

EECS 395

Day 9

Nathan Nichols

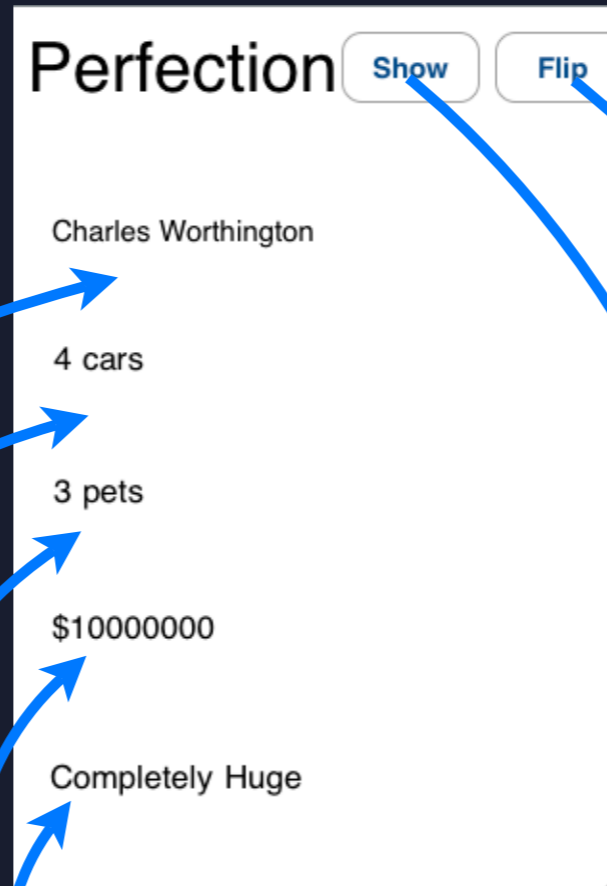
Today

- Recap / finish UITableView
- One way to setup NURecipe
- Inter-Language Comparison

NUHelloWorld

PerfectView

(subclass of UIView)



Methods

Variables

PerfectViewController

(subclass of
UIView
Controller)

PrimeGame

Cupertino
Glendale
Los Angeles
Palo Alto
San Diego
San Francisco
Santa Clara
Santa Monica
Sherman Oaks

There's nothing we can wire up in Interface Builder. How will we connect the TableView up to the rest of our program?

Methods

Variables

PrimeGameController

UITableView

By using the UITableView's delegate and datasource

UITableView

Follows the
UITableViewDataSource
protocol



Cupertino
Glendale
Los Angeles
Palo Alto
San Diego
San Francisco
Santa Clara
Santa Monica
Sherman Oaks

How many cells do I have?
It's a plain cell with text "Glendale"
What's the cell in section 0, row 1?

UITableView

Cupertino
Glendale
Los Angeles
Palo Alto
San Diego
San Francisco
Santa Clara
Santa Monica
Sherman Oaks

