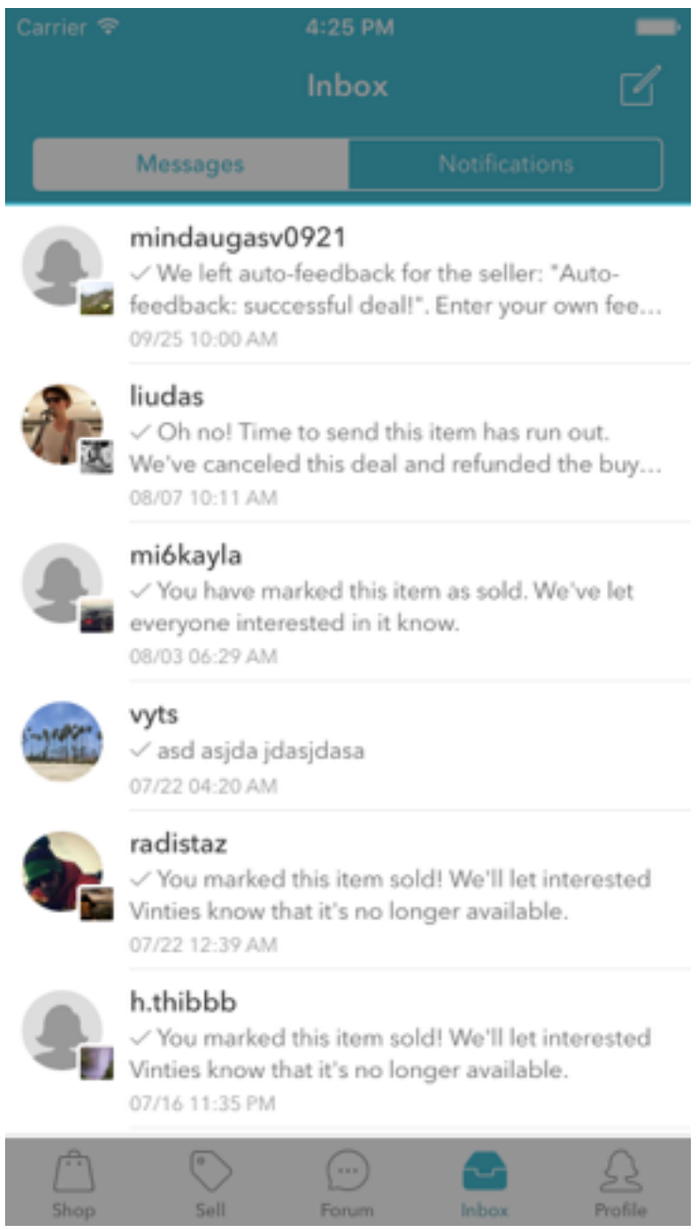
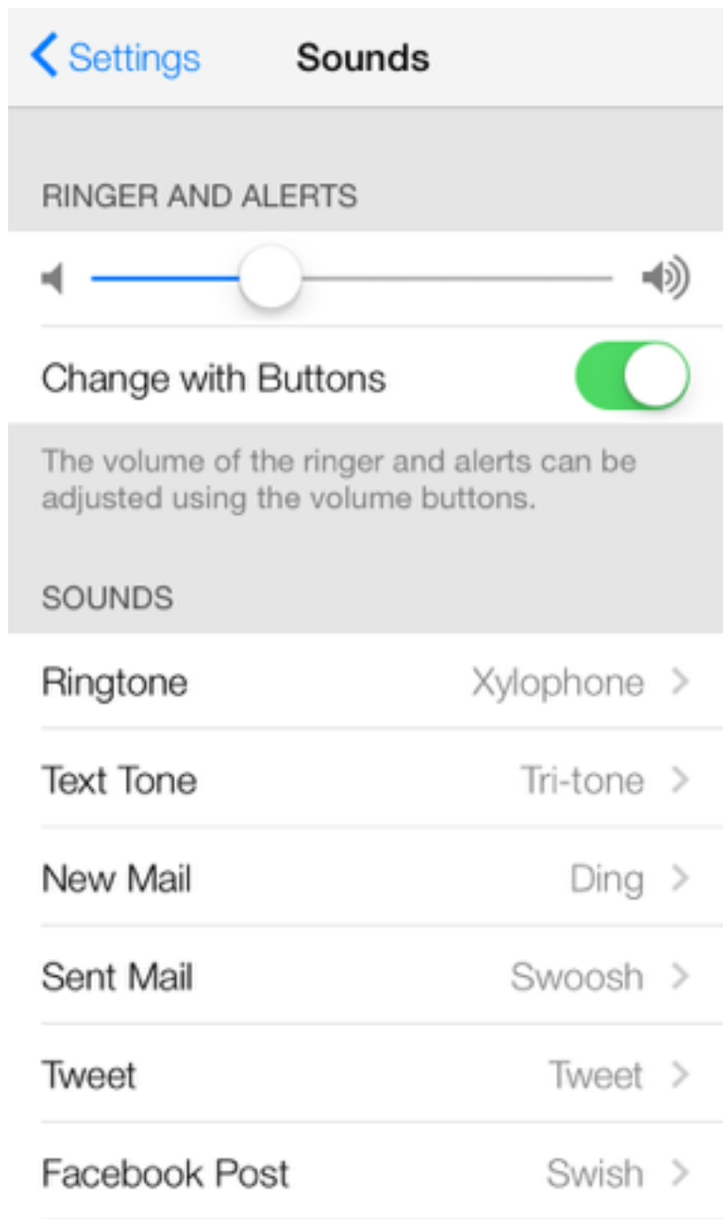
UITableView

