CardMatchingGame.h"

@interface CardMatchingGame()
@property (nonatomic, readwrite) NSInteger score;
@property (nonatomic, strong) NSMutableArray *cards; // of Card
@end

@implementation CardMatchingGame

- (NSMutableArray *)cards
{
    if (!_cards) _cards = [[NSMutableArray alloc] init];
    return _cards;
}

- (instancetype)initWithCardCount:(NSUInteger)count
                           usingDeck:(Deck *)deck
{
    self = [super init];

    if (self) {

    }

    return self;
}
```

Start off our initializer by letting our **super** class have a chance to initialize itself (and checking for failure return of **nil**).

```

// CardMatchingGame.h
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
    usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end

```

This is our class's designated initializer.
That means that it is not legally initialized unless this gets called at some point.
We must always call our superclass's designated initializer from our designated initializer
(if this were just a convenience initializer, we'd have to call our own designated initializer from it).

Classes can have multiple initializers,
but obviously only one designated initializer.

```

// CardMatchingGame.m
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.

#import "CardMatchingGame.h"

@interface CardMatchingGame()
@property (nonatomic, readwrite) NSInteger score;
@property (nonatomic, strong) NSMutableArray *cards; // of Card
@end

@implementation CardMatchingGame

- (NSMutableArray *)cards
{
    if (!_cards) _cards = [[NSMutableArray alloc] init];
    return _cards;
}

- (instancetype)initWithCardCount:(NSUInteger)count
    usingDeck:(Deck *)deck
{
    self = [super init]; // super's designated initializer
    if (self) {
    }
    return self;
}

@end

```

You should always comment which initializer is your designated initializer if it is different from your superclass's.

NSObject's designated initializer is init.

```

Matchismo
  2 targets, iOS SDK 7.0
    Matchismo
      Main.storyboard
      CardGameViewController.h
      CardGameViewController.m
    Model
      CardMatchingGame.h
      CardMatchingGame.m
      Card.h
      Card.m
      PlayingCard.h
      PlayingCard.m
      Deck.h
      Deck.m
      PlayingCardDeck.h
      PlayingCardDeck.m
      Images.xcassets
      Supporting Files
    MatchismoTests
    Frameworks
    Products

```

```

// CardMatchingGame.h
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
                           usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end

```

```

// CardMatchingGame.m
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import "CardMatchingGame.h"

@interface CardMatchingGame()
@property (nonatomic, readwrite) NSInteger score;
@property (nonatomic, strong) NSMutableArray *cards; // of Card
@end

@implementation CardMatchingGame

- (NSMutableArray *)cards
{
    if (!_cards) _cards = [[NSMutableArray alloc] init];
    return _cards;
}

- (instancetype)initWithCardCount:(NSUInteger)count
                           usingDeck:(Deck *)deck
{
    self = [super init]; // super's designated initializer

    if (self) {
        for (int i = 0; i < count; i++) {
            Card *card = [deck drawRandomCard];
            [self.cards addObject:card];
        }
    }
    return self;
}

@end

```

All we need to do to initialize our game is to iterate through the passed count of cards, drawRandomCard from the passed deck, then addObject: to our NSMutableArray of cards each time.

```

Matchismo
  2 targets, iOS SDK 7.0
    Matchismo
      Main.storyboard
      CardGameViewController.h
      CardGameViewController.m
    Model
      CardMatchingGame.h
      CardMatchingGame.m
      Card.h
      Card.m
      PlayingCard.h
      PlayingCard.m
      Deck.h
      Deck.m
      PlayingCardDeck.h
      PlayingCardDeck.m
      Images.xcassets
      Supporting Files
      MatchismoTests
      Frameworks
      Products

```

```

// CardMatchingGame.h
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
                           usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end

```

```

// CardMatchingGame.m
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import "CardMatchingGame.h"

@interface CardMatchingGame()
@property (nonatomic, readwrite) NSInteger score;
@property (nonatomic, strong) NSMutableArray *cards; // of Card
@end

@implementation CardMatchingGame

- (NSMutableArray *)cards
{
    if (!_cards) _cards = [[NSMutableArray alloc] init];
    return _cards;
}

- (instancetype)initWithCardCount:(NSUInteger)count
                           usingDeck:(Deck *)deck
{
    self = [super init]; // super's designated initializer

    if (self) {
        for (int i = 0; i < count; i++) {
            Card *card = [deck drawRandomCard];
            if (card) {
                [self.cards addObject:card];
            } else {
                self = nil;
                break;
            }
        }
    }
    return self;
}

```

Adding `nil` to an `NSMutableArray` will **crash your program. Let's protect against this!**

Note that we will return `nil` if we cannot initialize properly given the arguments passed.

Matchismo
2 targets, iOS SDK 7.0

Matchismo
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Model
CardMatchingGame.h
CardMatchingGame.m
Card.h
Card.m
PlayingCard.h
PlayingCard.m
Deck.h
Deck.m
PlayingCardDeck.h
PlayingCardDeck.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products

```

// CardMatchingGame.h
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
                           usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end

```

```

#import "CardMatchingGame.h"

@interface CardMatchingGame()
@property (nonatomic, readwrite) NSInteger score;
@property (nonatomic, strong) NSMutableArray *cards; // of Card
@end

@implementation CardMatchingGame

- (NSMutableArray *)cards
{
    if (!_cards) _cards = [[NSMutableArray alloc] init];
    return _cards;
}

- (instancetype)initWithCardCount:(NSUInteger)count
                           usingDeck:(Deck *)deck
{
    self = [super init]; // super's designated initializer
    if (self) {
        for (int i = 0; i < count; i++) {
            Card *card = [deck drawRandomCard];
            if (card) {
                [self.cards addObject:card];
            } else {
                self = nil;
                break;
            }
        }
    }
    return self;
}

- (Card *)cardAtIndex:(NSUInteger)index
{
    return self.cards[index];
}

@end

```

cardAtIndex: is easy.

```

Matchismo
  2 targets, iOS SDK 7.0
    Matchismo
      Main.storyboard
      CardGameViewController.h
      CardGameViewController.m
    Model
      CardMatchingGame.h
      CardMatchingGame.m
      Card.h
      Card.m
      PlayingCard.h
      PlayingCard.m
      Deck.h
      Deck.m
      PlayingCardDeck.h
      PlayingCardDeck.m
      Images.xcassets
      Supporting Files
      MatchismoTests
      Frameworks
      Products

```

```

// CardMatchingGame.h
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
                           usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end

```

```

#import "CardMatchingGame.h"

@interface CardMatchingGame()
@property (nonatomic, readwrite) NSInteger score;
@property (nonatomic, strong) NSMutableArray *cards; // of Card
@end

@implementation CardMatchingGame

- (NSMutableArray *)cards
{
    if (!_cards) _cards = [[NSMutableArray alloc] init];
    return _cards;
}

- (instancetype)initWithCardCount:(NSUInteger)count
                           usingDeck:(Deck *)deck
{
    self = [super init]; // super's designated initializer
    if (self) {
        for (int i = 0; i < count; i++) {
            Card *card = [deck drawRandomCard];
            if (card) {
                [self.cards addObject:card];
            } else {
                self = nil;
                break;
            }
        }
    }
    return self;
}

- (Card *)cardAtIndex:(NSUInteger)index
{
    return (index<[self.cards count]) ? self.cards[index] : nil;
}

@end

```

But let's be sure to check to be sure
the argument is not out of bounds.

```
// CardMatchingGame.h
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
                           usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end
```

Next comes `chooseCardAtIndex:`.
This contains the primary logic of this class.
We'll make some space to work here.

```
return self;
}

- (Card *)cardAtIndex:(NSUInteger)index
{
    return (index<[self.cards count]) ? self.cards[index] : nil;
}

- (void)chooseCardAtIndex:(NSUInteger)index
{
    // Implementation goes here
}

}
```

```
// CardMatchingGame.h
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
                           usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end

return self;
}

- (Card *)cardAtIndex:(NSUInteger)index
{
    return (index<[self.cards count]) ? self.cards[index] : nil;
}

- (void)chooseCardAtIndex:(NSUInteger)index
{
    Card *card = [self cardAtIndex:index];
}
```

Let's start by putting the card to be chosen into a local variable called card.

```
// CardMatchingGame.h
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
                           usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end

return self;
}

- (Card *)cardAtIndex:(NSUInteger)index
{
    return (index<[self.cards count]) ? self.cards[index] : nil;
}

- (void)chooseCardAtIndex:(NSUInteger)index
{
    Card *card = [self cardAtIndex:index];

    if (!card.isMatched) {

        // ...
    }
}

@end
```

We will only allow unmatched cards to be chosen
(i.e. once a card is matched, it's “out of the game”).

```
// CardMatchingGame.h
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
    usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end
```

If the card is already chosen, we'll "unchoose" it (so really this method is more like "toggle chosen state of card").

return self;

- (Card *)cardAtIndex:(NSUInteger)index

{

return (index<[self.cards count]) ? self.cards[index] : nil;

}

- (void)chooseCardAtIndex:(NSUInteger)index

{

Card *card = [self cardAtIndex:index];

if (!card.isMatched) {

if (card.isChosen) {

card.chosen = NO;

} else {

// match against other chosen cards

card.chosen = YES;

}

}

}

}

@end

Otherwise, it is being chosen and so we need to match it against other chosen cards.

And then we'll mark it as chosen.

```

Matchismo
2 targets, iOS SDK 7.0
Matchismo
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Model
CardMatchingGame.h
CardMatchingGame.m
Card.h
Card.m
PlayingCard.h
PlayingCard.m
Deck.h
Deck.m
PlayingCardDeck.h
PlayingCardDeck.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products

```

```

// CardMatchingGame.h
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
                           usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end

```

```

return self;
}

- (Card *)cardAtIndex:(NSUInteger)index
{
    return (index<[self.cards count]) ? self.cards[index] : nil;
}

- (void)chooseCardAtIndex:(NSUInteger)index
{
    Card *card = [self cardAtIndex:index];

    if (!card.isMatched) {
        if (card.isChosen) {
            card.chosen = NO;
        } else {
            // match against other chosen cards
            for (Card *otherCard in self.cards) {
                if (otherCard.isChosen && !otherCard.isMatched) {

```

So we'll just iterate through all the cards in the game, looking for ones that are unmatched and already chosen.

```

}
    }
    card.chosen = YES;
}
}

@end

```

Matchismo
2 targets, iOS SDK 7.0

Matchismo

- Main.storyboard
- CardGameViewController.h
- CardGameViewController.m
- Model

 - CardMatchingGame.h
 - CardMatchingGame.m
 - Card.h
 - Card.m
 - PlayingCard.h
 - PlayingCard.m
 - Deck.h
 - Deck.m
 - PlayingCardDeck.h
 - PlayingCardDeck.m

Images.xcassets

Supporting Files

MatchismoTests

Frameworks

Products

```

// CardMatchingGame.h
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
    usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end

```

If we find another chosen, unmatched card, we check to see if it matches the just chosen card using Card's match: method.

```

return self;
}

- (Card *)cardAtIndex:(NSUInteger)index
{
    return [self.cards objectAtIndex:index];
}

- (void)chooseCardAtIndex:(NSUInteger)index
{
    Card *card = [self cardAtIndex:index];

    if (!card.isMatched) {
        if (card.isChosen) {
            card.chosen = NO;
        } else {
            // match against other chosen cards
            for (Card *otherCard in self.cards) {
                if (otherCard.isChosen & !otherCard.isMatched) {
                    int matchScore = [card match:@[otherCard]];
                    if (matchScore > card.matchScore) {
                        card.matchScore = matchScore;
                    }
                }
            }
            card.chosen = YES;
        }
    }
}

```

Yes, it is perfectly legal to use the `@[]` array creation syntax here!

Remember that match: returns how good a match it was (zero if not a match).

match: takes an `NSArray` of other cards in case a subclass can match multiple cards. Since our matching game is only a 2-card matching game, we just create an array with one card in it.

Matchismo
2 targets, iOS SDK 7.0

Matchismo

- Main.storyboard
- CardGameViewController.h
- CardGameViewController.m

Model

- CardMatchingGame.h
- CardMatchingGame.m
- Card.h
- Card.m
- PlayingCard.h
- PlayingCard.m
- Deck.h
- Deck.m
- PlayingCardDeck.h
- PlayingCardDeck.m

Images.xcassets

Supporting Files

MatchismoTests

Frameworks

Products

```
// CardMatchingGame.h
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

// This could just as easily be a #define.

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
    usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end
```

If there's a match (of any kind), bump our score!

Otherwise, impose a penalty for choosing mismatching cards.

```
return self;
}

- (Card *)cardAtIndex:(NSUInteger)index
{
    return (index<[self.cards count]) ? self.cards[index] : nil;
}

static const int MISMATCH_PENALTY = 2;

- (void)chooseCardAtIndex:(NSUInteger)index
{
    Card *card = [self cardAtIndex:index];

    if (!card.isMatched) {
        if (card.isChosen) {
            card.chosen = NO;
        } else {
            // match against other chosen cards
            for (Card *otherCard in self.cards) {
                if (otherCard.isChosen && !otherCard.isMatched) {
                    int matchScore = [card match:@[otherCard]];
                    if (matchScore) {
                        self.score += matchScore;
                    } else {
                        self.score -= MISMATCH_PENALTY;
                    }
                }
            }
            card.chosen = YES;
        }
    }
}
```

```
// CardMatchingGame.h
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
                           usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end
```

```
return self;
}

- (Card *)cardAtIndex:(NSUInteger)index
{
    return (index<[self.cards count]) ? self.cards[index] : nil;
}

static const int MISMATCH_PENALTY = 2;
static const int MATCH_BONUS = 4;

- (void)chooseCardAtIndex:(NSUInteger)index
{
    Card *card = [self cardAtIndex:index];

    if (!card.isMatched) {
        if (card.isChosen) {
            card.chosen = NO;
        } else {
            // match against other chosen cards
            for (Card *otherCard in self.cards) {
                if (otherCard.isChosen && !otherCard.isMatched) {
                    int matchScore = [card match:@[otherCard]];
                    if (matchScore) {
                        self.score += matchScore * MATCH_BONUS;
                    } else {
                        self.score -= MISMATCH_PENALTY;
                    }
                }
            }
            card.chosen = YES;
        }
    }
}
```

We can even give a bonus to matches if we want.

```

Matchismo
2 targets, iOS SDK 7.0
Matchismo
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Model
CardMatchingGame.h
CardMatchingGame.m
Card.h
Card.m
PlayingCard.h
PlayingCard.m
Deck.h
Deck.m
PlayingCardDeck.h
PlayingCardDeck.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products

```

```

// CardMatchingGame.h
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
    usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end

```

```

return self;
}

- (Card *)cardAtIndex:(NSUInteger)index
{
    return (index<[self.cards count]) ? self.cards[index] : nil;
}

static const int MISMATCH_PENALTY = 2;
static const int MATCH_BONUS = 4;

- (void)chooseCardAtIndex:(NSUInteger)index
{
    Card *card = [self cardAtIndex:index];

    if (!card.isMatched) {
        if (card.isChosen) {
            card.chosen = NO;
        } else {
            // match against other chosen cards
            for (Card *otherCard in self.cards) {
                if (otherCard.isChosen && !otherCard.isMatched) {
                    int matchScore = [card match:@[otherCard]];
                    if (matchScore) {
                        self.score += matchScore * MATCH_BONUS;
                    }
                } else {
                    self.score -= MISMATCH_PENALTY;
                    otherCard.chosen = NO;
                }
            }
        }
    }
    card.chosen = YES;
}

```

If it's a mismatch, “unchoose” the mismatching other card.

If we allowed more than 2 card matches, we might not necessarily do this. One could also imagine unchoosing *both* mismatching cards, but that would require a delay unchoosing the second one and we haven't really learned how to animate delays in the UI like that, thus this approach.

```

Matchismo
2 targets, iOS SDK 7.0
Matchismo
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Model
CardMatchingGame.h
CardMatchingGame.m
Card.h
Card.m
PlayingCard.h
PlayingCard.m
Deck.h
Deck.m
PlayingCardDeck.h
PlayingCardDeck.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products

```

```

// CardMatchingGame.h
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
                           usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end

```

If it's a match, mark both matching cards as **matched**.

```

return self;
}

- (Card *)cardAtIndex:(NSUInteger)index
{
    return (index<[self.cards count]) ? self.cards[index] : nil;
}

static const int MISMATCH_PENALTY = 2;
static const int MATCH_BONUS = 4;

- (void)chooseCardAtIndex:(NSUInteger)index
{
    Card *card = [self cardAtIndex:index];

    if (!card.isMatched) {
        if (card.isChosen) {
            card.chosen = NO;
        } else {
            // match against other chosen cards
            for (Card *otherCard in self.cards) {
                if (otherCard.isChosen && !otherCard.isMatched) {
                    int matchScore = [card match:@[otherCard]];
                    if (matchScore) {
                        self.score += matchScore * MATCH_BONUS;
                        otherCard.matched = YES;
                        card.matched = YES;
                    } else {
                        self.score -= MISMATCH_PENALTY;
                        otherCard.chosen = NO;
                    }
                }
            }
            card.chosen = YES;
        }
    }
}

```

Stanford CS193p

Fall 2013

Matchismo
2 targets, iOS SDK 7.0

Matchismo

- Main.storyboard
- CardGameViewController.h
- CardGameViewController.m
- Model
 - CardMatchingGame.h
 - CardMatchingGame.m
 - Card.h
 - Card.m
 - PlayingCard.h
 - PlayingCard.m
 - Deck.h
 - Deck.m
 - PlayingCardDeck.h
 - PlayingCardDeck.m
- Images.xcassets
- Supporting Files
- MatchismoTests
- Frameworks
- Products

Since we only allow matching 2 cards and we've found 2 chosen cards at this point, we can **break** out of the **for** loop.

In next week's homework (not this week's), you'll be supporting matching more than 2 cards, so you'll be doing this all slightly differently.