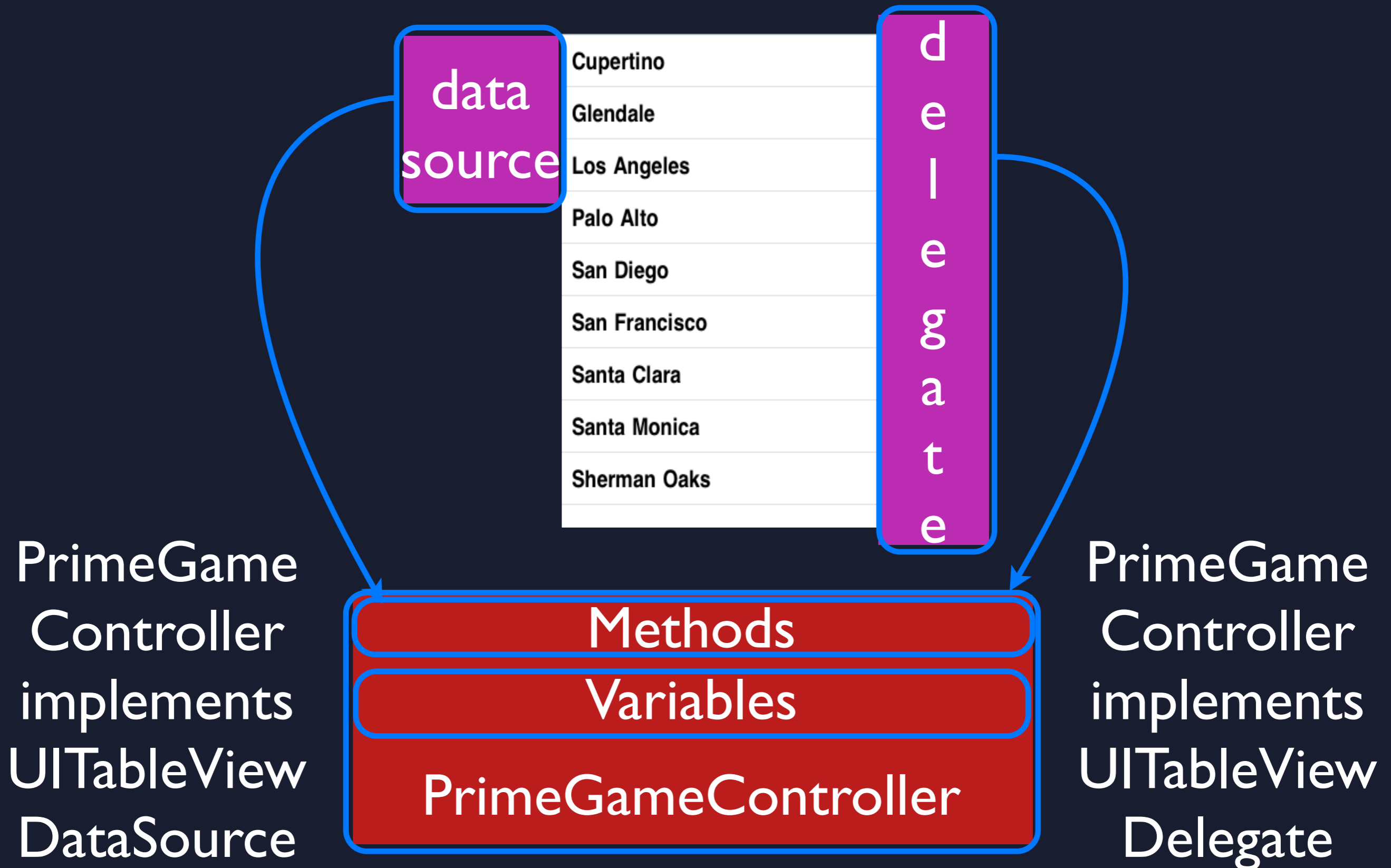
d
e
l
e
g
a
t
e

Follows the
UITableViewDelegate
protocol



UITableViewDataSource
UITableViewDelegate
UITableView
UITableViewCell
UITableViewRowAction
UITableViewRowAnimation
UITableViewRowAnimationNone
UITableViewRowAnimationFade
UITableViewRowAnimationSlide
UITableViewRowAnimationTop
UITableViewRowAnimationBottom
UITableViewRowAnimationMiddle
UITableViewRowAnimationAutomatic
UITableViewRowAnimationVertical
UITableViewRowAnimationHorizontal
UITableViewRowAnimationNone
UITableViewRowAnimationFade
UITableViewRowAnimationSlide
UITableViewRowAnimationTop
UITableViewRowAnimationBottom
UITableViewRowAnimationMiddle
UITableViewRowAnimationAutomatic
UITableViewRowAnimationVertical
UITableViewRowAnimationHorizontal

UITableView



PrimeGame

```
@interface PrimeGameController : UITableViewController \
    <UITableViewDelegate, UITableViewDataSource> {

}

@end
```