UITableView

- Navigate through hierarchically structured data
- View an indexed list of items
- See detail information and controls in visually distinct groupings
- Interact with a selectable list of options

UITableView





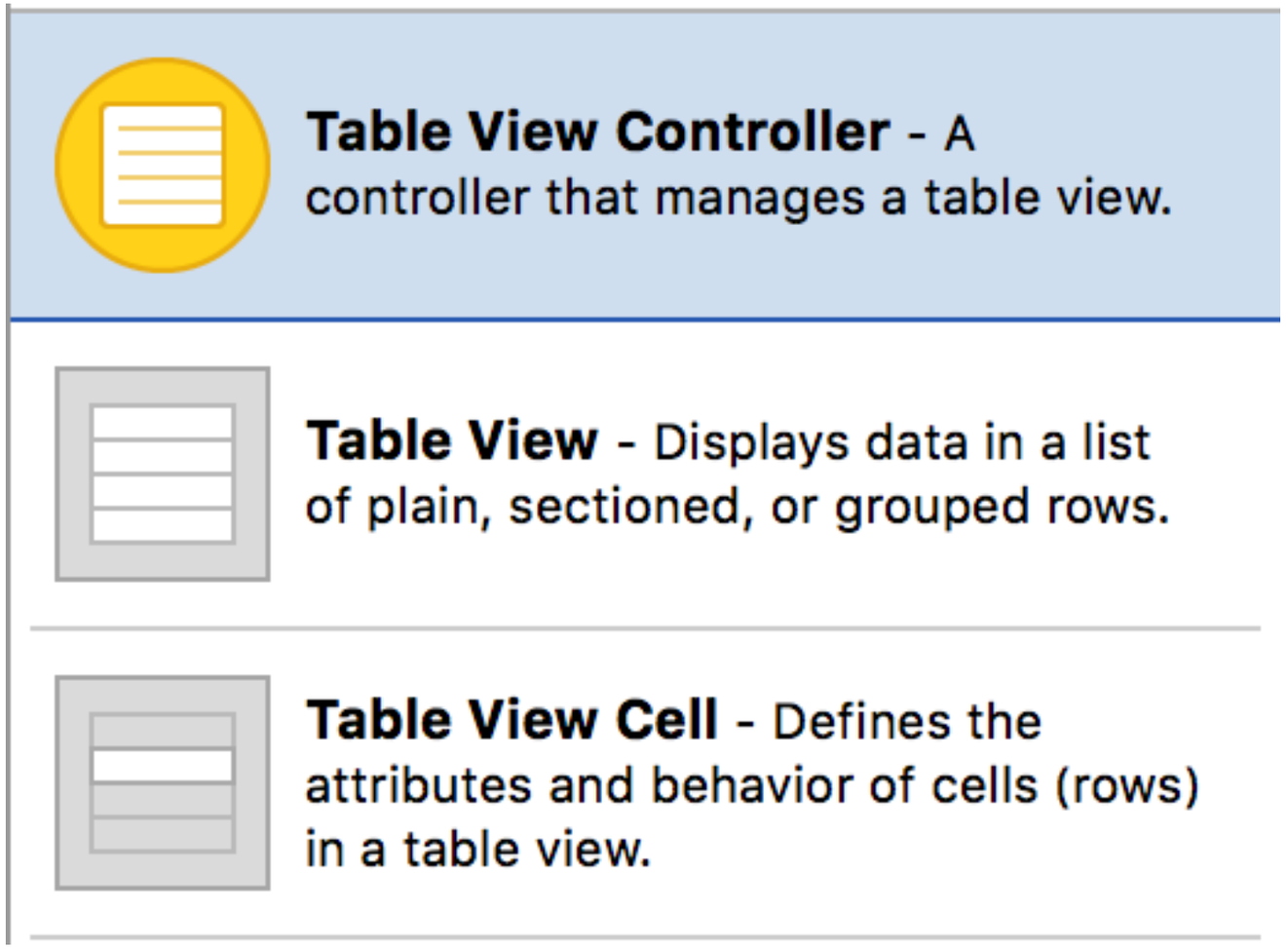
UITableView

In Code

```