```
// CardMatchingGame.h
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
                           usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end

return self;
}

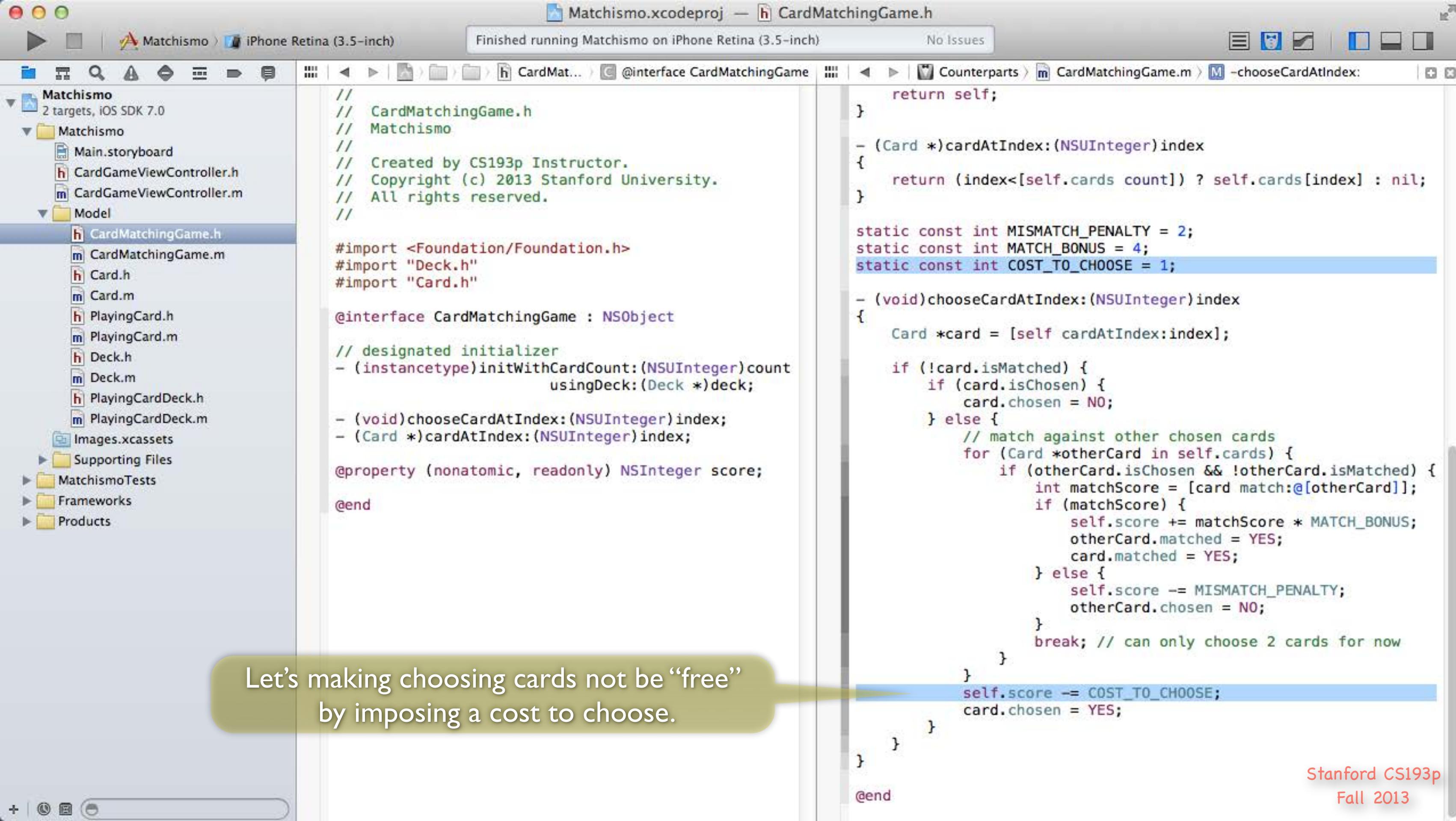
- (Card *)cardAtIndex:(NSUInteger)index
{
    return (index<[self.cards count]) ? self.cards[index] : nil;
}

static const int MISMATCH_PENALTY = 2;
static const int MATCH_BONUS = 4;

- (void)chooseCardAtIndex:(NSUInteger)index
{
    Card *card = [self cardAtIndex:index];

    if (!card.isMatched) {
        if (card.isChosen) {
            card.chosen = NO;
        } else {
            // match against other chosen cards
            for (Card *otherCard in self.cards) {
                if (otherCard.isChosen && !otherCard.isMatched) {
                    int matchScore = [card match:@[otherCard]];
                    if (matchScore) {
                        self.score += matchScore * MATCH_BONUS;
                        otherCard.matched = YES;
                        card.matched = YES;
                    } else {
                        self.score -= MISMATCH_PENALTY;
                        otherCard.chosen = NO;
                    }
                }
            }
            break; // can only choose 2 cards for now
        }
    }
    card.chosen = YES;
}

@end
```



Let's making choosing cards not be “free” by imposing a cost to choose.

```

Matchismo
2 targets, iOS SDK 7.0
Matchismo
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Model
CardMatchingGame.h
CardMatchingGame.m
Card.h
Card.m
PlayingCard.h
PlayingCard.m
Deck.h
Deck.m
PlayingCardDeck.h
PlayingCardDeck.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
CardMatchingGame.h
Matchismo
Created by CS193p Instructor.
Copyright (c) 2013 Stanford University.
All rights reserved.
That's it!
Pretty simple really.
But there's one more change we'll want to make to our Model.
Specifically, PlayingCard's match: algorithm.

@interface CardMatchingGame : NSObject
// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
    usingDeck:(Deck *)deck;
- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;
@property (nonatomic, readonly) NSInteger score;
@end

return self;
}
- (Card *)cardAtIndex:(NSUInteger)index
{
    return (index<[self.cards count]) ? self.cards[index] : nil;
}

static const int MISMATCH_PENALTY = 2;
static const int MATCH_BONUS = 4;
static const int COST_TO_CHOOSE = 1;

- (void)chooseCardAtIndex:(NSUInteger)index
{
    Card *card = [self cardAtIndex:index];
    if (!card.isMatched) {
        if (card.isChosen) {
            card.chosen = NO;
        } else {
            // match against other chosen cards
            for (Card *otherCard in self.cards) {
                if (otherCard.isChosen && !otherCard.isMatched) {
                    int matchScore = [card match:@[otherCard]];
                    if (matchScore) {
                        self.score += matchScore * MATCH_BONUS;
                        otherCard.matched = YES;
                        card.matched = YES;
                    } else {
                        self.score -= MISMATCH_PENALTY;
                        otherCard.chosen = NO;
                    }
                }
            }
            self.score -= COST_TO_CHOOSE;
            card.chosen = YES;
        }
    }
}
@end

```

Matchismo.xcodeproj — Card.h

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Matchismo 2 targets, iOS SDK 7.0

Main.storyboard CardGameViewController.h CardGameViewController.m Model CardMatchingGame.h CardMatchingGame.m Card.h Card.m PlayingCard.h PlayingCard.m Deck.h Deck.m PlayingCardDeck.h PlayingCardDeck.m Images.xcassets Supporting Files MatchismoTests Frameworks Products

```
// Card.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import "PlayingCard.h"

@interface Card : NSObject

@property (strong, nonatomic) NSString *contents;
@property (nonatomic, getter=isChosen) BOOL chosen;
@property (nonatomic, getter=isMatched) BOOL matched;

- (int)match:(NSArray *)otherCards;
@end
```

Click on Card in the Navigator so that we remind ourselves what its match: algorithm is.

```
// Card.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import "Card.h"

@implementation Card

- (int)match:(NSArray *)otherCards
{
    int score = 0;

    for (Card *card in otherCards) {
        if ([card.contents isEqualToString:self.contents]) {
            score = 1;
        }
    }

    return score;
}

@end
```

Card matches only if the cards are exactly the same (that is to say, their contents @property values are equal). PlayingCards should match if the suit and/or rank is the same. Let's go to PlayingCard and override Card's implementation of match: to make this so.

Matchismo > iPhone Retina (3.5-inch) Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Matchismo
2 targets, iOS SDK 7.0

Matchismo
Main.storyboard
CardGameViewController.h
CardGameViewController.m

Model
CardMatchingGame.h
CardMatchingGame.m
Card.h
Card.m
PlayingCard.h
PlayingCard.m
Deck.h
Deck.m
PlayingCardDeck.h
PlayingCardDeck.m

Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products

```
// PlayingCard.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import "Card.h"

@interface PlayingCard : Card

@property (nonatomic) NSString *contents;
@property (nonatomic) NSUInteger rank;

+ (NSArray *)validSuits;
+ (NSUInteger)maxRank;

@end
```

Click to switch to PlayingCard.

```
// PlayingCard.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import "PlayingCard.h"

@implementation PlayingCard

- (NSString *)contents
{
    NSArray *rankStrings = [PlayingCard rankStrings];
    return [rankStrings[self.rank] stringByAppendingString:self.suit];
}

@synthesize suit = _suit;

+ (NSArray *)validSuits
{
    return @[@"♠", @"♦", @"♥", @"♦"];
}

- (void)setSuit:(NSString *)suit
{
    if ([[PlayingCard validSuits] containsObject:suit]) {
        _suit = suit;
    }
}

- (NSString *)suit
{
    return _suit ? _suit : @"?";
}

+ (NSArray *)rankStrings
{
    return @[@"?", @"A", @"2", @"3", @"4", @"5", @"6",
             @"7", @"8", @"9", @"10", @"J", @"Q", @"K"];
}
```

Add an implementation for match::

```

// PlayingCard.h
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import "Card.h"

@interface PlayingCard : Card

@property (strong, nonatomic) NSString *suit;
@property (nonatomic) NSUInteger rank;

+ (NSArray *)validSuits;
+ (NSUInteger)maxRank;

@end

```

Note that even though PlayingCard is overriding its superclass's implementation of a method (`match:`), it is not required to redeclare `match:` in its header file. Generally we do not redeclare overridden methods.

```

// PlayingCard.m
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import "PlayingCard.h"

@implementation PlayingCard

- (int)match:(NSArray *)otherCards
{
    int score = 0;

    return score;
}

- (NSString *)contents
{
    NSArray *rankStrings = [PlayingCard rankStrings];
    return [rankStrings[self.rank] stringByAppendingString:self.suit];
}

@synthesize suit = _suit;

+ (NSArray *)validSuits
{
    return @[@"♦",@"♠",@"♥",@"♦"];
}

```

Often a subclass's implementation of a method will call its superclass's implementation by invoking `super` (e.g. `[super match:...]`), but `PlayingCard` has its own, standalone implementation of this method and thus does not need to call `super`'s implementation.

Matchismo > iPhone Retina (3.5-inch) Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Matchismo
2 targets, iOS SDK 7.0

Matchismo
Main.storyboard
CardGameViewController.h
CardGameViewController.m

Model
CardMatchingGame.h
CardMatchingGame.m
Card.h
Card.m
PlayingCard.h
PlayingCard.m
Deck.h
Deck.m
PlayingCardDeck.h
PlayingCardDeck.m

Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products

```
// PlayingCard.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import "Card.h"

@interface PlayingCard : Card

@property (strong, nonatomic) NSString *suit;
@property (nonatomic) NSUInteger rank;

+ (NSArray *)validSuits;
+ (NSUInteger)maxRank;

@end
```

// PlayingCard.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import "PlayingCard.h"

@implementation PlayingCard

- (int)match:(NSArray *)otherCards
{
 int score = 0;

 if ([otherCards count] == 1) {

 }

 return score;
}

- (NSString *)contents
{
 NSArray *rankStrings = [PlayingCard rankStrings];
 return [rankStrings[self.rank] stringByAppendingString:self.suit];
}

@synthesize suit = _suit;

+ (NSArray *)validSuits
{
 return @[@"♠", @"♦", @"♥", @"♦"];
}

First, we will only match a single other card (next week's homework assignment will have to do better than this).

```
// PlayingCard.h
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import "Card.h"

@interface PlayingCard : Card

@property (strong, nonatomic) NSString *suit;
@property (nonatomic) NSUInteger rank;

+ (NSArray *)validSuits;
+ (NSUInteger)maxRank;

@end
```

```
// PlayingCard.m
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import "PlayingCard.h"

@implementation PlayingCard

- (int)match:(NSArray *)otherCards
{
    int score = 0;

    if ([otherCards count] == 1) {
        PlayingCard *otherCard = [otherCards firstObject];
    }
}

return score;      [array objectAtIndex:0]
}

- (NSString *)contents
{
    NSArray *rankStrings = [PlayingCard rankStrings];
    return [rankStrings[self.rank] stringByAppendingString:suit];
}

@synthesize suit = _suit;

+ (NSArray *)validSuits
{
    return @[@"♠",@"♦",@"♥",@"♦"];
}
```

Let's get the card in the array
(there will only be one card in the array
if we got this far).

firstObject is an NSArray method. It is just like [array objectAtIndex:0]
except that it will not crash if the array is empty
(it will just return nil).
Convenient.

```
// PlayingCard.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import "Card.h"

@interface PlayingCard : Card

@property (strong, nonatomic) NSString *suit;
@property (nonatomic) NSUInteger rank;

+ (NSArray *)validSuits;
+ (NSUInteger)maxRank;

@end
```

```
// PlayingCard.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import "PlayingCard.h"

@implementation PlayingCard

- (int)match:(NSArray *)otherCards
{
    int score = 0;

    if ([otherCards count] == 1) {
        PlayingCard *otherCard = [otherCards firstObject];
        if (otherCard.rank == self.rank) {
            score = 4;
        }
    }

    return score;
}

- (NSString *)contents
{
    NSArray *rankStrings = [PlayingCard rankStrings];
    return [rankStrings[self.rank] stringByAppendingString:suit];
}

@synthesize suit = _suit;

+ (NSArray *)validSuits
{
    return @[@"♠", @"♦", @"♥", @"♦"];
}
```

Give 4 points for matching the rank ...

Matchismo > iPhone Retina (3.5-inch) Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

// PlayingCard.h
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import "Card.h"

@interface PlayingCard : Card

@property (strong, nonatomic) NSString *suit;
@property (nonatomic) NSUInteger rank;

+ (NSArray *)validSuits;
+ (NSUInteger)maxRank;

@end

// PlayingCard.m
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.

#import "PlayingCard.h"

@implementation PlayingCard

- (int)match:(NSArray *)otherCards
{
 int score = 0;

 if ([otherCards count] == 1) {
 PlayingCard *otherCard = [otherCards firstObject];
 if (otherCard.rank == self.rank) {
 score = 4;
 } **else if ([otherCard.suit isEqualToString:self.suit])** {
 score = 1;
 }
 }

 return score;
}

- (NSString *)contents
{
 NSArray *rankStrings = [PlayingCard rankStrings];
 return [rankStrings [self.rank] stringByAppendingString:self.suit];
}

@synthesize suit = _suit;

+ (NSArray *)validSuits
{
 return @[@"♦",@"♥",@"♣",@"♠"];
}

And only 1 point for matching the suit ...

There are only 3 cards that will match a given card's rank, but 12 which will match its suit, so this makes some sense.

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Matchismo 2 targets, iOS SDK 7.0

Main.storyboard Card Game View Automatic CardGameViewController.m -touchCardButton:

#import "CardGameViewController.h"
#import "PlayingCardDeck.h"

@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@end

@implementation CardGameViewController

- (Deck *)deck
{
 if (!_deck) _deck = [self createDeck];
 return _deck;
}

- (Deck *)createDeck
{
 return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
 if ([sender.currentTitle length]) {
 [sender setBackgroundImage:[UIImage imageNamed:@"cardback"]
 forState:UIControlStateNormal];
 [sender setTitle:@"" forState:UIControlStateNormal];
 } else {
 Card *randomCard = [self.deck drawRandomCard];
 if (randomCard) {
 [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"]
 forState:UIControlStateNormal];
 [sender setTitle:randomCard.contents forState:
 UIControlStateNormal];
 }
 }
 self.flipCount++;
}

- (void)setFlipCount:(int)flipCount
{
 _flipCount = flipCount;

That's it for our Model.
Back to our View and Controller.

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Automatic CardGameViewController.m -touchCardButton:

```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"

@interface CardGameViewController ()  
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;  
@property (nonatomic) int flipCount;  
@property (strong, nonatomic) Deck *deck;  
@end

@implementation CardGameViewController

- (Deck *)deck  
{  
    if (!_deck) _deck = [self createDeck];  
    return _deck;  
}  
- (IBAction)touchCardButton:(UIButton *)sender  
{  
    if ([sender.currentTitle length]) {  
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];  
        [sender setTitle:@"" forState:UIControlStateNormal];  
    } else {  
        Card *randomCard = [self.deck drawRandomCard];  
        if (randomCard) {  
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];  
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];  
        }  
    }  
    self.flipCount++;  
}  
- (void)setFlipCount:(int)flipCount  
{  
    _flipCount = flipCount;  
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];  
    NSLog(@"flipCount changed to %d", self.flipCount);  
}
```

Close the Navigator.

All we have left to do is use our new CardMatchingGame Model in our Controller and add some UI to show the score.

Almost done.

Flips: 0

Stanford CS193p Fall 2013

The screenshot shows the Xcode IDE with the project "Matchismo" open. The main window displays the file "CardGameViewController.m". The code implements a card matching game where four cards are dealt in a 3x4 grid. A green callout bubble points to the import statement for "CardMatchingGame.h" with the text "Don't forget the #import of the CardMatchingGame.". Another green callout bubble points to the "@property" declaration for the "deck" variable with the text "We need a @property for our game Model.". The storyboard view on the left shows a green screen with a 3x4 grid of cards, each labeled with a large red letter 'S'. A label at the bottom left says "Flips: 0".

```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"

@interface CardGameViewController : UIViewController
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@property (strong, nonatomic) CardMatchingGame *game;
@end

@implementation CardGameViewController

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
    }
    self.flipCount++;
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
}

```

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Matchismo > iPhone Retina (3.5-inch)

Card G... View Automatic CardGameViewController.m -game

```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"

@interface CardGameViewController ()
```

Let's lazily instantiate it.