PrimeGame

```
//PrimeGameController.m
-(void)viewDidLoad {
    [self.tableView setDelegate: self];
    [self.tableView setDataSource: self];
    [self.tableView reloadData];
}
```

UITableViewDataSource

- Two required methods:
 - One takes a UITableView and NSIndexPath and returns the appropriate UITableViewCell
 - One takes a UITableView and section number and returns the number of cells
- About nine optional methods
 - Header and footer text, etc.
- It's not clear to me why some methods are in UITableViewDataSource and some are in UITableViewDelegate
 - Just use the documentation

UITableViewDataSource

```
- (NSInteger)tableView:(UITableView *)tableView \
    numberOfRowsInSection:(NSInteger)section {
    return 100000000;
}
```

UITableViewDataSource

```
- (UITableViewCell *)tableView:(UITableView *)tableView \
    cellForRowAtIndexPath:(NSIndexPath *)indexPath {
    static NSString *CellIdentifier = @"Cell";
    UITableViewCell *cell = [tableView \
        dequeueReusableCellWithIdentifier:CellIdentifier];
    if (cell == nil) {
        cell = [[[UITableViewCell alloc] \
            initWithFrame:CGRectZero reuseIdentifier:CellIdentifier] \
            autorelease];
    }
    //Could do all kinds of stuff with cell here
    [cell setAccessoryType:UITableViewCellAccessoryNone];
    [cell setText:[NSString stringWithFormat:@"%i", indexPath.row]];
    return cell;
}
```

UITableViewDelegate

- Actually has zero required methods
- About 16 optional methods
- A lot based around creating new rows, deleting rows, reordering rows, etc.
- Also handles row height and indentation level
- And handles finger presses
 - One method for when user selects a row
 - One method for when the user is *about* to select a row, and you get a chance to say yes or no
 - This is the only one PrimeGame implements

UITableViewDelegate

```
- (NSIndexPath *)tableView:(UITableView *)tableView \
willSelectRowAtIndexPath:(NSIndexPath *)indexPath {

    NSInteger num = indexPath.row;
    NSInteger divisor = isNotPrime(num);
    if (divisor) {
        //Show alert box
    }
    else {
        UITableViewCell *cell = [tableView \
cellForRowAtIndexPath:indexPath];
        [cell setAccessoryType:UITableViewCellAccessoryCheckmark];
    }
    return nil;
}
```

