# Tables

PROG31975 – Week 4

Jawaad Sheikh

Jawaad.Sheikh@SheridanCollege.ca

# Outline

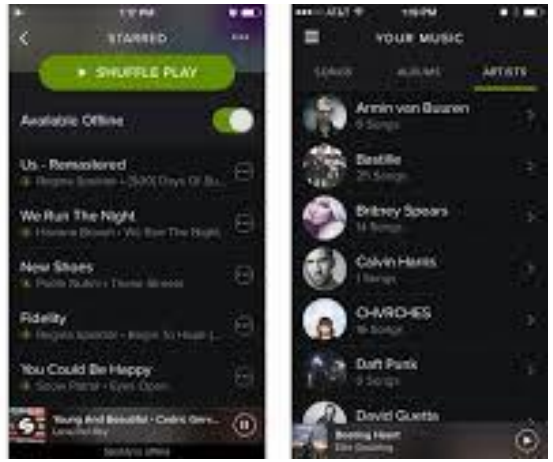
- Introduction
- Table Objects.
- How Tables Work.
- Delegate Methods.
- Table Editing Actions
- Exercise 1.
- Cell Event Handling.
- Customizing Table Cells.
- Exercise 2.

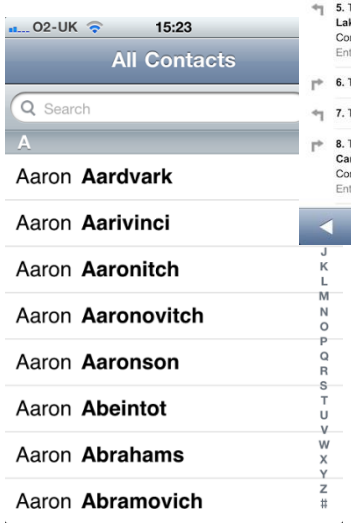
# Introduction

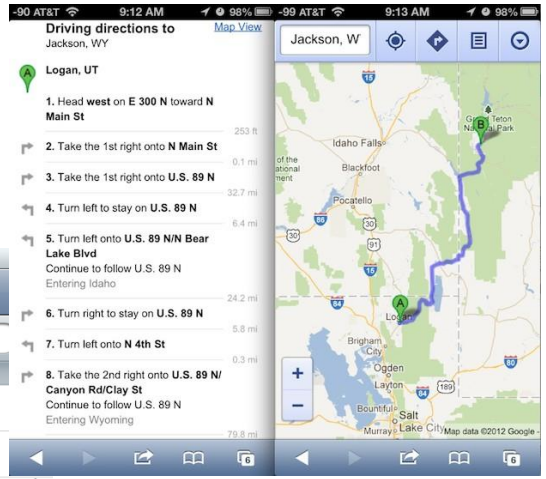
- Tables (aka TablesView's) are one of the most important objects to learn in iOS app development.
- In the Android world, known as ListView's.
- Tables are everywhere in almost every app you can think of.

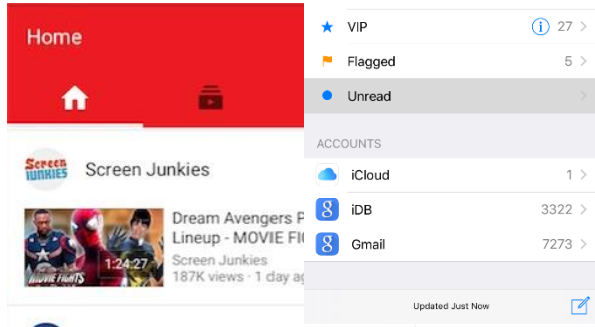
# Introduction

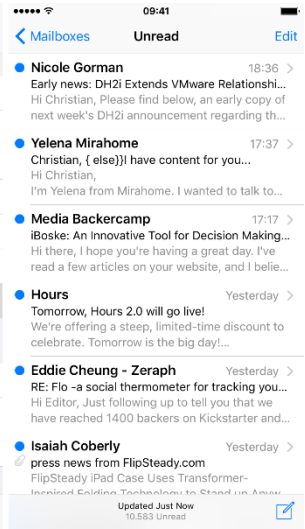
- Tables can be customized to look any way you want.