```
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@property (nonatomic) CardMatchingGame *game;
@end
```

```
@implementation CardGameViewController
```

- (CardMatchingGame *)game

```
{
```

if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:0 usingDeck:[self createDeck]];

return _game;

```
}
```

- (Deck *)deck

```
{
```

if (!_deck) _deck = [self createDeck];

return _deck;

```
}
```

- (Deck *)createDeck

```
{
```

return [[PlayingCardDeck alloc] init];

```
}
```

- (IBAction)touchCardButton:(UIButton *)sender

```
{
```

if ([sender.currentTitle length] == 0) {

[sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];

[sender setTitle:@"" forState:UIControlStateNormal];

```
}
```

else {

Card *randomCard = [self.deck drawRandomCard];

if (randomCard) {

[sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];

[sender setTitle:randomCard.contents forState:UIControlStateNormal];

```
}
```

Told you createDeck would come in handy!

We need the number of cards here.

How many are there?

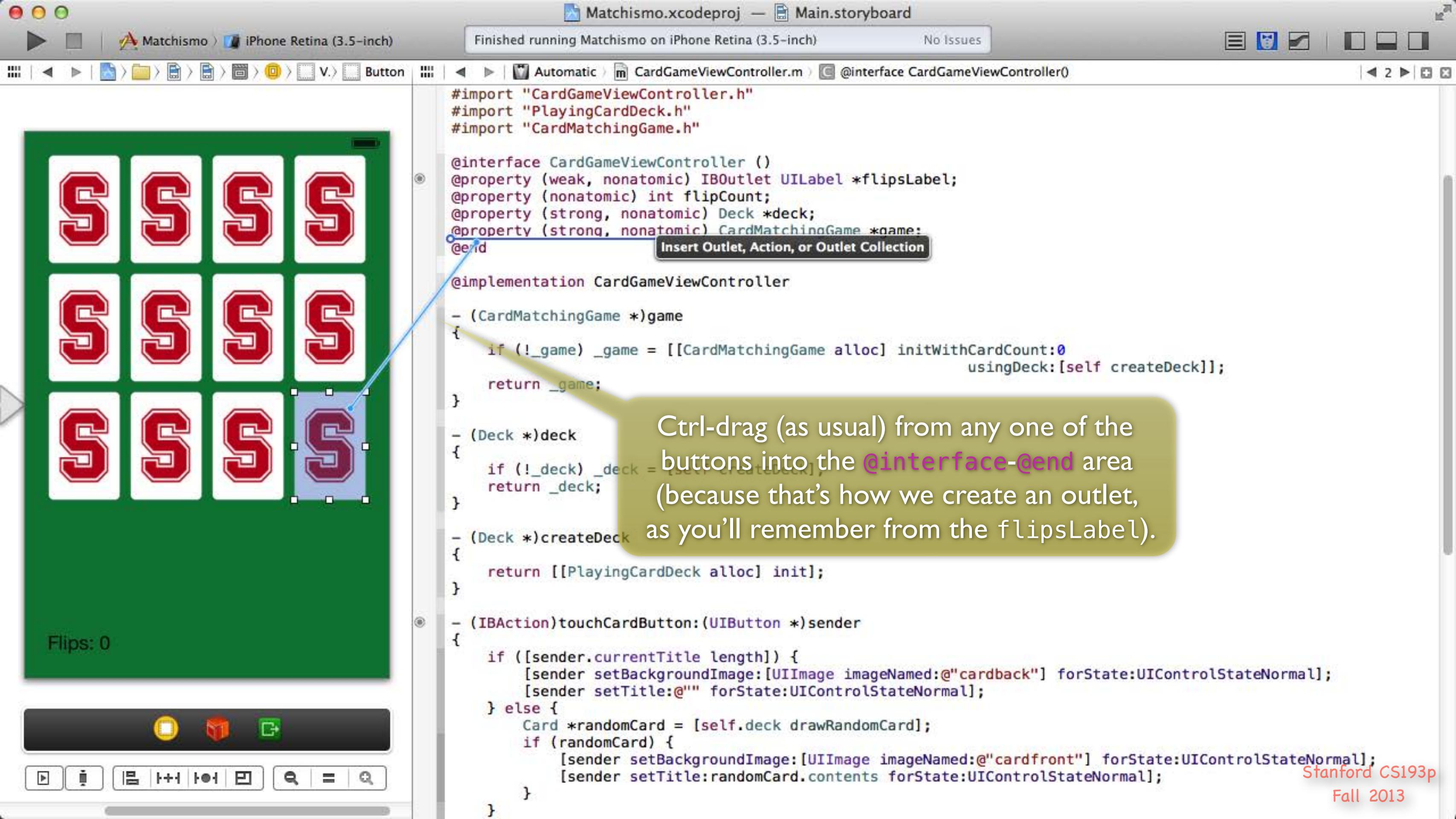
We're going to connect a new kind of outlet that points to all of the card buttons.

Instead of being a pointer to a single instance (like `flipsLabel`), it's going to be an `NSArray` (with multiple `UIButtons` in it).

Then we'll simply ask the array how many items are in it.

Stanford CS193p
Fall 2013

```
        [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
        [sender setTitle:randomCard.contents forState:UIControlStateNormal];
    }
}
```



ctrl-drag (as usual) from any one of the buttons into the @interface-@end area because that's how we create an outlet, you'll remember from the `flipsLabel`).



```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"

@interface CardGameViewController : UIViewController
@property (weak) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@property (strong, nonatomic) CardMatchingGame *game;
@end

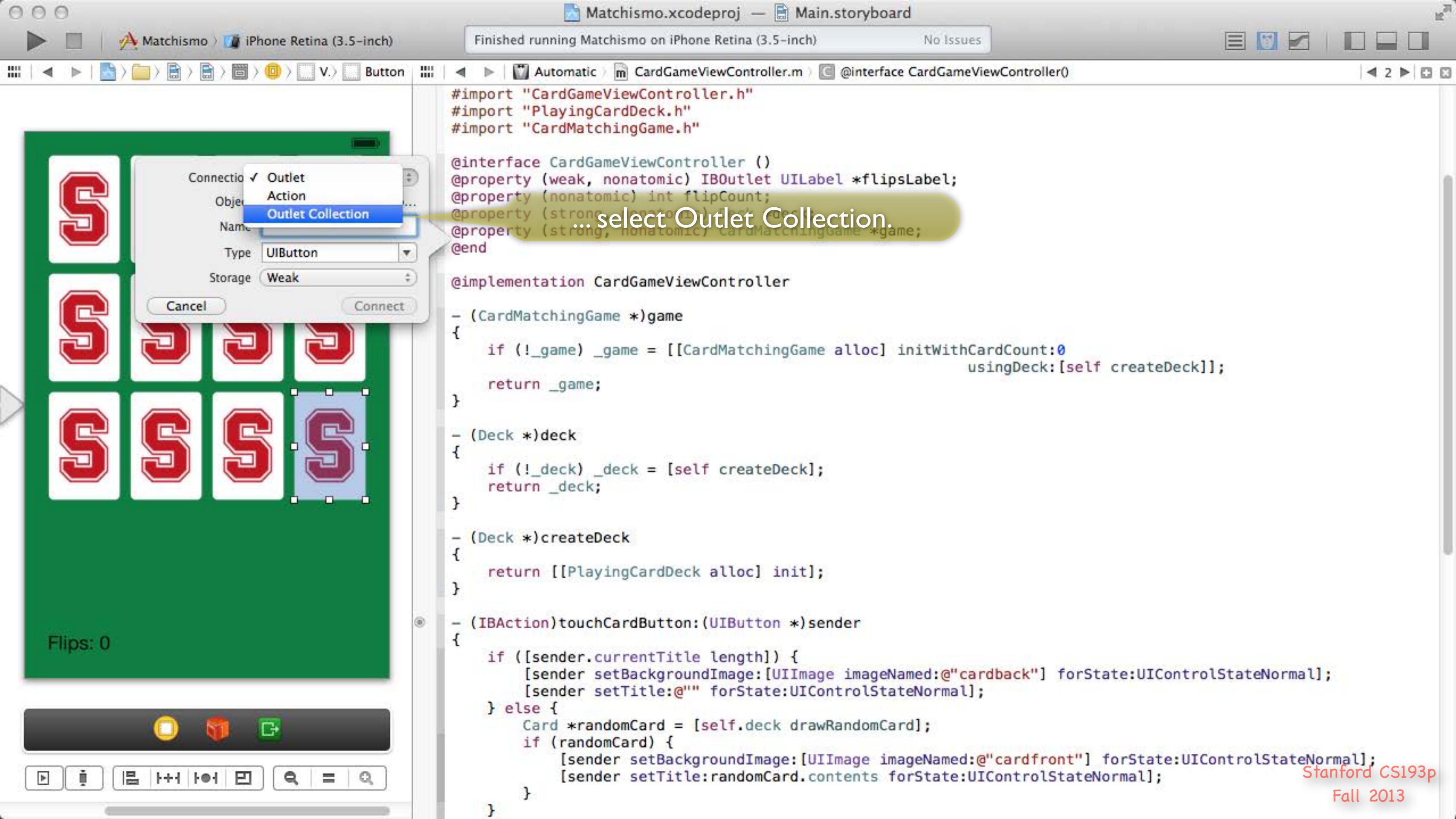
@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:0
                                                               usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
    }
}
```



Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"

@interface CardGameViewController ()

@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@property (strong, nonatomic) CardMatchingGame *game;

@end

@implementation CardGameViewController

- (CardMatchingGame *)game

{
 if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:0
 usingDeck:[self createDeck]];
 return _game;
}

- (Deck *)deck

{
 if (!_deck) _deck = [self createDeck];
 return _deck;
}

- (IBAction)touchCardButton:(UIButton *)sender

{
 if ([sender.currentTitle length]) {
 [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
 [sender setTitle:@"" forState:UIControlStateNormal];
 } else {
 Card *randomCard = [self.deck drawRandomCard];
 if (randomCard) {
 [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
 [sender setTitle:randomCard.contents forState:UIControlStateNormal];
 }
 }
}

Connection **Outlet Collection**

Object **Card Game View Co...**

Name

Type **UIButton**

Cancel Connect

Flips: 0

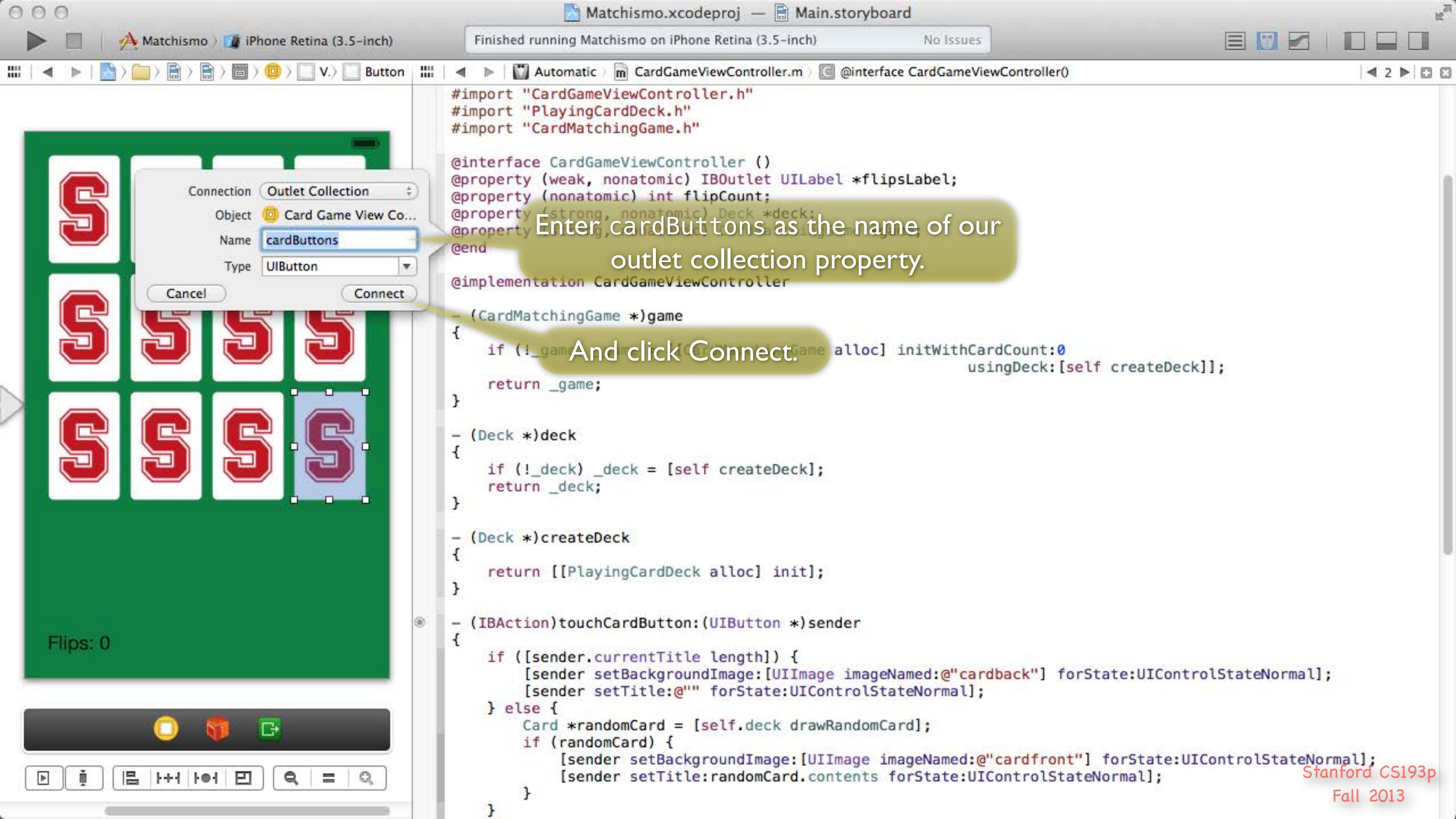
Stanford CS193p
Fall 2013

Outlet Collections are **NSArray @properties**.

This specifies that the array will contain **UIButton** instances.

This is only for Xcode's benefit only.
Objective-C doesn't let you specify the class of objects in an array.

Outlet Collection arrays are always **strong**, so Xcode has removed that option from the dialog.
While the View will point **strongly** to all of the buttons **inside** the array, it will not point to the array itself at all (only our Controller will) so our outlet needs to be **strongly** held in the heap by our Controller.



Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Automatic CardGameViewController.m @interface CardGameViewController()

```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"

@interface CardGameViewController : UIViewController
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@property (strong, nonatomic) CardMatchingGame *game;
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *cardButtons;
@end

@implementation CardGameViewController
- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:0
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
    }
}

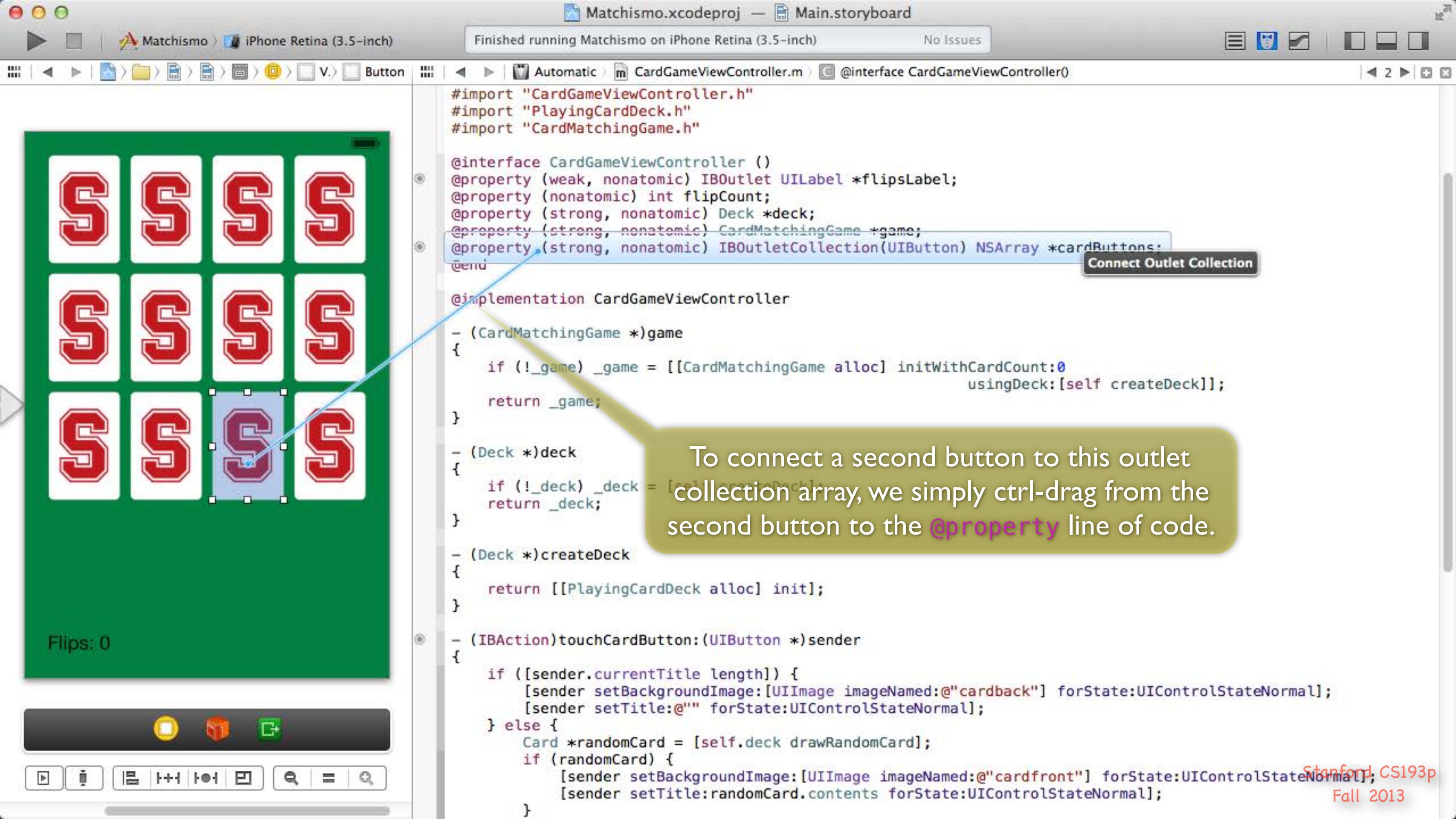
```

As expected, this outlet is an `NSArray`.

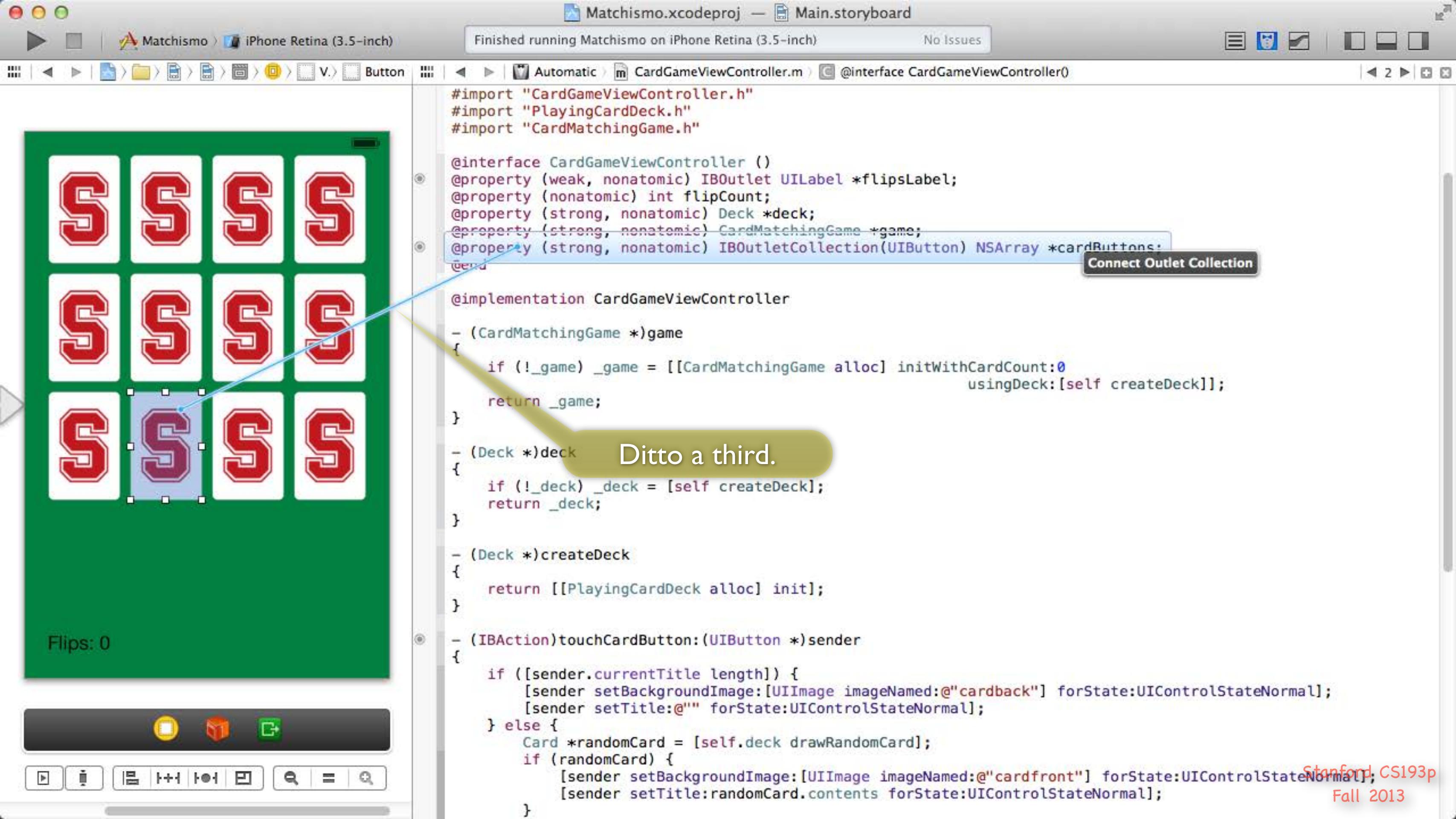
See? `strong`.

This `IBOutletConnection(UIButton)` is again just something Xcode puts in there to remember that this is an outlet not just a random `NSArray`.
The compiler ignores this.

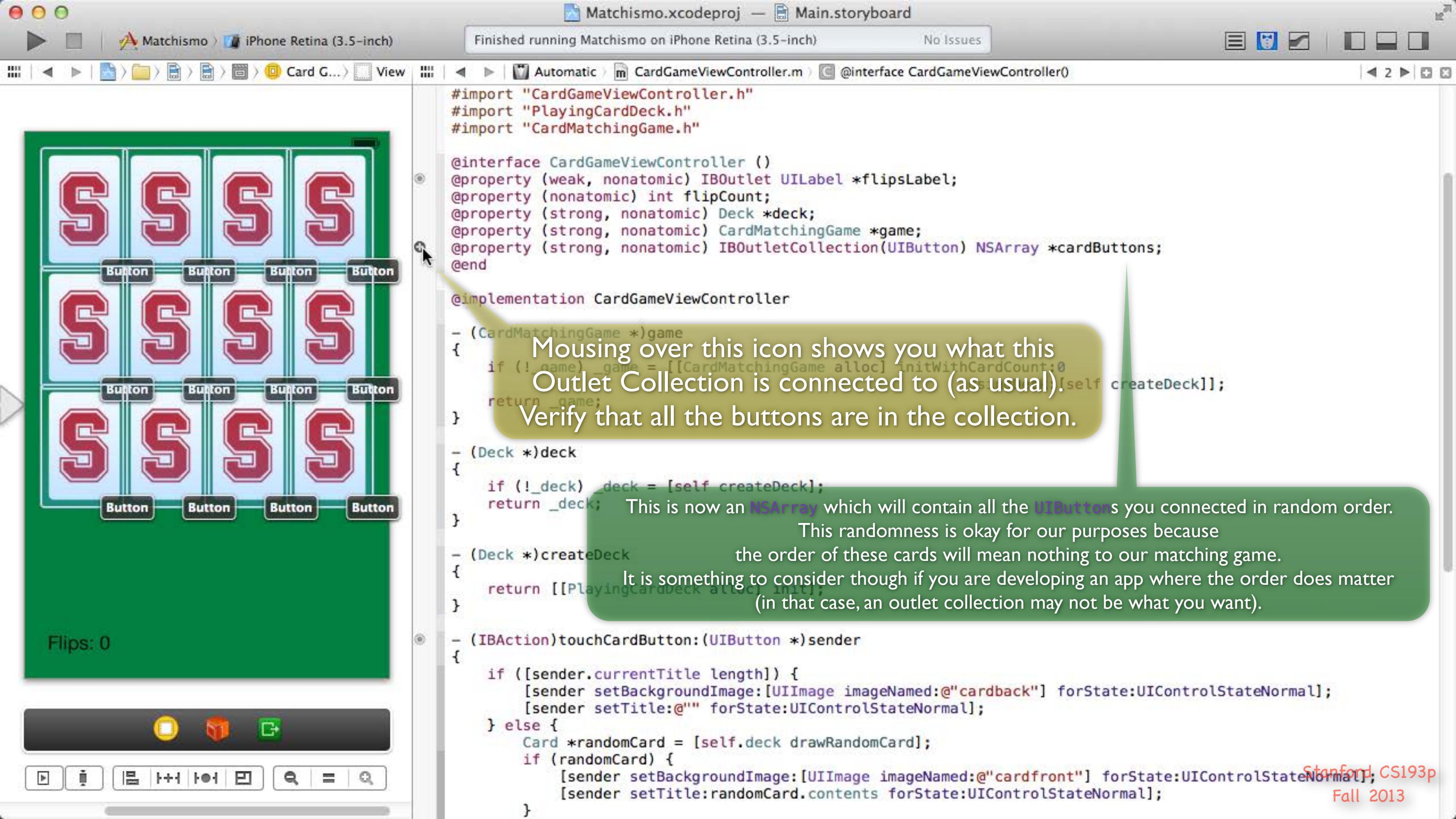
Flips: 0



To connect a second button to this outlet collection array, we simply ctrl-drag from the second button to the `@property` line of code.



to a third.