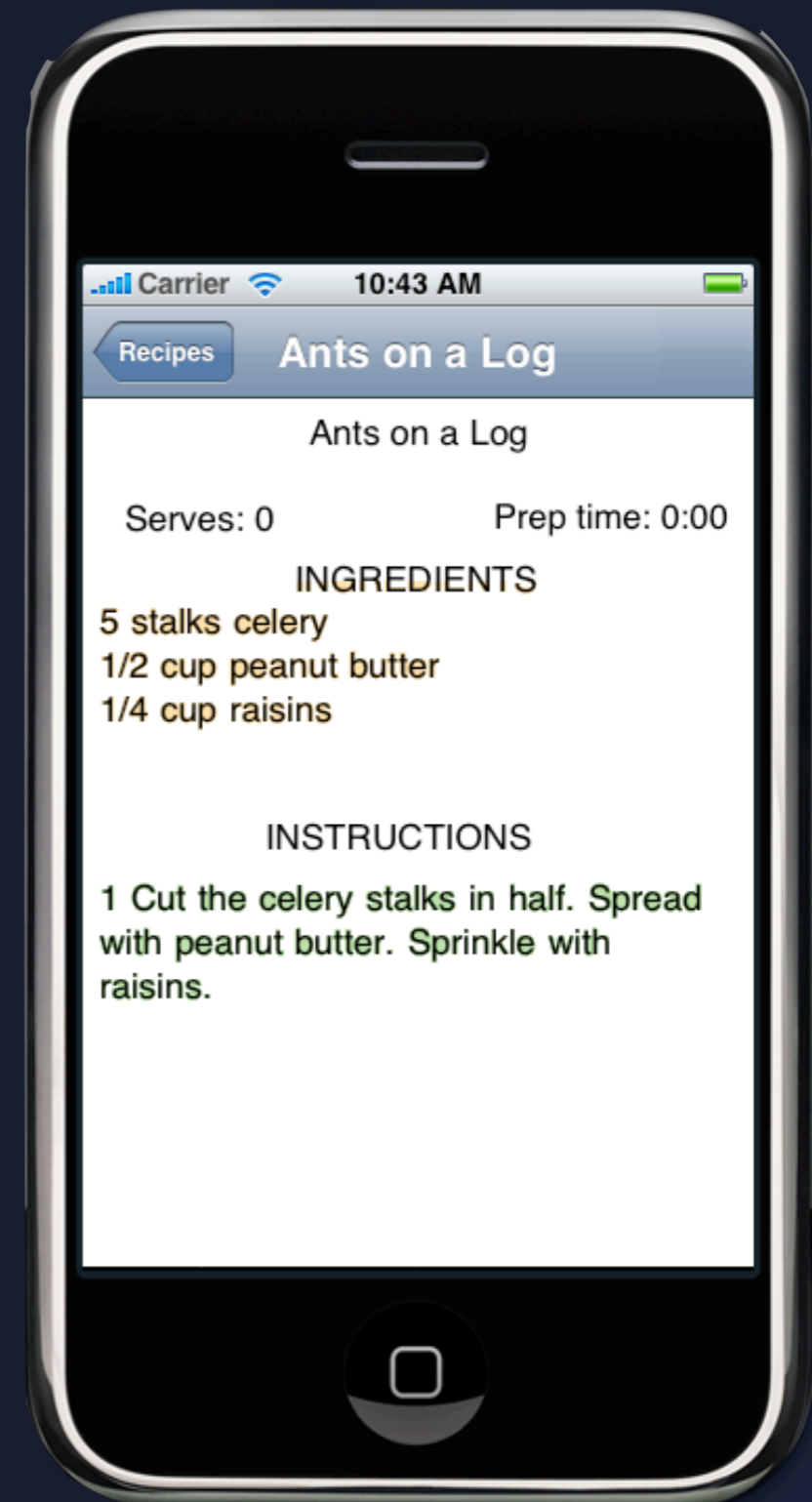
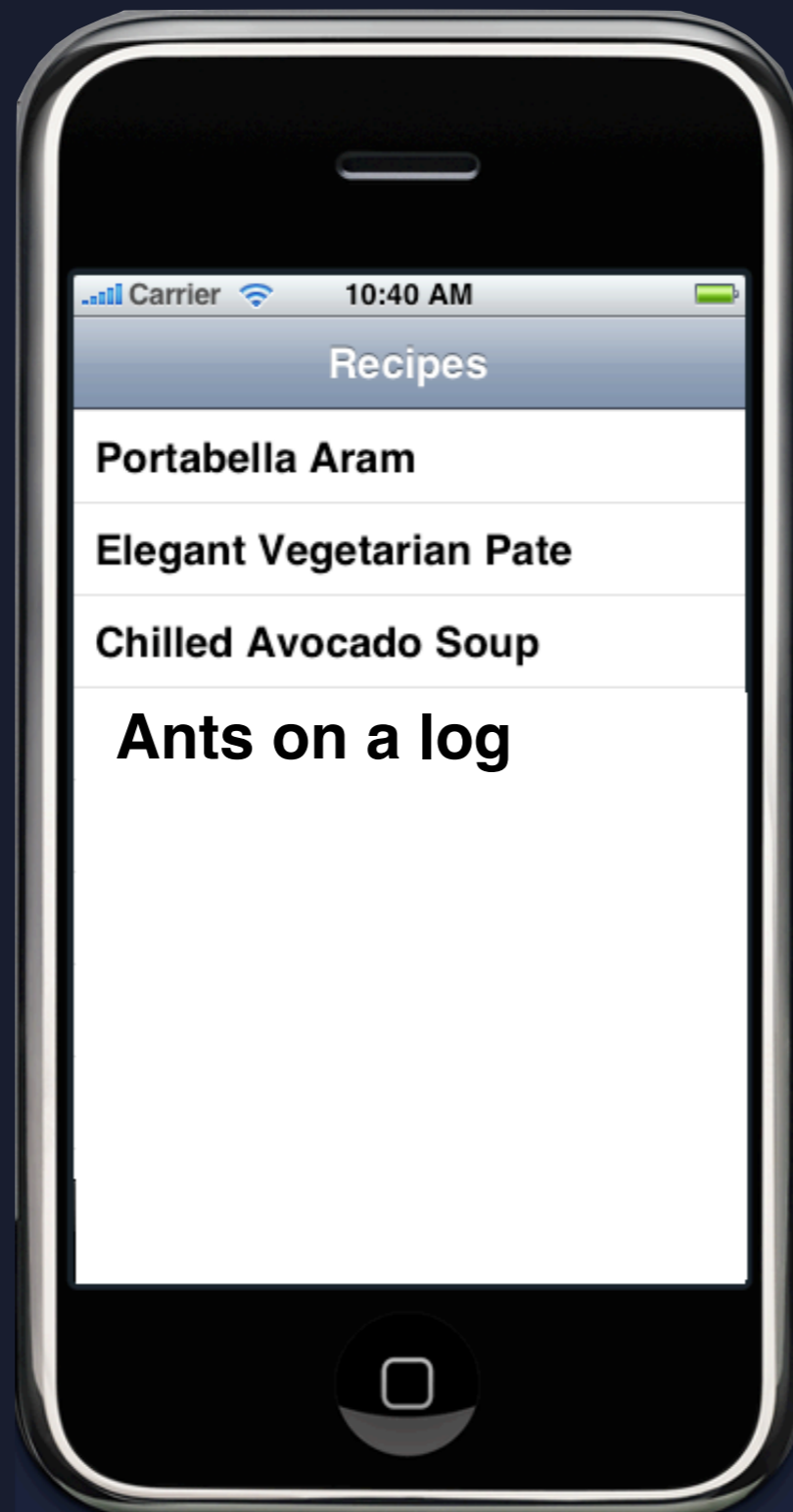
Let's talk about NURecipe part I.