# Introduction

- Tables have many uses:
  - To let users navigate through data.
  - To present an indexed list of items.
  - To display detail information and controls in visually distinct groupings.
  - To present a selectable list of options.

# Table Objects

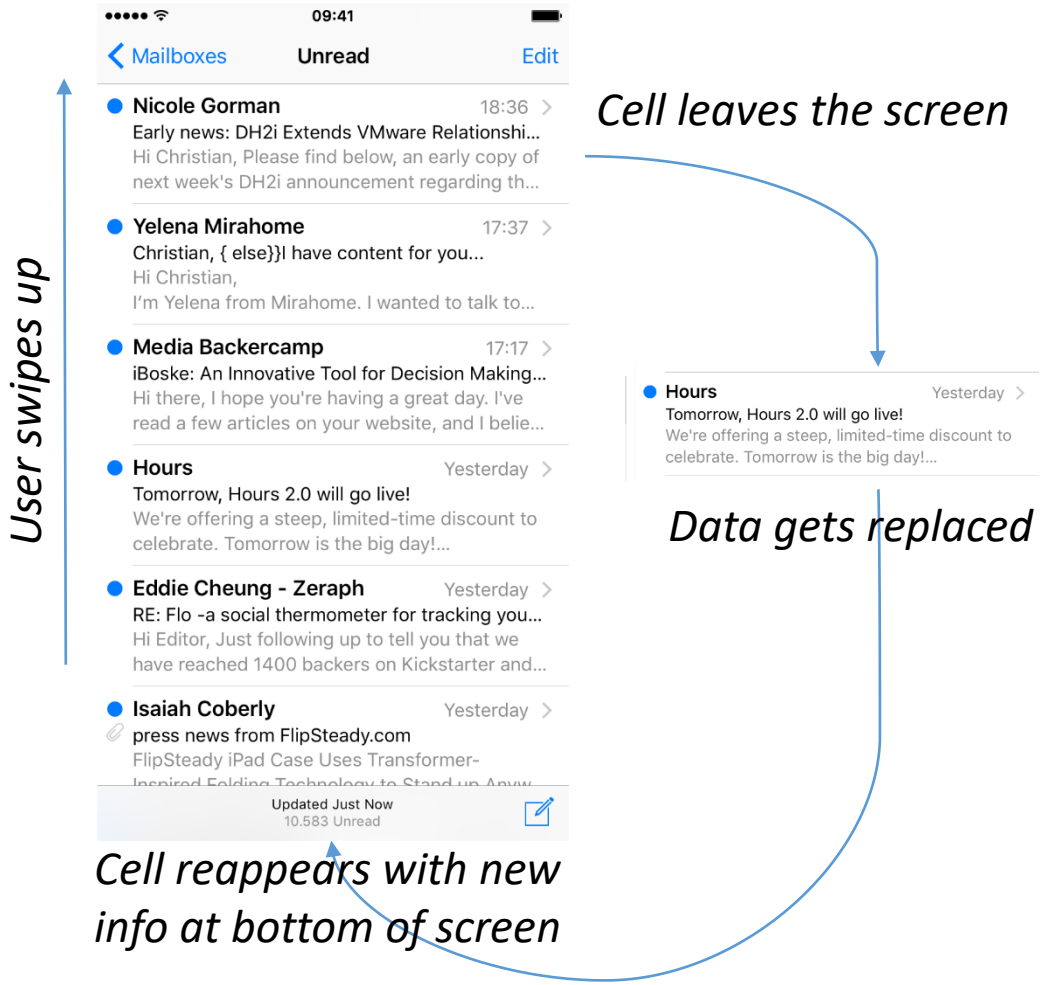
- UITableView – Storyboard object that will display a TableView.
- UITableViewController – ViewController object (subclass of UIViewController) enhanced for Table use.
- UITableViewCell – Object used to customize each cell in a table.

# How Tables Work

- Each row in a table is called a cell.
- Typically each cell is paralleled with an array.
- Data supplied to each cell comes from an index in the array.
- Question: How many cells are instantiated for an array of 1000 items?