```
import "CardGameViewController.h"
import "PlayingCardDeck.h"
import "CardMatchingGame.h"

interface CardGameViewController () {
    property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
    property (nonatomic) int flipCount;
    property (strong, nonatomic) Deck *deck;
    property (strong, nonatomic) CardMatchingGame *game;
    property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *cardButtons;
}

implementation CardGameViewController

(CardMatchingGame *)game
if (!game) _game = [[CardMatchingGame alloc] initWithCardCount:0
    delegate:self cancelButtonTitle:@"Shuffle" otherButtonTitles:@"Deal"];
[self createDeck];
return _game;

(Deck *)deck
if (!_deck) _deck = [self createDeck];
return _deck; This is now an NSArray which will contain all the UIButtons you connected in random order.
This randomness is okay for our purposes because
the order of these cards will mean nothing to our matching game.

(Deck *)createDeck
return [[PlayingCardDeck alloc] init];
It is something to consider though if you are developing an app where the order does matter
(in that case, an outlet collection may not be what you want).

IBAction)touchCardButton:(UIButton *)sender
if ([sender.currentTitle length]) {
    [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
    [sender setTitle:@"" forState:UIControlStateNormal];
} else {
    Card *randomCard = [self.deck drawRandomCard];
    if (randomCard) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
        [sender setTitle:randomCard.contents forState:UIControlStateNormal];
    }
}
```

Mousing over this icon shows you what this Outlet Collection is connected to (as usual). Verify that all the buttons are in the collection.

Stanford CS1
Fall 2013

Mousing over this icon shows you what this Outlet Collection is connected to (as usual). Verify that all the buttons are in the collection.

Verify that all the buttons are in the collection.

This is now an `NSArray` which will contain all the `UIButton`s you connected in random order.

This randomness is okay for our purposes because the order of these cards will mean nothing to our matching game.

It is something to consider though if you are developing an app where the order does matter (in that case, an outlet collection may not be what you want).

```
-(IBAction)touchCardButton:(UIButton *)sender
```

```
if ([sender.currentTitle length]) {
    [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
    [sender setTitle:@"" forState:UIControlStateNormal];
} else {
    Card *randomCard = [self.deck drawRandomCard];
    if (randomCard) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
        [sender setTitle:randomCard.contents forState:UIControlStateNormal];
    }
}
```

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Card G... View Automatic CardGameViewController.m -game

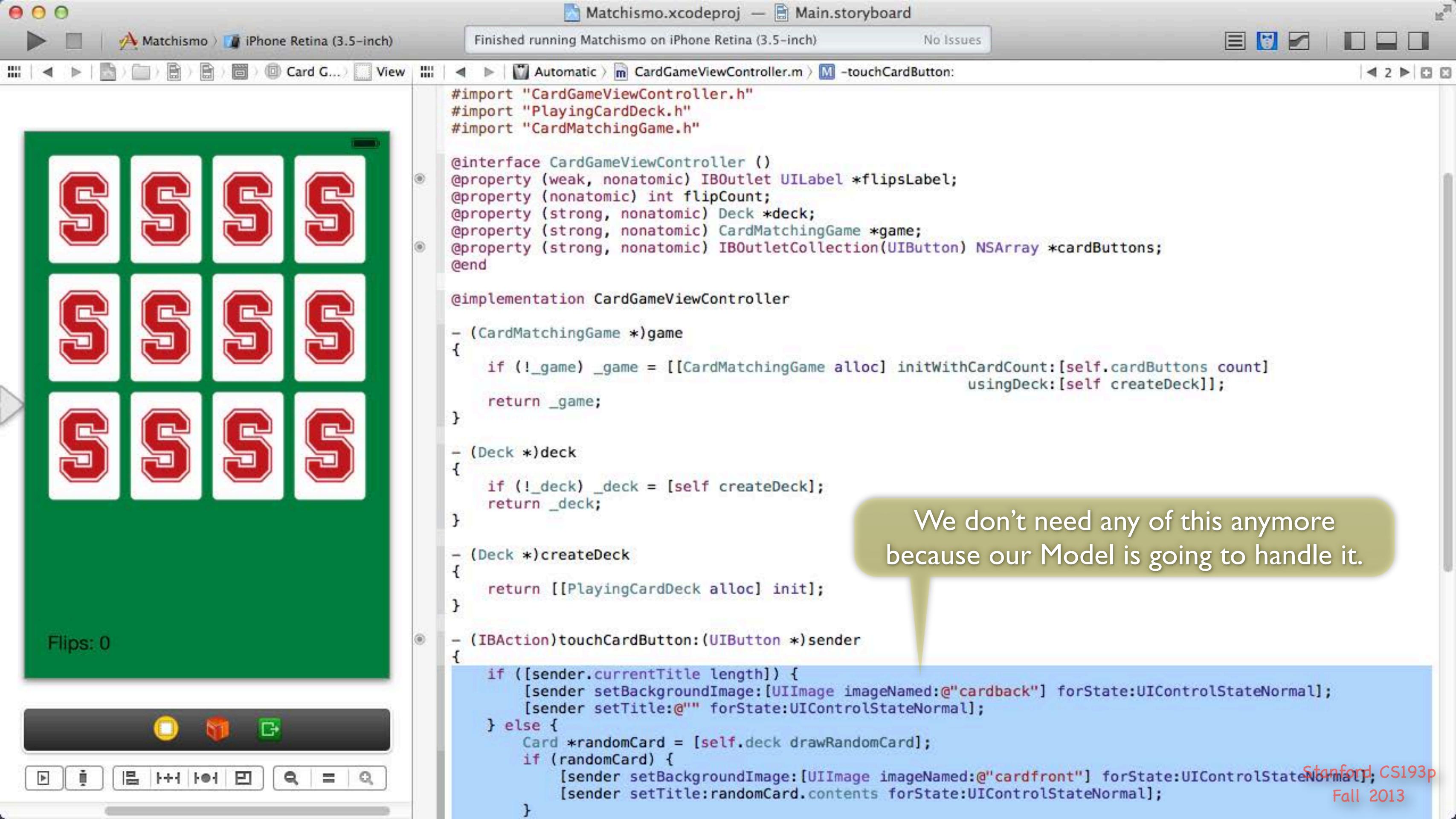
```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"

@interface CardGameViewController ()  
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;  
@property (nonatomic) int flipCount;  
@property (strong, nonatomic) Deck *deck;  
@property (strong, nonatomic) CardMatchingGame *game;  
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *cardButtons;  
@end

@implementation CardGameViewController  
- (CardMatchingGame *)game  
{  
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]  
                           usingDeck:[self createDeck]];  
    return _game;  
}  
- (Deck *)deck  
{  
    if (!_deck) _deck = [self createDeck];  
    return _deck;  
}  
- (Deck *)createDeck  
{  
    return [[PlayingCardDeck alloc] init];  
}  
- (IBAction)touchCardButton:(UIButton *)sender  
{  
    if ([sender.currentTitle length]) {  
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];  
        [sender setTitle:@"" forState:UIControlStateNormal];  
    } else {  
        Card *randomCard = [self.deck drawRandomCard];  
        if (randomCard) {  
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];  
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];  
        }  
    }  
}
```

Now we know how many cards there are in the UI.

Stanford CS193p; Fall 2013



Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Automatic CardGameViewController.m -touchCardButton:

```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"

@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@property (strong, nonatomic) CardMatchingGame *game;
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *cardButtons;
@end

@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
    }
}
```

We don't need any of this anymore because our Model is going to handle it.

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Automatic CardGameViewController.m -touchCardButton:

```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"

@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@property (strong, nonatomic) CardMatchingGame *game;
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *cardButtons;
@end

@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    self.flipCount++;
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}
```

Flips: 0

So delete it.

Stanford CS193p
Fall 2013

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Card Game View Controller

```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"

@interface CardGameViewController : UIViewController
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@property (strong, nonatomic) CardMatchingGame *game;
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *cardButtons;
@end

@implementation CardGameViewController
- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                       usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

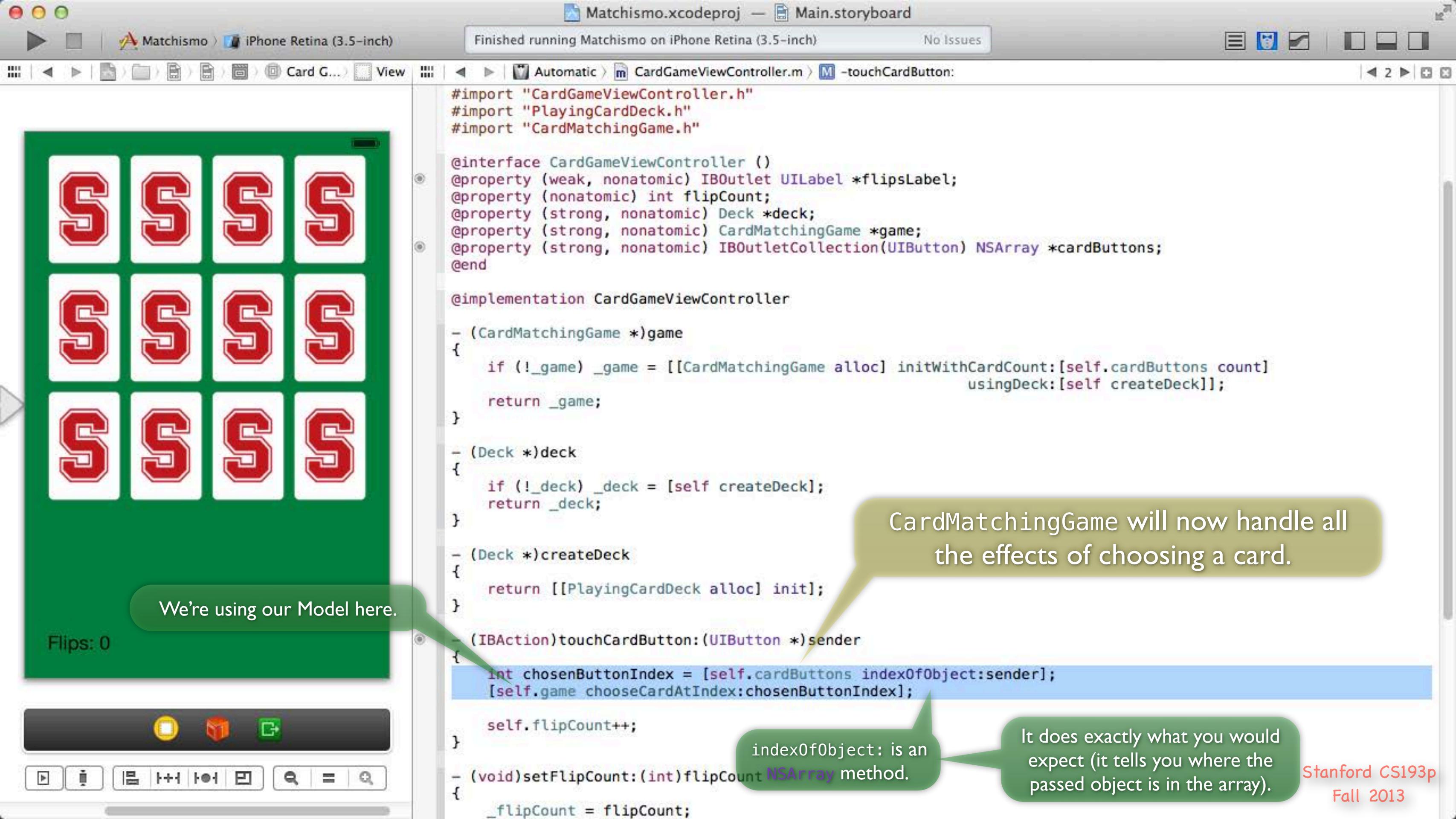
- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [self.game chooseCardAtIndex:chosenButtonIndex];

    self.flipCount++;
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
}
```

We're using our Model here.

Flips: 0



CardMatchingGame will now handle all the effects of choosing a card.

indexOfObject: is an NSArray method.

It does exactly what you would expect (it tells you where the passed object is in the array).

Stanford CS193p Fall 2013

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"

@interface CardGameViewController : UIViewController
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@property (strong, nonatomic) CardMatchingGame *game;
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *cardButtons;
@end

@implementation CardGameViewController
- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                     usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [self.game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
    self.flipCount++;
}

- (void)updateUI { }

- (void)setFlipCount:(int)flipCount
{
}
```

However, our Controller must still do its job of interpreting the Model into the View. We'll implement `updateUI` in a moment.