let tableView = UITableView(frame: view.bounds);
view.addSubview(tableView);
```

UITableView

In Storyboard



UITableView

In Storyboard

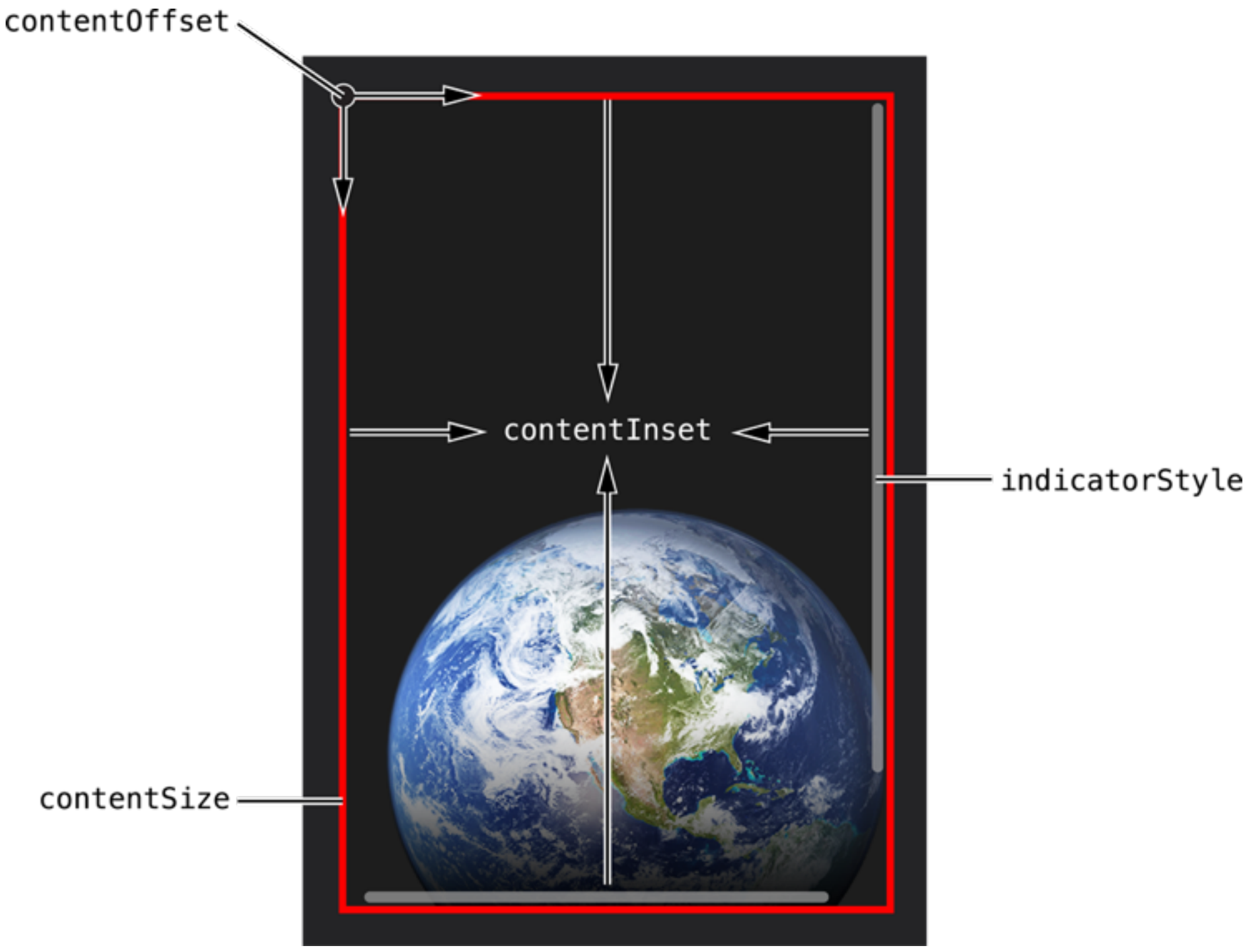
```
@IBOutlet weak var tableView: UITableView!
```

UIScrollView

- Used to represent more content than fits to the screen.