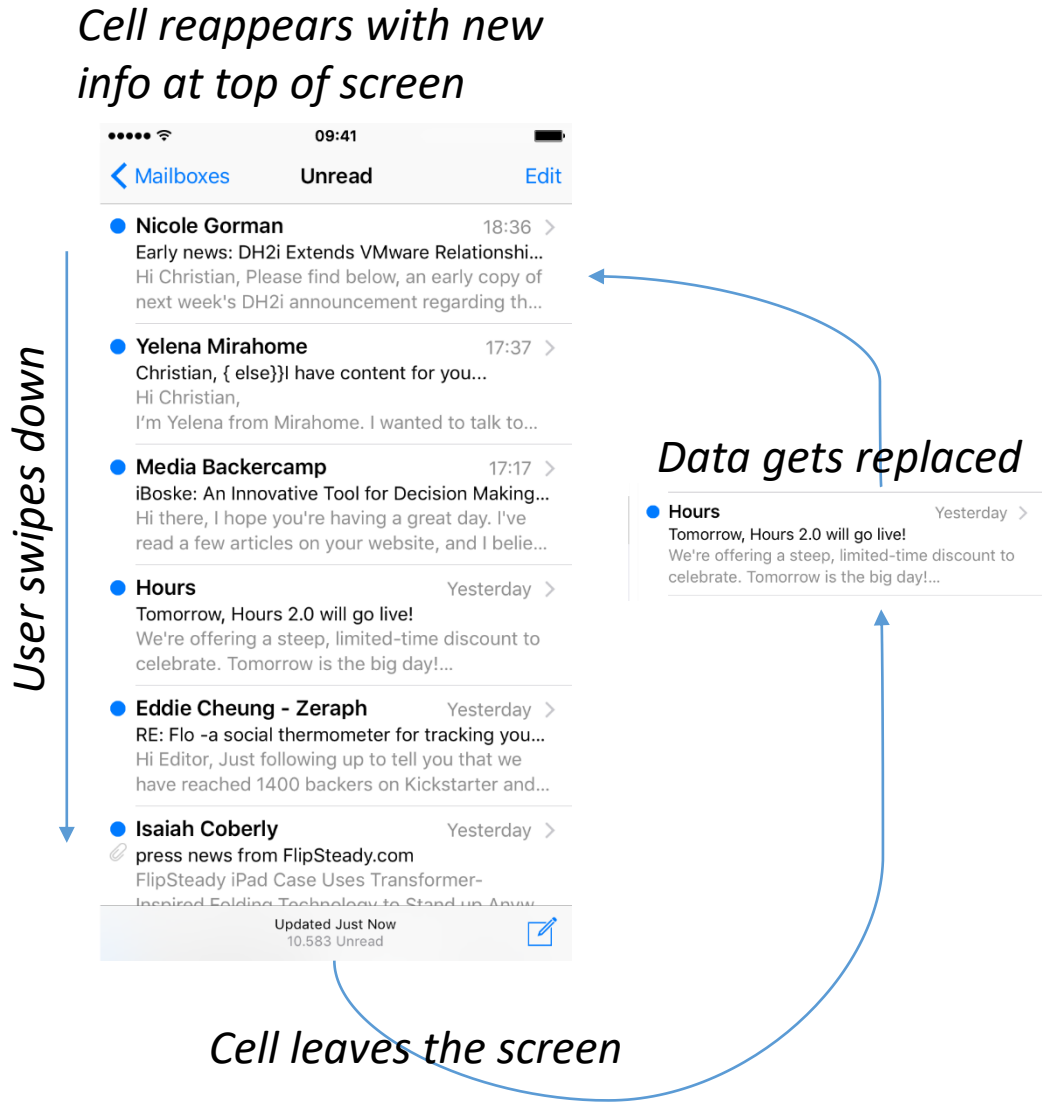
# How Tables Work

- Answer: How every many will fit on the screen and no more.
- At viewDidLoad() time, TableView will instantiate and display however many Cells will display on screen.