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch)

No Issues

```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"

@interface CardGameViewController ()  
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;  
@property (nonatomic) int flipCount;  
@property (strong, nonatomic) Deck *deck;  
@property (strong, nonatomic) CardMatchingGame *game;  
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *cardButtons;  
@end

@implementation CardGameViewController

- (CardMatchingGame *)game  
{  
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]  
                           usingDeck:[self createDeck]];  
    return _game;  
}

- (Deck *)deck  
{  
    if (!_deck) _deck = [self createDeck];  
    return _deck;  
}

- (Deck *)createDeck  
{  
    return [[PlayingCardDeck alloc] init];  
}

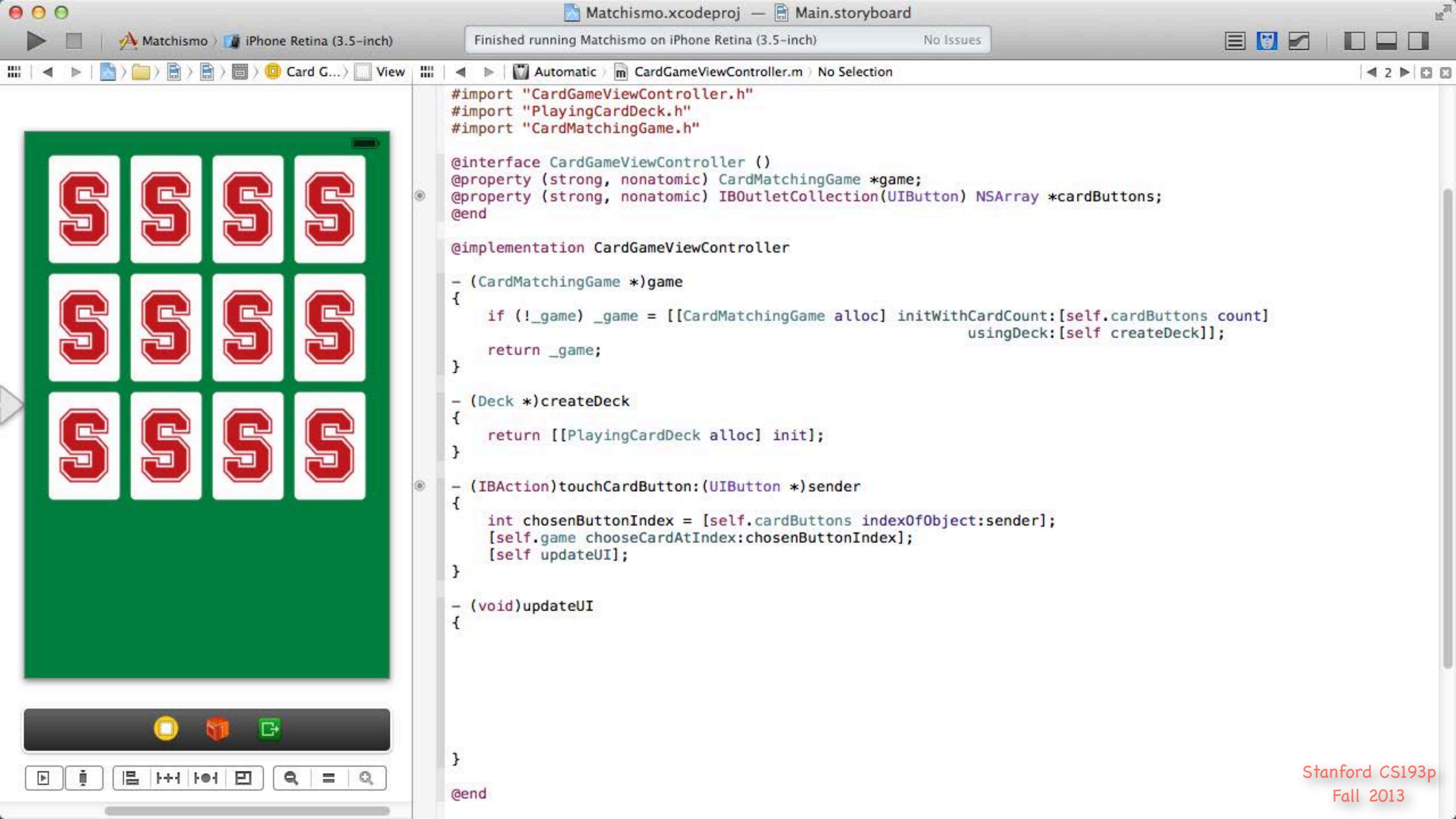
- (IBAction)touchCardButton:(UIButton *)sender  
{  
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];  
    [self.game chooseCardAtIndex:chosenButtonIndex];  
    [self updateUI];  
    self.flipCount++;  
}

- (void)updateUI { }

- (void)setFlipCount:(int)flipCount
```

Let's get rid of all the old stuff from our simpler version.
We don't need the Deck, nor the flip stuff.

Stanford CS193p
Fall 2013





```
Automatic CardGameViewController.m @implementation CardGameViewController  
@implementation CardGameViewController  
  
- (CardMatchingGame *)game  
{  
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]  
                           usingDeck:[self createDeck]];  
    return _game;  
}  
  
- (Deck *)createDeck  
{  
    return [[PlayingCardDeck alloc] init];  
}  
  
- (IBAction)touchCardButton:(UIButton *)sender  
{  
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];  
    [self.game chooseCardAtIndex:chosenButtonIndex];  
    [self updateUI];  
}  
  
- (void)updateUI  
{  
  
}  
}  
  
@end
```

Scanning down to make space ...

```
Matchismo.xcodeproj — Main.storyboard
Finished running Matchismo on iPhone Retina (3.5-inch)
Automatic CardGameViewController.m -updateUI
@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                               usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [_game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

- (void)updateUI
{
    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [_game cardAtIndex:cardButtonIndex];
    }
}
```

Updating the UI is quite straightforward.
We are just going to cycle through all the cardButtons and,
based on the corresponding card in our Model ...

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) 1 2

Automatic CardGameViewController.m -updateUI

```
@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                               usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [_game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

- (void)updateUI
{
    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [_game cardAtIndex:cardButtonIndex];
        [cardButton setTitle: (NSString *) forState:UIControlStateNormal];
        [cardButton setBackgroundImage: (UIImage *) forState:UIControlStateNormal];
    }
}
```

@end

... we'll set the title and background image of the cardButton.

We'll create some helper methods to calculate the title and image based on the card.

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) 1 2

Automatic CardGameViewController.m -titleForCard:

```
@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                               usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [self.game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

- (void)updateUI
{
    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [self.game cardAtIndex:cardButtonIndex];
        [cardButton setTitle:(NSString *) forState:UIControlStateNormal];
        [cardButton setBackgroundImage:(UIImage *) forState:UIControlStateNormal];
    }
}

- (NSString *)titleForCard:(Card *)card
{
    return card.isChosen ? card.contents : @"";
}

- (UIImage *)backgroundImageForCard:(Card *)card
{
    return [UIImage imageNamed:card.isChosen ? @"cardfront" : @"cardback"];
}
```

The title and background image just depend on whether the card is “chosen” (chosen is face up, not chosen is face down).

The screenshot shows the Xcode IDE with the project "Matchismo" open. The main window displays the file "CardGameViewController.m". The code implements a card matching game where each card has a red letter 'S' on it. A callout bubble points to the line of code that sets the button's title and background image based on the card's state.

```
@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                               usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [_game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

- (void)updateUI
{
    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [_game cardAtIndex:cardButtonIndex];
        [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
        [cardButton setBackgroundImage:[self backgroundImageForCard:card] forState:UIControlStateNormal];
    }
}

- (NSString *)titleForCard:(Card *)card
{
    return card.isChosen ? card.contents : @"";
}

- (UIImage *)backgroundImageForCard:(Card *)card
{
    return [UIImage imageNamed:card.isChosen ? @"cardfront" : @"cardback"];
}

@end
```

Now let's use those helper methods.

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch)

No Issues

Automatic CardGameViewController.m -updateUI

```
@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                               usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [self.game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

- (void)updateUI
{
    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [self.game cardAtIndex:cardButtonIndex];
        [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
        [cardButton setBackgroundImage:[self backgroundImageForCard:card] forState:UIControlStateNormal];
        cardButton.enabled = !card.isMatched;
    }
}

- (NSString *)titleForCard:(Card *)card
{
    return card.isChosen ? card.contents : @"";
}

- (UIImage *)backgroundImageForCard:(Card *)card
{
    return [UIImage imageNamed:card.isChosen ? @"cardfront" : @"cardback"];
}

@end
```

If a card.`isMatched`, we can disable the corresponding `cardButton`.

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Automatic CardGameViewController.m -updateUI

```
@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                               usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [_game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

- (void)updateUI
{
    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [_game cardAtIndex:cardButtonIndex];
        [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
        [cardButton setBackgroundImage:[self backgroundImageForCard:card] forState:UIControlStateNormal];
        cardButton.enabled = !card.isMatched;
    }
}

- (NSString *)titleForCard:(Card *)card
{
    return card.isChosen ? card.contents : @"";
}

- (UIImage *)backgroundImageForCard:(Card *)card
{
    return [UIImage imageNamed:card.isChosen ? @"cardfront" : @"cardback"];
}

@end
```

We really need to see the score!

The screenshot shows the Xcode interface with the CardGameViewController.m file open. The code implements a card matching game where four cards are shown in a 3x4 grid. Each card has a large red letter 'S' on it. A green callout bubble points to the code with the text "We really need to see the score!". At the bottom of the screen, there is a toolbar with various icons.

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Automatic CardGameViewController.m -updateUI

```
@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                               usingDeck:[self createDeck]];

    return _game;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [_game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

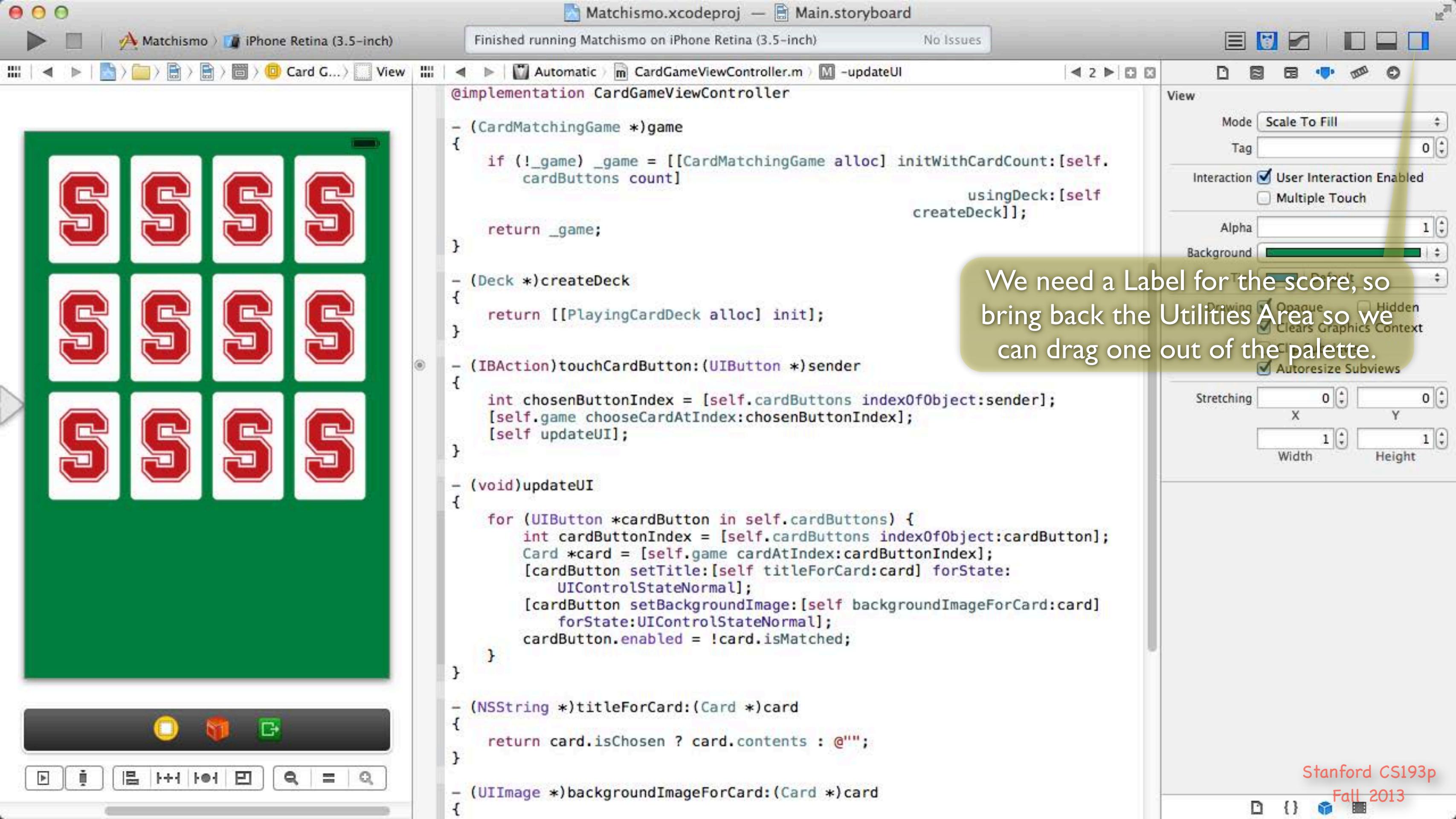
- (void)updateUI
{
    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [_game cardAtIndex:cardButtonIndex];
        [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
        [cardButton setBackgroundImage:[self backgroundImageForCard:card]
                           forState:UIControlStateNormal];
        cardButton.enabled = !card.isMatched;
    }
}

- (NSString *)titleForCard:(Card *)card
{
    return card.isChosen ? card.contents : @"";
}

- (UIImage *)backgroundImageForCard:(Card *)card
{
```

We need a Label for the score, so bring back the Utilities Area so we can drag one out of the palette.

Mode Scale To Fill
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background Default
Drawing Opaque
Hidden
Clears Graphics Context
Autoresizing Subviews
Stretching X 0 Y 0
Width 1 Height 1



Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Card Game View Controller

```
@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                               usingDeck:[self createDeck]];

    return _game;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [_game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

- (void)updateUI
{
    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [_game cardAtIndex:cardButtonIndex];
        [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
        [cardButton setBackgroundImage:[self backgroundImageForCard:card]
                           forState:UIControlStateNormal];
        cardButton.enabled = !card.isMatched;
    }
}

- (NSString *)titleForCard:(Card *)card
{
    return card.isChosen ? card.contents : @"";
}

- (UIImage *)backgroundImageForCard:(Card *)card
{
```

View

Mode Scale To Fill

Tag 0

Interaction User Interaction Enabled
 Multiple Touch

Alpha 1

Background

Tint Default

Drawing Opaque Hidden
 Clears Graphics Context
 Clip Subviews
 Autoresizes Subviews

Stretching 0 0

X 1 Y 1

Width 1 Height 1

Object Palette.

objects and controllers not directly available in Interface...

Label Label - A variably sized amount of static text.

Button Button - Intercepts touch events and sends an action message to a target object when it's tapped.

Segmented Control Segmented Control - Displays multiple segments, each of which functions as a discrete button.

Text Field Text Field - Displays editable

Stanford CS193p Fall 2013

Matchismo > iPhone Retina (3.5-inch)



```

@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.
        cardButtons count]
                                                usingDeck:[self
createDeck]];

    return _game;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [_game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

- (void)updateUI
{
    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [_game cardAtIndex:cardButtonIndex];
        [cardButton setTitle:[self titleForCard:card] forState:
            UIControlStateNormal];
        [cardButton setBackgroundImage:[self backgroundImageForCard:card]
            forState:UIControlStateNormal];
        cardButton.enabled = !card.isMatched;
    }
}

- (NSString *)titleForCard:(Card *)card
{
    return card.isChosen ? card.contents : @"";
}

- (UIImage *)backgroundImageForCard:(Card *)card
{
}

```

Drag a Label from the
Object Library to your View.

View

Mode **Scale To Fill**
Tag 0
Interaction User Interaction Enabled
 Multiple Touch
Alpha 1
Background
Tint **Default**
Drawing Opaque Hidden
 Clears Graphics Context
 Clip Subviews
 Autoresizes Subviews
Stretching 0 0
X 1 Y 1
Width 1 Height 1

objects and controllers not
directly available in Interface...

Label Label – A variably sized amount
of static text. A variably sized amount
of static text.

Button Button – Intercepts touch events
and sends an action message to a
target object when it's tapped.

Segmented Control Segmented Control – Displays
multiple segments, each of which
functions as a discrete button.

Stanford CS193p
Text Field – Displays editable
Fall 2013

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Card Game View Controller

```
@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                               usingDeck:[self createDeck]];

    return _game;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [_game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

- (void)updateUI
{
    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [_game cardAtIndex:cardButtonIndex];
        [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
        [cardButton setBackgroundImage:[self backgroundImageForCard:card]
                           forState:UIControlStateNormal];
        cardButton.enabled = !card.isMatched;
    }
}

- (NSString *)titleForCard:(Card *)card
{
    return card.isChosen ? card.contents : @"";
}

- (UIImage *)backgroundImageForCard:(Card *)card
{
```

View

Mode Scale To Fill

Tag 0

Interaction User Interaction Enabled
 Multiple Touch

Alpha 1

Background

Tint Default

Drawing Opaque Hidden
 Clears Graphics Context
 Clip Subviews
 Autoresizes Subviews

Stretching 0 0

X 1 Y 1

Width 1 Height 1

objects and controllers not directly available in Interface...

Label Label - A variably sized amount of static text.

Button Button - Intercepts touch events and sends an action message to a target object when it's tapped.

Segmented Control Segmented Control - Displays multiple segments, each of which functions as a discrete button.

Text Field Text Field - Displays editable

Stanford CS193p Fall 2013

Lower left corner
(where Flips used to be)
is a good spot.

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Automatic CardGameViewController.m -updateUI

```
@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                               usingDeck:[self createDeck]];

    return _game;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [_game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

- (void)updateUI
{
    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        [_game chooseCardAtIndex:cardButtonIndex];
        [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
        [cardButton setBackgroundImage:[self backgroundImageForCard:card]
                           forState:UIControlStateNormal];
        cardButton.enabled = !card.isMatched;
    }
}

- (NSString *)titleForCard:(Card *)card
{
    return card.isChosen ? card.contents : @"";
}

- (UIImage *)backgroundImageForCard:(Card *)card
{
```

Score: 0

Label

Text Plain

Score: 0

Color Black Color

Font System 17.0

Alignment

Lines 1

Behavior Enabled

Highlighted

Baseline Align Baselines

Line Breaks Truncate Tail

Autoshrink Fixed Font Size

Tighten Letter Spacing

Highlighted Default

Shadow Default

Shadow Offset 0 -1

objects and controllers not directly available in Interface...

Label Label - A variably sized amount of static text.