- Table View content can be bigger than screen size.

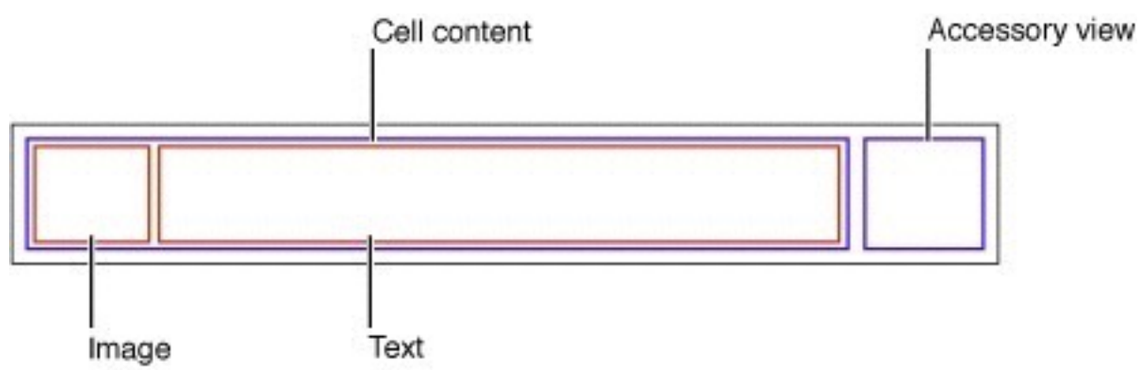
UIScrollView



ContentView in UITableView

- You provide all the info needed to calculate the contents of *UITableView*
- *UITableViewCell*, *UITableViewHeaderFooterView* are the main contents of *UITableView*
- *UITableView* also has *headerView* and *footerView* properties