# How Tables Work

- A swipe up or down will result in a cell leaving the screen, having its data replaced and reappearing on the other side



# Table Delegate Methods

- To bring a table into a View Controller you need to implement the following:
  - UITableViewDelegate
  - UITableViewDataSource
- This delegate has a wide variety of support methods to fine tune your TableView.

# Table Delegate Methods

- However there are 3 support methods you need to bring in and implement to get a table to appear properly.
  - numberOfRowsInSection
  - heightForRowAt
  - cellForRowAt

# Table Delegate Methods

- numberOfRowsInSection – number of cells to work with (typically array length)
- heightForRowAt – not mandatory but for looks, how many pixels high will each cell be
- cellForRowAt – what to put in each cell

# Table Delegate Methods

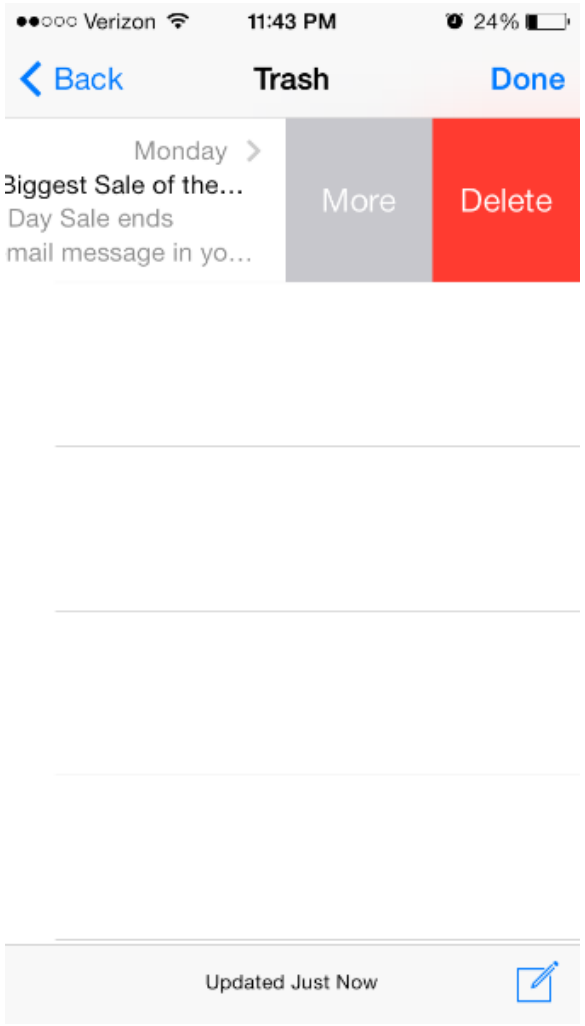
- While the first two methods are executed at view load time, the cellForRowAt method is unique.
- It is constantly being called.
  - At first it will be called for each cell that can be displayed on screen.
  - Then it will be called each time a cell leaves the screen.
  - The variable indexPath will tell you which row to work with.