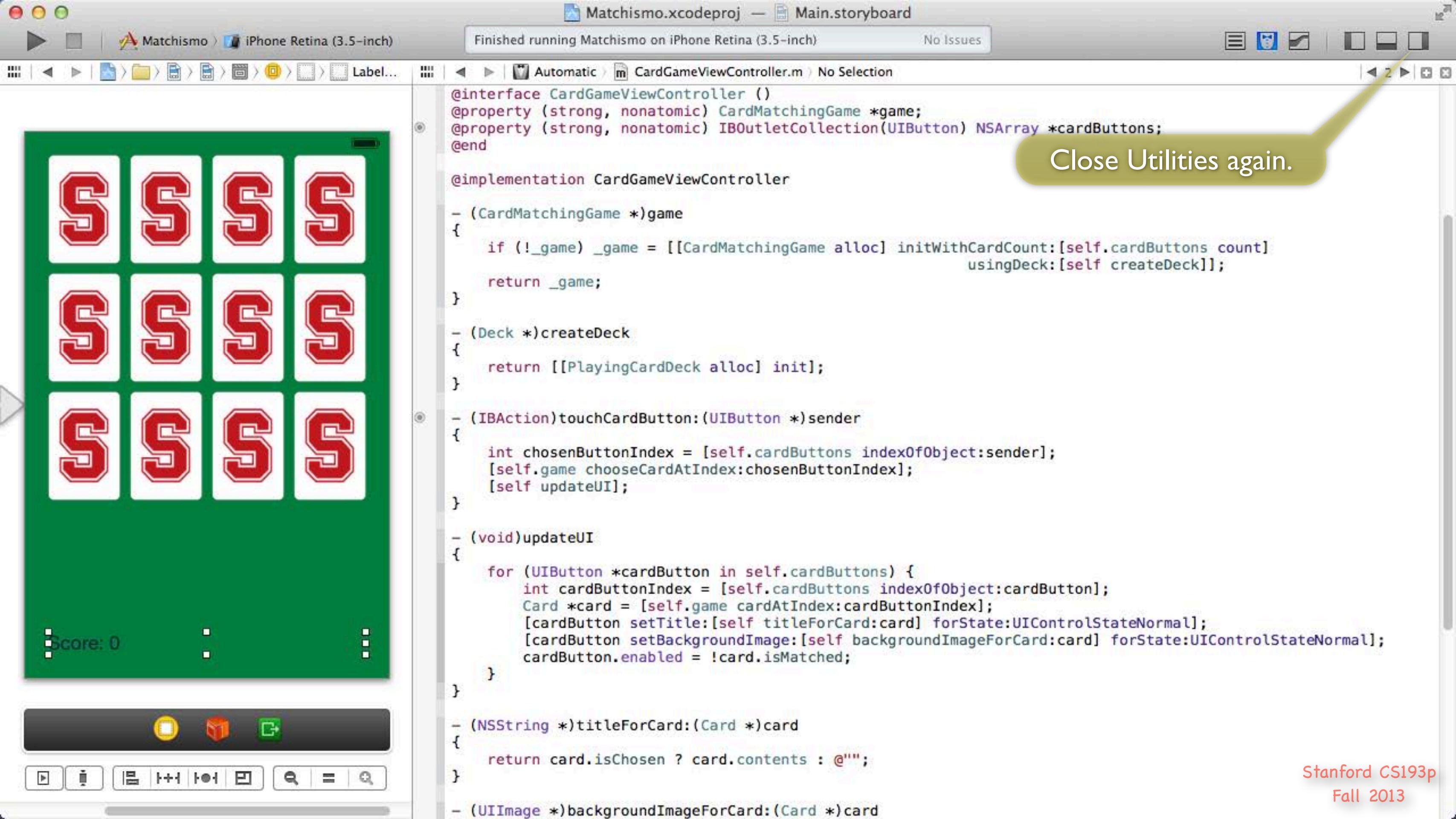
Button Button - Intercepts touch events and sends an action message to a target object when it's tapped.

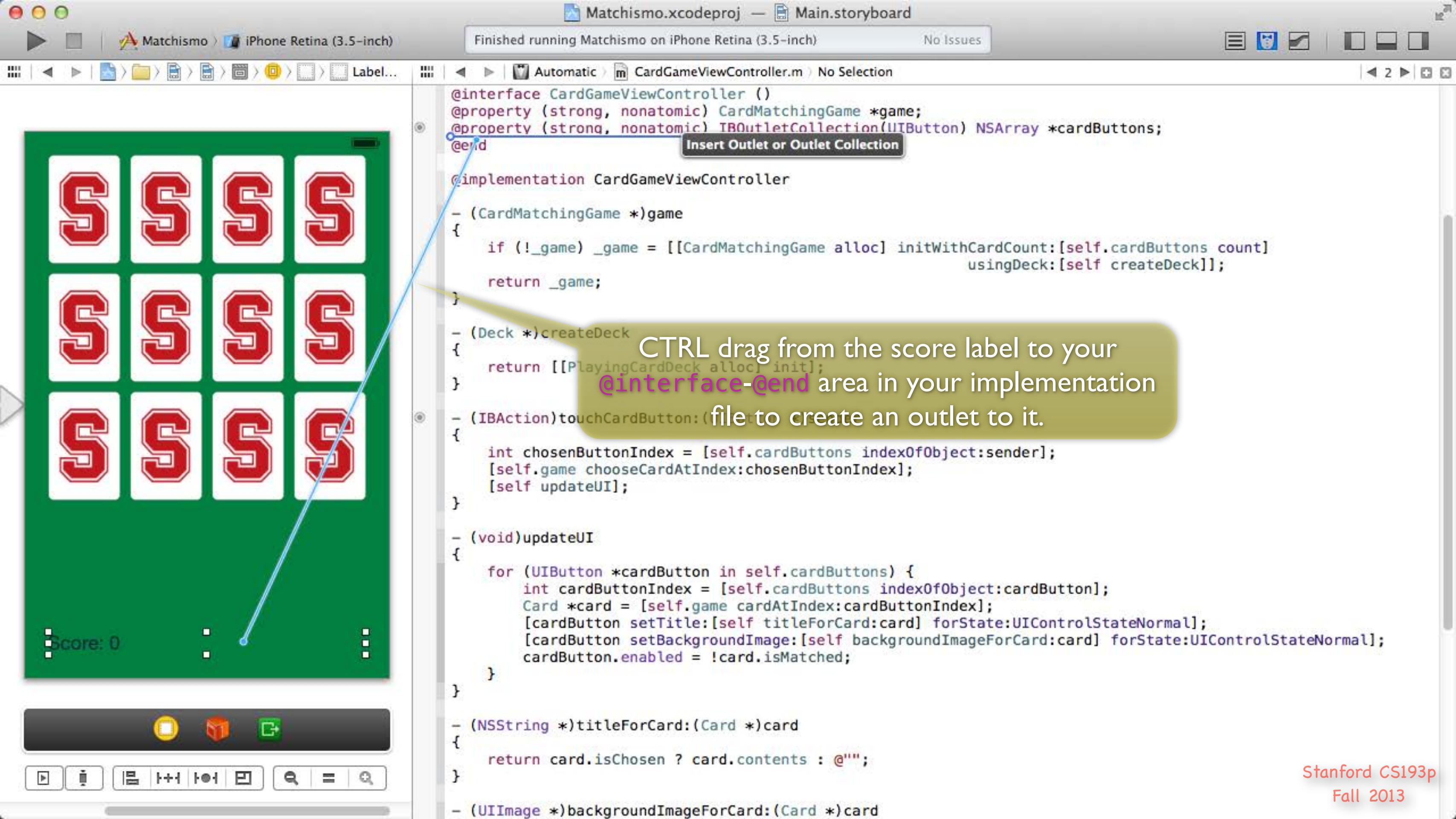
Segmented Control Segmented Control - Displays multiple segments, each of which functions as a discrete button.

Text Field Text Field - Displays editable

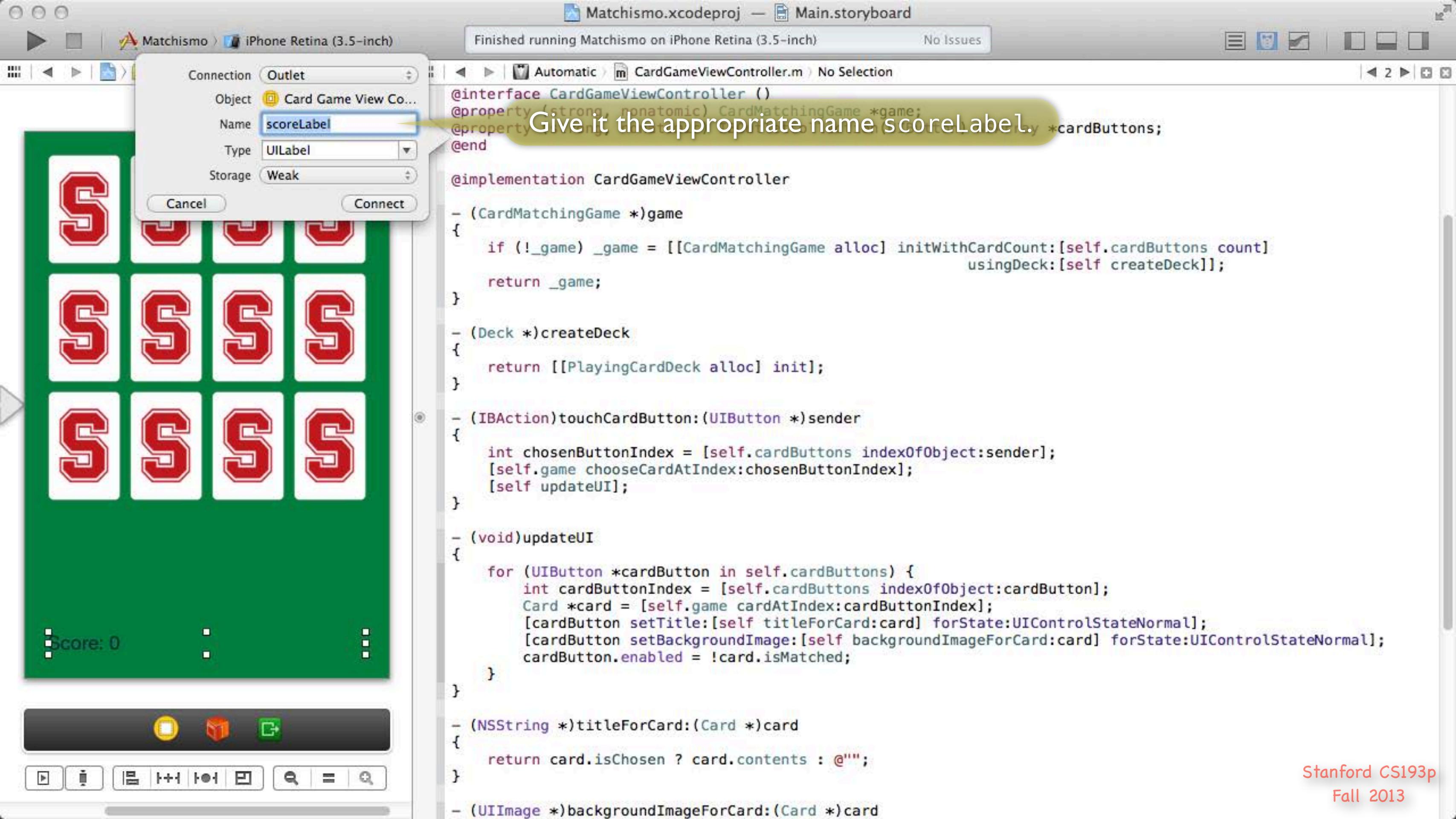
Stanford CS193p Fall 2013

Start with “Score: 0” and make it wider.





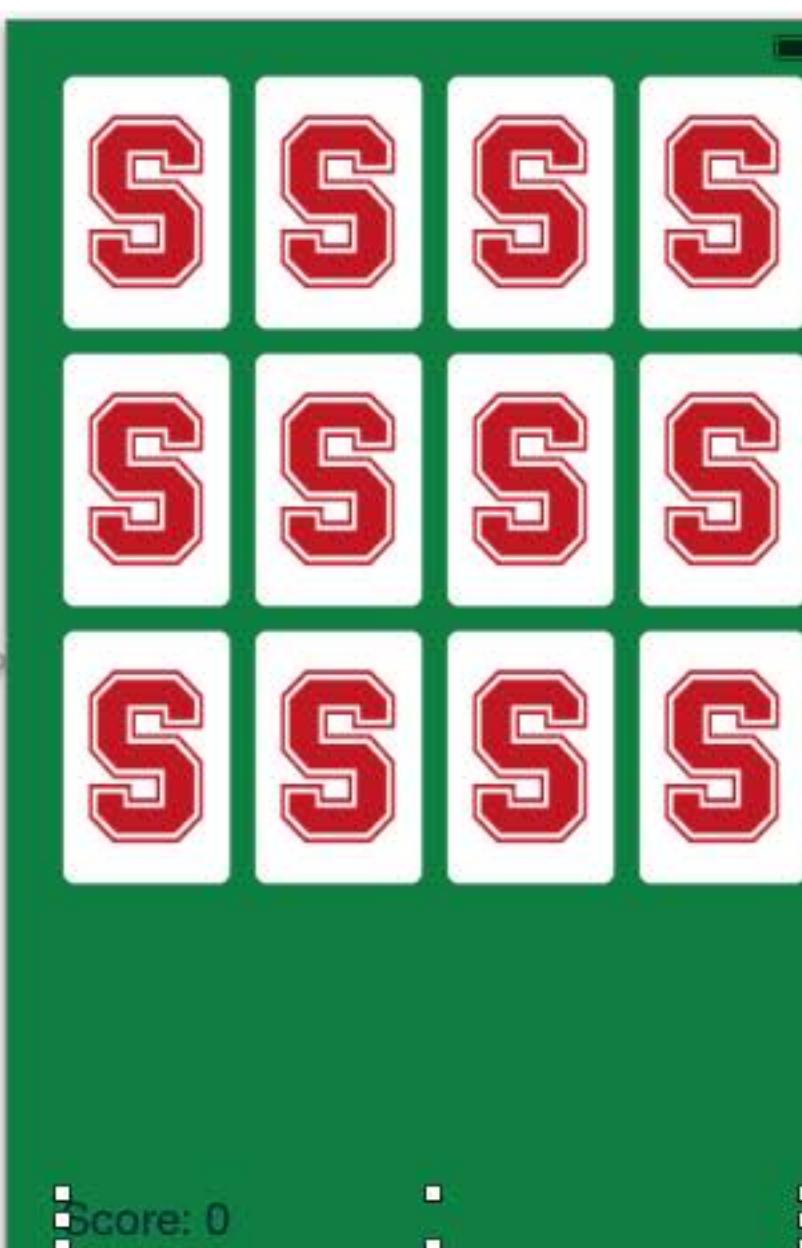
CTRL drag from the score label to your
@interface-@end area in your implementation
file to create an outlet to it.



Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

Automatic CardGameViewController.m No Selection



```
@interface CardGameViewController ()  
@property (strong, nonatomic) CardMatchingGame *game;  
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *cardButtons;  
@property (weak, nonatomic) IBOutlet UILabel *scoreLabel;  
@end  
  
@implementation CardGameViewController  
- (CardMatchingGame *)game  
{  
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]  
                           usingDeck:[self createDeck]];  
    return _game;  
}  
- (Deck *)createDeck  
{  
    return [[PlayingCardDeck alloc] init];  
}  
- (IBAction)touchCardButton:(UIButton *)sender  
{  
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];  
    [_game chooseCardAtIndex:chosenButtonIndex];  
    [self updateUI];  
}  
- (void)updateUI  
{  
    for (UIButton *cardButton in self.cardButtons) {  
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];  
        Card *card = [_game cardAtIndex:cardButtonIndex];  
        [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];  
        [cardButton setBackgroundImage:[self backgroundImageForCard:card] forState:UIControlStateNormal];  
        cardButton.enabled = !card.isMatched;  
    }  
}  
- (NSString *)titleForCard:(Card *)card  
{  
    return card.isChosen ? card.contents : @"";  
}
```

Matchismo.xcodeproj — Main.storyboard

Finished running Matchismo on iPhone Retina (3.5-inch) No Issues

```
@interface CardGameViewController ()  
@property (strong, nonatomic) CardMatchingGame *game;  
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *cardButtons;  
@property (weak, nonatomic) IBOutlet UILabel *scoreLabel;  
@end  
  
@implementation CardGameViewController  
- (CardMatchingGame *)game  
{  
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]  
                           usingDeck:[self createDeck]];  
    return _game;  
}  
  
- (Deck *)createDeck  
{  
    return [[PlayingCardDeck alloc] init];  
}  
  
- (IBAction)touchCardButton:(UIButton *)sender  
{  
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];  
    [self.game chooseCardAtIndex:chosenButtonIndex];  
    [self updateUI];  
}  
  
- (void)updateUI  
{  
    for (UIButton *cardButton in self.cardButtons) {  
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];  
        Card *card = [self.game cardAtIndex:cardButtonIndex];  
        [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];  
        [cardButton setBackgroundImage:[self backgroundImageForCard:card] forState:UIControlStateNormal];  
        cardButton.enabled = !card.isMatched;  
        self.scoreLabel.text = [NSString stringWithFormat:@"Score: %d", self.game.score];  
    }  
}  
  
- (NSString *)titleForCard:(Card *)card  
{  
    return card.isChosen ? card.contents : @"";  
}
```

Update the score
(using the same code as we used for Flips)
whenever we update the rest of the UI.

We're using our Model here.