I'm determined to not give you more code for you to fill in, but we should talk more in English about one way to set it up.

This is not at all the only way to do it. (I'm hoping some of you guys do it in a totally different way.)

But this is a fine way to do it.

NURecipe Part I



NURecipe Pragmatics

- There's a really good everything-related-to-tables document in the documentation
- You are (probably) going to need to stuff a `UITableViewController` into a `UINavigationController`
- For Part 1, you will need to handbuild a few recipes to fill the `UITableView`
- In Part 2 and beyond, these will be pulled from the database

Objective-C

```
-(Recipe *)getCheapestRecipe (NSArray *)recipes {  
    for (Recipe *recipe in recipes) {  
        //do stuff  
    }  
    //Choose easy recipe and return it  
}
```

Java

```
public Recipe getCheapestRecipe (ArrayList recipes) {  
    for (Recipe recipe : recipes) {  
        //do stuff  
    }  
    //Choose easy recipe and return it  
}
```

Actionscript
Javascript

```
public Recipe getCheapestRecipe (recipes Array) {  
    for (var recipe:Recipe in recipes) {  
        //do stuff  
    }  
    //Choose easy recipe and return it.  
}
```

ObjC

```
-(void)doSomething {  
    NSArray *recipes;  
    //Build array of recipes  
    Recipe *suggestedRecipe = [self getCheapestRecipe:recipes];  
}
```

Java

```
public void doSomething() {  
    ArrayList recipes;  
    //Build recipes  
    Recipe suggestRecipe = this.getCheapestRecipe(recipes);  
}
```

AS
JS

```
public void doSomething() {  
    var recipes:Array;  
    //Build recipe list  
    var suggestedRecipe:Recipe = this.getCheapestRecipe(recipes);  
}
```