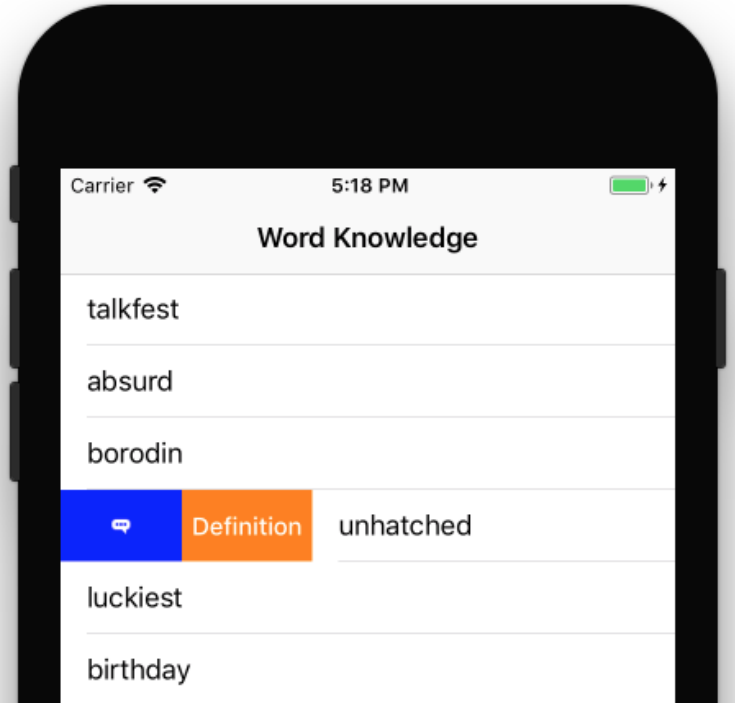
# Table Delegate Methods

- cellForRowAt is complex and involves 4 steps.
  - Check if cell is leaving the screen, do nothing – otherwise instantiate a new cell.
  - Populate the cell
  - Configure the cell
  - Return the cell

# Table Editing

- More and more apps are taking advantage of the ability to swipe left and right for options at each cell:



# Table Editing Methods

- If you only want to do swipe right actions then the editActionsForRowAt method is fine.
- Each option is a variable of type UITableRowAction
- Ex.

let delete = UITableViewRowAction(style: .normal, title: “Delete”) { action, index in …. }

# Table Editing Methods

- For left swipe, the leadingSwipeActionsConfigurationForRowAt method is needed
- In which you will need a variable of type UIContextualAction()
- Ex.

```
let modifyAction = UIContextualAction(style: .normal, title: “Modify”, handler: …. )
```

# Exercise 1

- Create an iPhone app with a home page and a sub page.
- The home page will only have a button to navigate to the sub page.
- On the sub page, create a table that will display 5 Toronto sports teams (Jays, Leafs, Raptors, Marlies & FC)
- Enable left and right swiping action where
  - Left swipe has “Modify”
  - Right swipe has “Share”, “Favourite” and “More”

# Table Event Handling

- Clicking on a table cell, by default, will do nothing.
- There is a fourth delegate method to implement.
  - didSelectRowAt
    - The variable indexPath will tell you which row number was clicked

# Customizing Table Cells

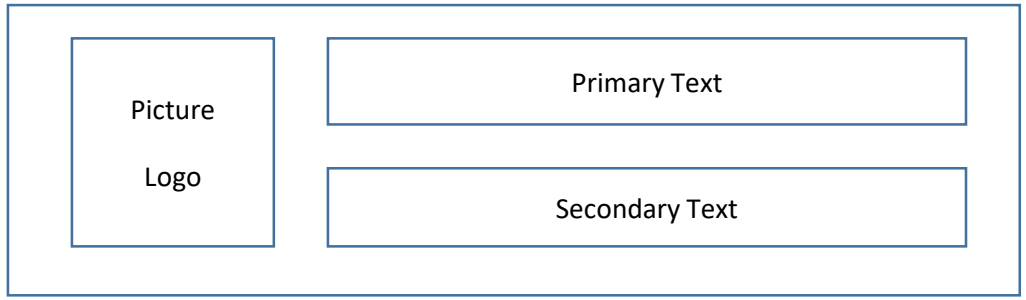
- Creating your own table cell involves adding a custom class of UITableViewCell
- Two ways to implement:
  - Define your cell using the Storyboard
  - Define your cell using code – we will use code here.

# Customizing Table Cells

- In your custom UITableViewCell, define all the items that will exist (labels, images, etc) – without IBOutlet
- Use init(style: reuseIdentifier: ) to instantiate / define properties of above items.
- Use layoutSubviews() to define size and location of items.
- See SiteCell.swift for implementation details.

# Customizing Table Cells

- In SiteCell.swift, we are implementing the following cell look:
  - Primary Text
    - Standard 30 font, black
    - x,y location = (100,5),  width = 460, height = 30
  - Secondary Text
    - Standard 12 font, blue
    - x,y location = (100,40), width = 460, height = 20
  - Picture logo
    - x,y location = (5,5), width = 45, height = 45



# Exercise 2

- Create an iPhone app with a home page and 2 sub pages.
- The home page will have a button to begin and redirect to the second page which will have a table view defining the Toronto sports teams.
- Use a custom cell to show logo, team name and team url.
- Add an event handler to click and redirect to a third page which will show a WebView displaying the website of the team selected.