Stanford CS193p
Fall 2013

Matchismo.xcodeproj — Main.storyboard

Running Matchismo on iPhone Retina (3.5-inch) No Issues

Automatic CardGameViewController.m @implementation CardGameViewController

```
if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
usingDeck:[self createDeck]];
return _game;
```

(Deck *)createDeck

```
return [[PlayingCardDeck alloc] init];
```

(IBAction)touchCardButton:(UIButton *)sender

```
int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
[self.game chooseCardAtIndex:chosenButtonIndex];
[self updateUI];
```

(void)updateUI

```
for (UIButton *cardButton in self.cardButtons) {
    int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
    Card *card = [self.game cardAtIndex:cardButtonIndex];
    [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
    [cardButton setBackgroundImage:[self backgroundImageForCard:card] forState:UIControlStateNormal];
    cardButton.enabled = !card.isMatched;
    self.scoreLabel.text = [NSString stringWithFormat:@"Score: %d", self.game.score];
}
```

(NSString *)titleForCard:(Card *)card

```
return card.isChosen ? card.contents : @"";
```

(UIImage *)backgroundImageForCard:(Card *)card

Carrier



Score: 0

Done! Final run!

```
if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
usingDeck:[self createDeck]];
return _game;
```

(Deck *)createDeck

```
return [[PlayingCardDeck alloc] init];
```

(IBAction)touchCardButton:(UIButton *)sender

```
int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
[self.game chooseCardAtIndex:chosenButtonIndex];
[self updateUI];
```

(void)updateUI

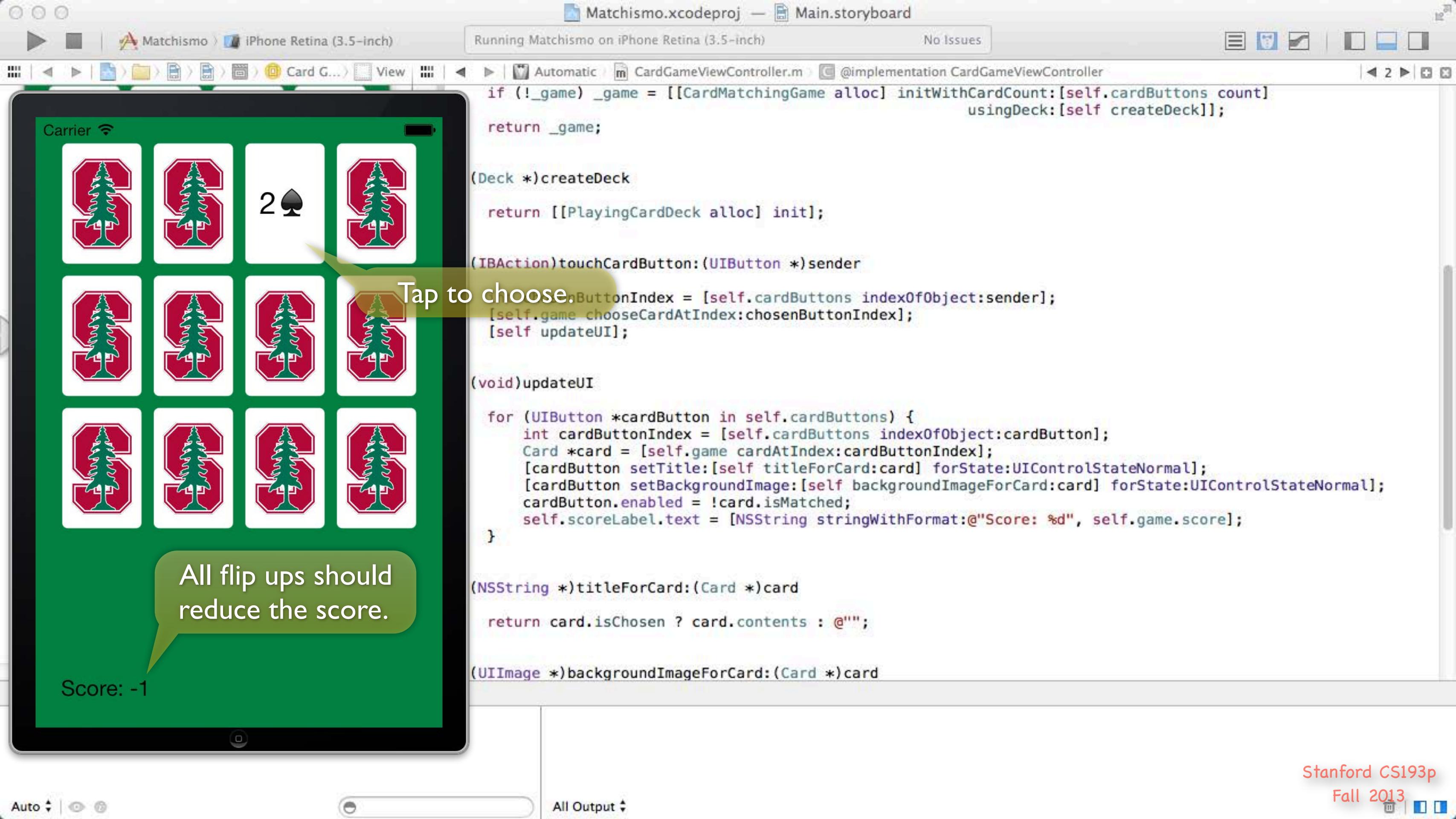
```
for (UIButton *cardButton in self.cardButtons) {
    int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
    Card *card = [self.game cardAtIndex:cardButtonIndex];
    [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
    [cardButton setBackgroundImage:[self backgroundImageForCard:card] forState:UIControlStateNormal];
    cardButton.enabled = !card.isMatched;
    self.scoreLabel.text = [NSString stringWithFormat:@"Score: %d", self.game.score];
}
```

(NSString *)titleForCard:(Card *)card

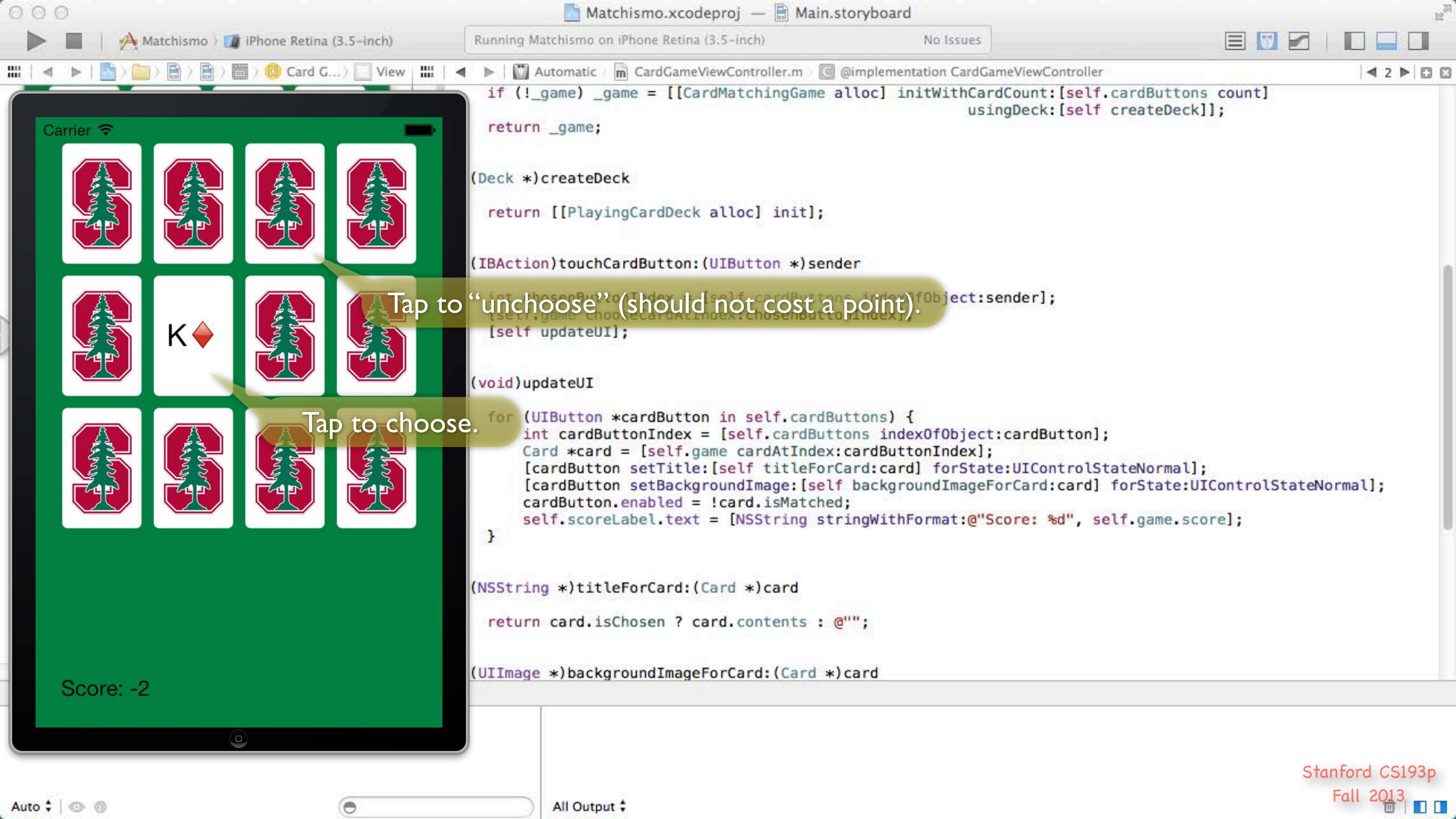
```
return card.isChosen ? card.contents : @"";
```

(UIImage *)backgroundImageForCard:(Card *)card

All Output

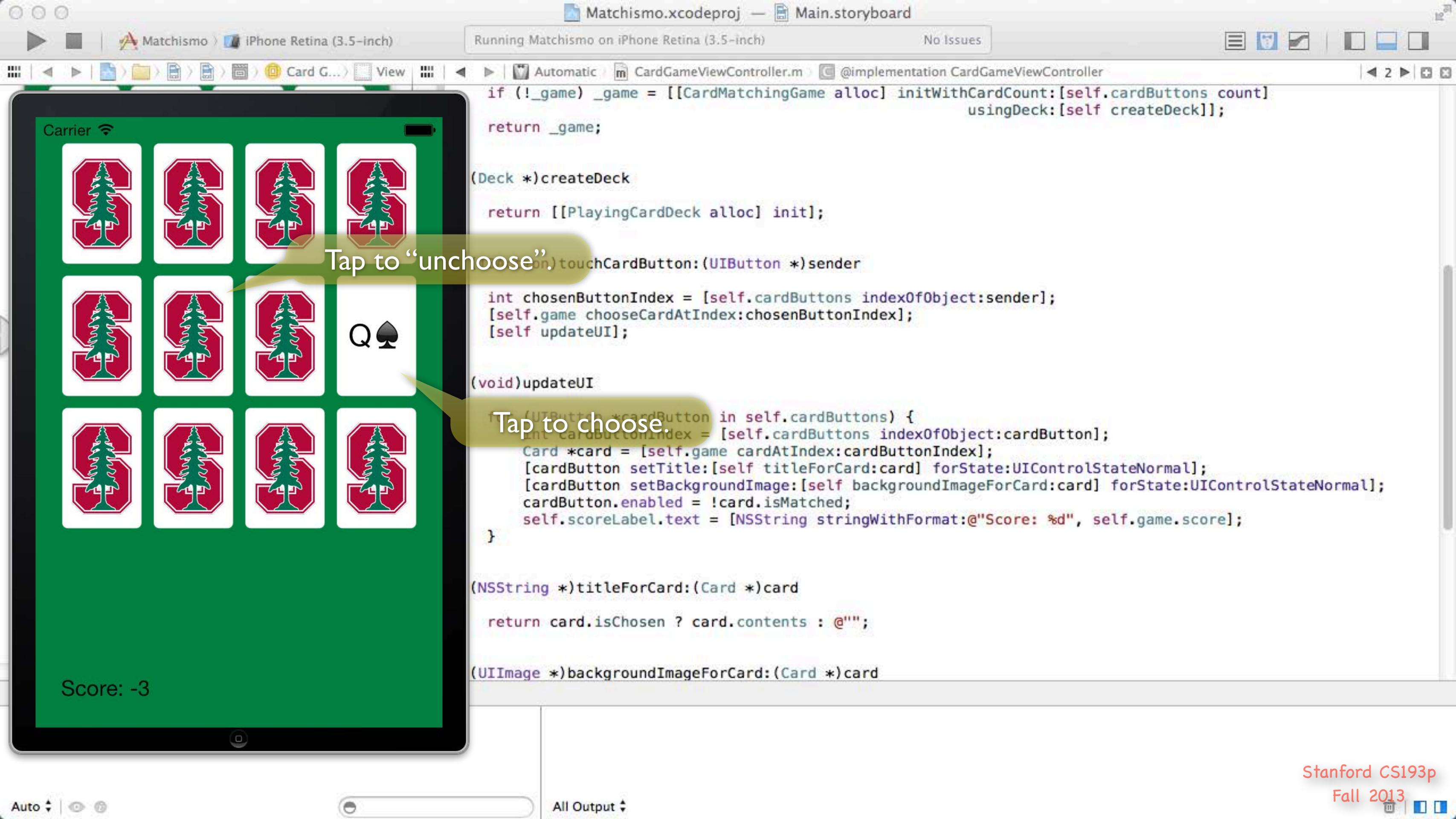


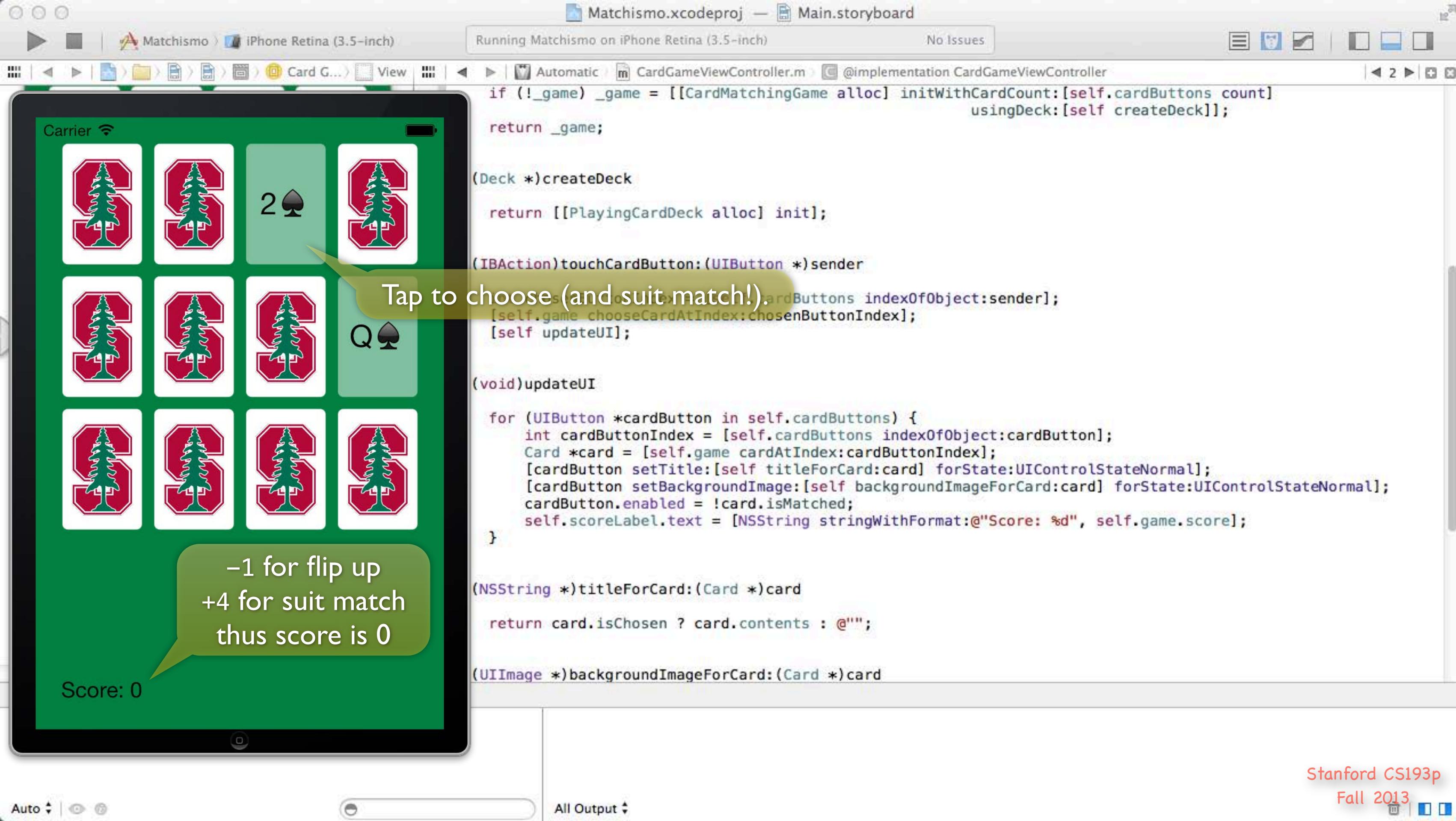
Stanford CS193p
Fall 2013



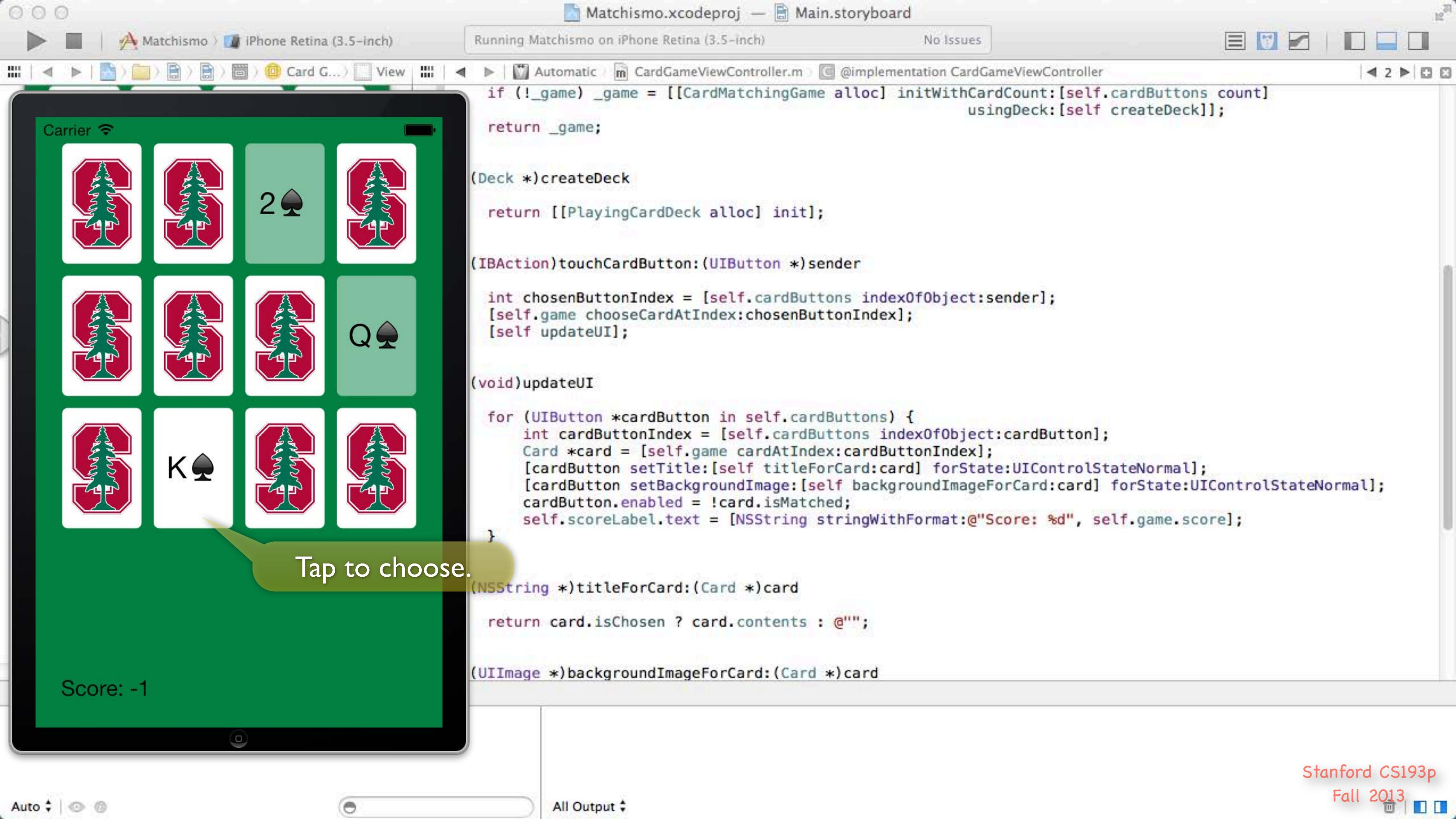
Stanford CS193p

Fall 2013



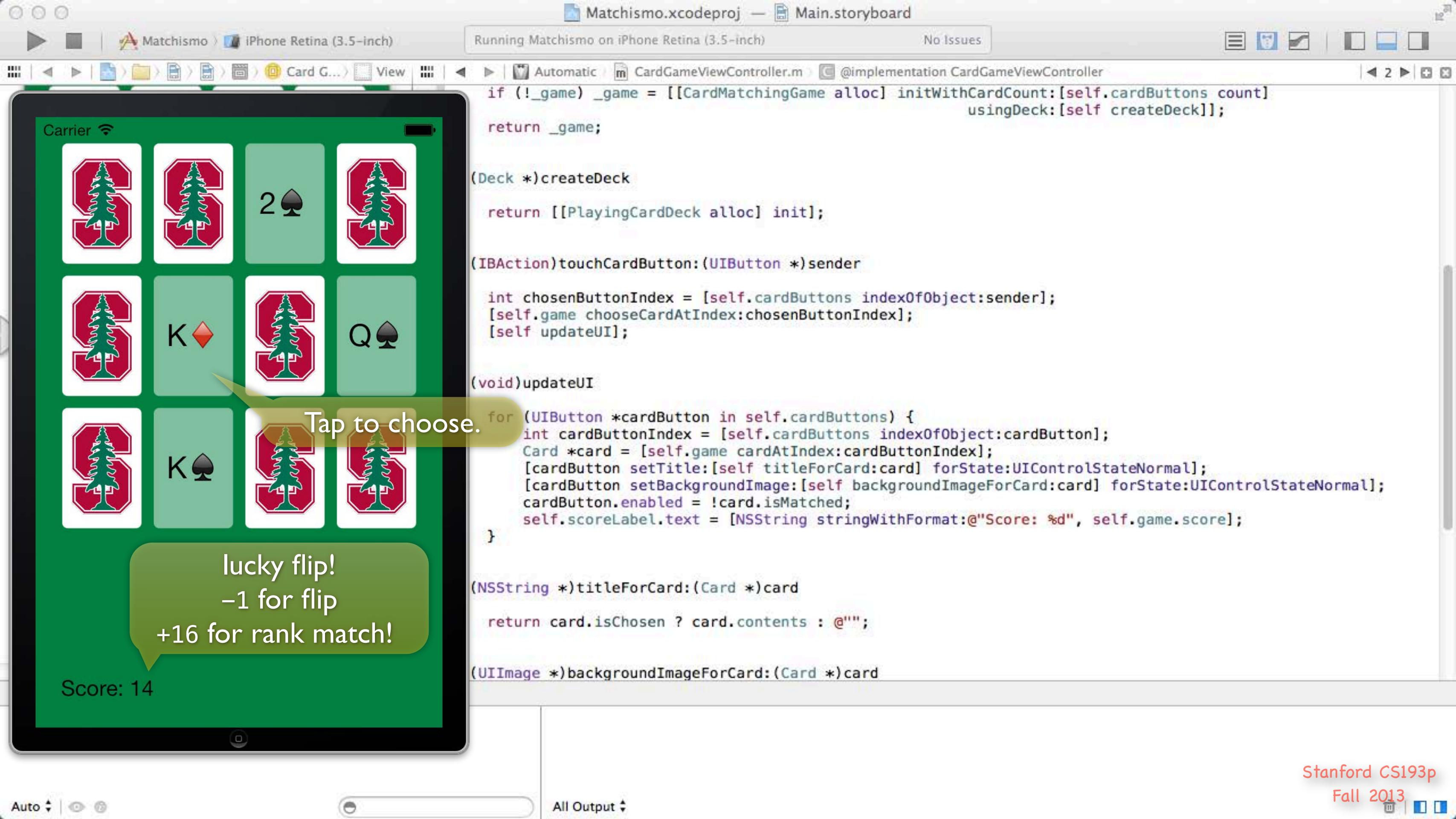


Stanford CS193p
Fall 2013



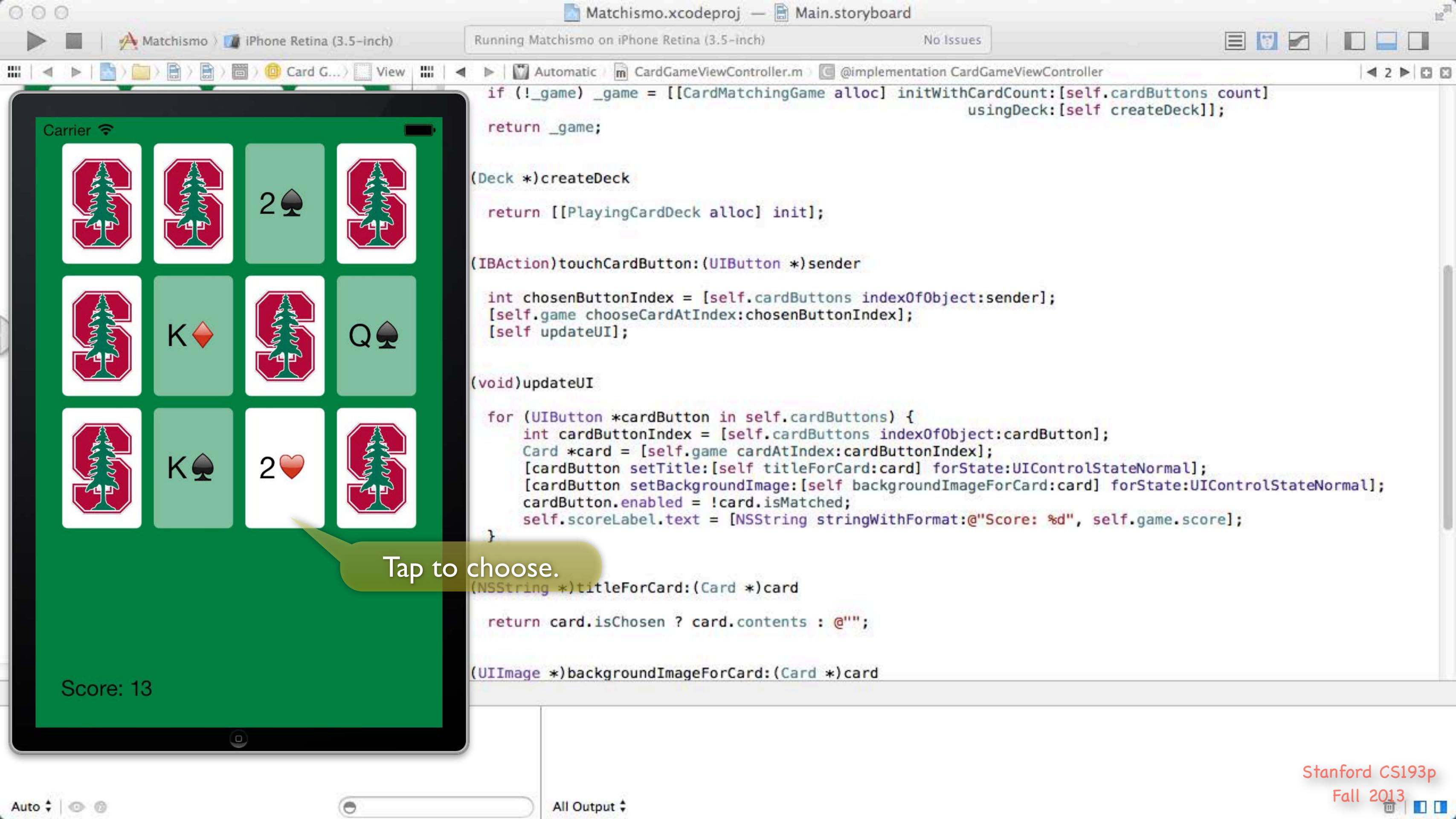
Stanford CS193p

Fall 2013



Stanford CS193p

Fall 2013



Stanford CS193p

Fall 2013

Matchismo.xcodeproj — Main.storyboard

Running Matchismo on iPhone Retina (3.5-inch) No Issues

Automatic CardGameViewController.m @implementation CardGameViewController

```
if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                     usingDeck:[self createDeck]];
return _game;

(Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

IBAction touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [_game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

for (UIButton *cardButton in self.cardButtons) {
    int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
    Card *card = [_game cardAtIndex:cardButtonIndex];
    [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
    [cardButton setBackgroundImage:[self backgroundImageForCard:card] forState:UIControlStateNormal];
    cardButton.enabled = !card.isMatched;
    cardButton.tag = [NSString stringWithFormat:@"%d", game.score];
}

 NSString *(Card *)titleForCard:(Card *)card
{
    return card.isChosen ? card.contents : @"";
}

 UIImage *(Card *)backgroundImageForCard:(Card *)card
{
    return card.isChosen ? card.contents : @"";
}
```

Carrier

-1 for flip
-2 for mismatch!

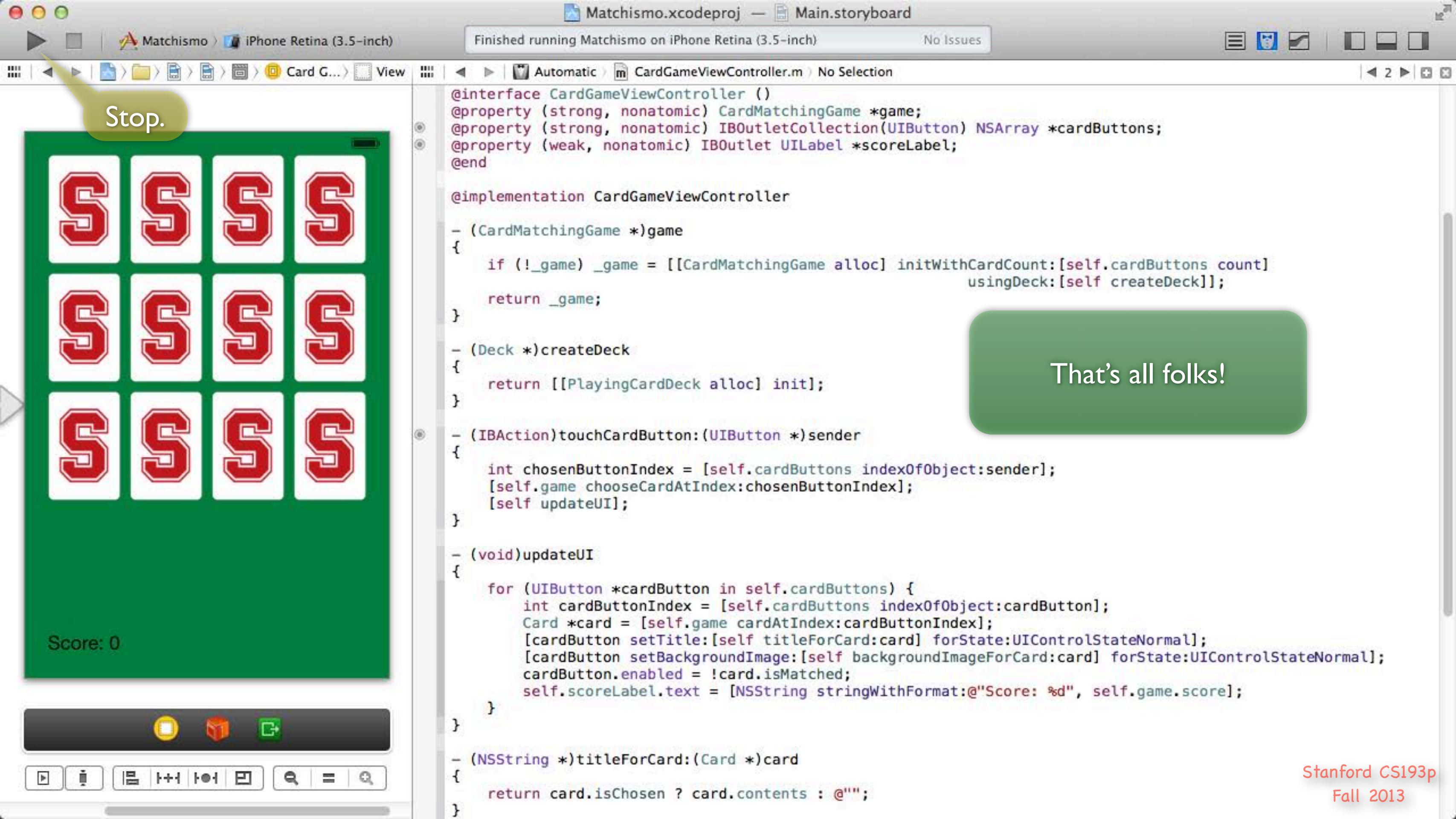
Score: 10

Tap to choose (and cause mismatch!)

Automatically “unchosen” on mismatch.

Be sure to check for rank and suit matches and also for mismatches.

Stanford CS193p
Fall 2013



Review

- ⌚ Things you should know by now ...

MVC

Xcode

Basic Objective-C

Review (MVC)

- ⦿ Model is UI-independent

Cards and Decks, not UIButton and UILabel

- ⦿ View is (so far) completely generic UI elements

UIButton

UILabel

- ⦿ Controller interprets Model for View (and vice-versa)

Example: converting isChosen to selected state of a button

Example: converting isMatched to enabled state of a button

Example: taking a button touch and turning it into a chooseCardAtIndex: in the Model

Target/Action and Outlets (so far)

Review (Xcode)

>Create a Project and maneuver through Xcode's UI

Hide/Show Navigator, Utilities, Assistant Editor, etc., etc. Also how to run in the Simulator.

Edit

Not just code, but also your storyboard, use Attributes Inspector to edit buttons, labels, et. al.

Ctrl-drag to make connections (actions and outlets).

Right click on buttons, etc., to find out about and disconnect connections.

Look at warnings and errors (and get rid of them hopefully!). Debugger on Friday this week.

Add classes to your project

e.g. you added the Card, etc., Model classes in your Homework assignment.

Use the documentation

Many ways to get to documentation, but ALT-clicking on a keyword is one of the coolest.

Once there, search and click on links to find what you want.

Crucial to being a good iOS programming to become familiar with all the documentation.

Review (Basic Objective-C)

• Classes

Header .h (public) versus Implementation .m (private)

@interface MyClass : MySuperclass ... @end (only in header file)

@interface MyClass() ... @end (only in implementation file)

@implementation ... @end (only in implementation file)

#import

• Properties

@property (nonatomic) <type> <property name> (always nonatomic in this course)

It's just setter and getter methods. Default ones automatically generated for you by compiler.

Better than instance variables alone (lazy instantiation, consistency checking, UI updating, etc.).

@property (strong or weak) <type which is a pointer to an object> <property name>

@property (getter=<getter name>) ...

@property (readonly) ... & @property (readwrite) ...

Invoking setter and getter using dot notation, e.g., self.cards = ... or if (rank > self.rank) ...

@synthesize <prop name> = _<prop name> (only if you implement both setter and getter)

Review (Basic Objective-C)

⌚ Types and Memory

Types: `MyClass *`, `BOOL` (`YES` or `NO`), `int`, `NSUInteger`, etc. (`id` not fully explained yet.)

All objects live in the heap (i.e. we only have pointers to them).

Object storage in the heap is managed automatically (guided by `strong` and `weak` declarations).

Lazy instantiation (using a `@property`'s getter to `allocate` and `initialize` the object that the `@property` points to in an “on demand” fashion). Not everything is lazily instantiated, btw. :)

If a pointer has the value `nil` (i.e. `0`), it means the pointer does not point to anything.

⌚ Methods

Declaring and defining instance methods, e.g., `- (int)match:(NSArray *)otherCards`

Declaring and defining class methods, e.g., `+ (NSArray *)validSuits`

Invoking instance methods, e.g., `[myArray addObject:anObject]`

Invoking class methods, e.g., `unsigned int rank = [PlayingCard maxRank]`

Method's name and its parameters are interspersed, e.g., `[deck addCard:aCard atTop:YES]`

Review (Basic Objective-C)

• **NSString**

Immutable and usually created by manipulating other strings or `@""` notation or class methods.

e.g. `NSString *myString = @“hello”`

e.g. `NSString *myString = [otherString stringByAppendingString:yetAnotherString]`

e.g. `NSString *myString = [NSString stringWithFormat:@“%d%@”, myInt, myObj]`

There is an `NSMutableString` subclass but we almost never use it.

Instead, we create new strings by asking existing ones to create a modified version of themselves.

Review (Basic Objective-C)

• NSArray

Immutable and usually created by manipulating other arrays (not seen yet) or with `@[]` notation.

`[@[@"a",@"b"]` is the same as `[[NSArray alloc] initWithObjects:@"a",@"b",nil]`.

Access the array using `[]` notation (like a normal C array), e.g., `myArray[index]`.

`myArray[index]` works the same as `[myArray objectAtIndex:index]`.

The method `count` (which returns `NSUInteger`) will tell you how many items in the array.

(We accidentally used dot notation to call this method in Lecture 2!)

Be careful not to access array index out of bounds (crashes). Only `lastObject` immune.

Can contain any mix of objects of any class) No syntax to say which it contains.

Use `NSMutableArray` subclass if mutability is needed. Then you get ...

- `(void)addObject:(id)anObject;`
- `(void)insertObject:(id)anObject atIndex:(int)index;`
- `(void)removeObjectAtIndex:(int)index;`

Usually created with `[[NSMutableArray alloc] init]`

Review (Basic Objective-C)

Creating Objects in the Heap

Allocation (`NSObject`'s alloc`) and initialization (with an `init...` method) always happen together!

e.g. `[[NSMutableArray alloc] init]`

e.g. `[[CardMatchingGame alloc] initWithCardCount:c usingDeck:d]`

Writing initializers for your own classes ...

Two kinds of initializers: designated (one per class) and convenience (zero or more per class).

Only denoted by comments (not enforced by the syntax of the language in any way).

Must call `your super`'s designated initializer` (from your designated initializer)
or `your own designated initializer` (from your own convenience initializers).

This whole concept takes some getting used to.

Luckily, because of lazy instantiation, et. al., we don't need initializers that much in Objective-C.

And calling initializers is easy (it's just `alloc` plus whatever `initializer` you can find that you like).

Review (Basic Objective-C)

⌚ Other

Fast enumeration: `for (MyClass *myObject in arrayOfMyObjects) { }.`

`#define`

`NSLog(@"show this object %@", anObject)`

⌚ Quiz

What does this do?

`cardA.contents = @[cardB.contents, cardC.contents] [[cardB match:@[cardC]] ? 1 : 0]`

This line has a **setter**, **getters**, **method invocation**, **array creation** and **array accessing** all in one.

And lots of square brackets.

Coming Up

⌚ Next Lecture

More detail about Objective-C

More Foundation classes (besides strings and arrays)

Attributed strings

⌚ Next week ...

Multiple MVCs in your storyboard

View Controller Lifecycle