UITableViewCell

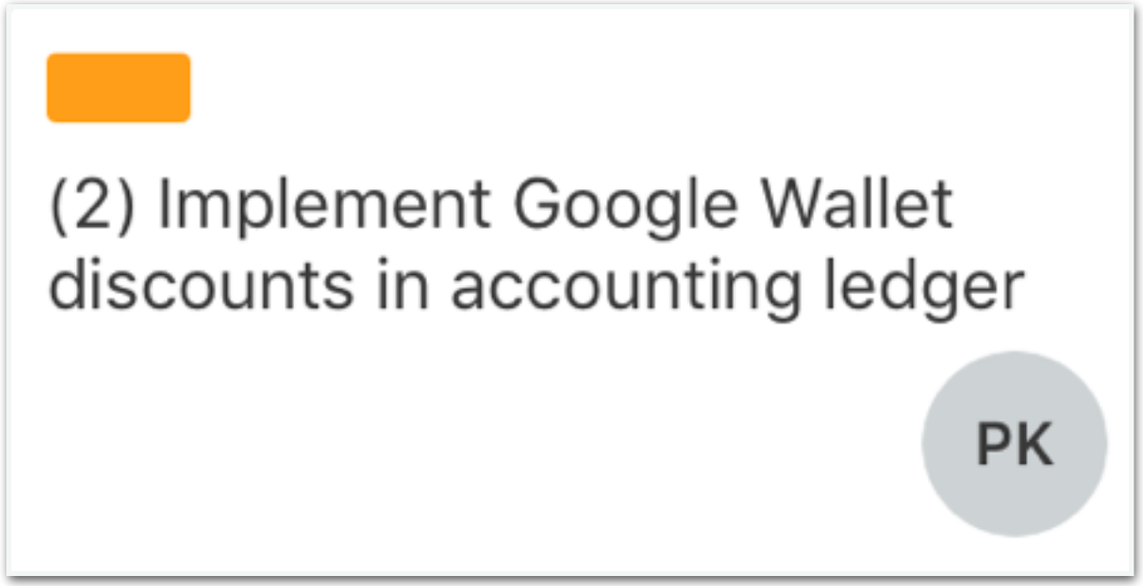


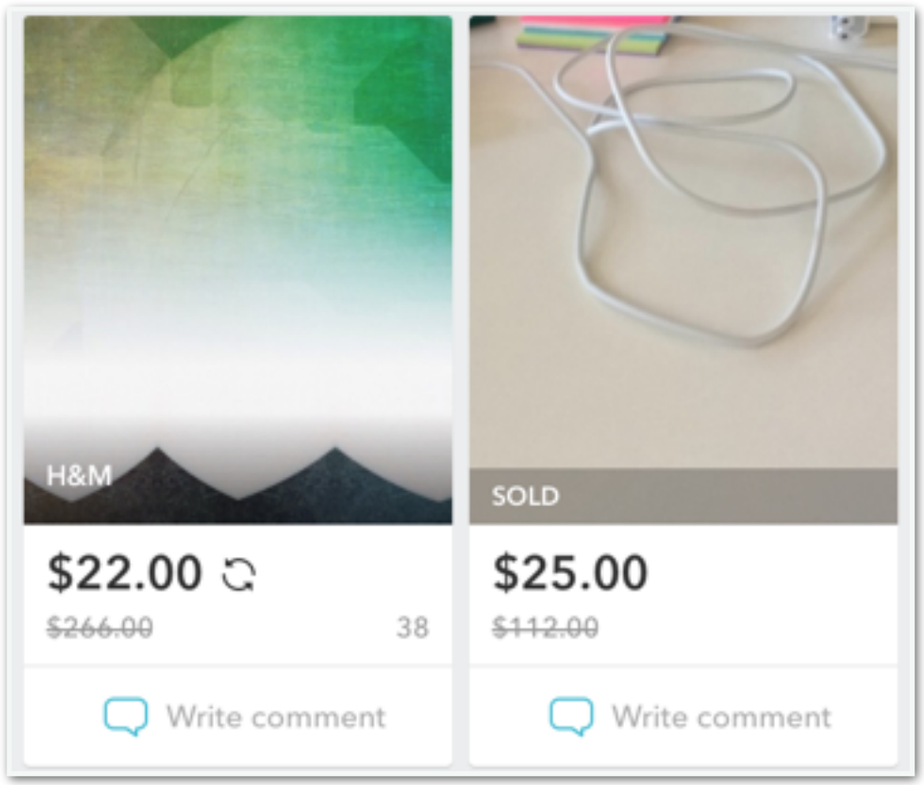
Default UITableViewCells

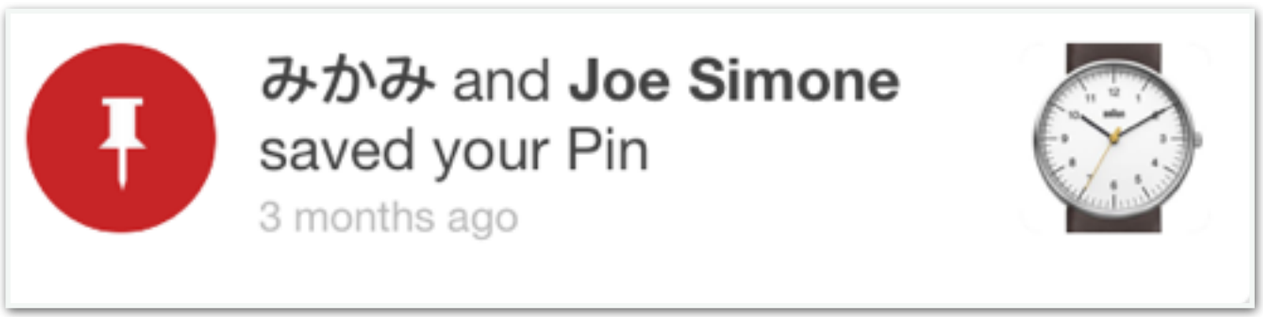
```
public enum UITableViewCellStyle : Int {

    case Default // Simple cell with text label and optional
image view (behavior of UITableViewCell in iPhoneOS 2.x)
    case Value1 // Left aligned label on left and right
aligned label on right with blue text (Used in Settings)
    case Value2 // Right aligned label on left with blue
text and left aligned label on right (Used in Phone/
Contacts)
    case Subtitle // Left aligned label on top and left
aligned label on bottom with gray text (Used in iPod).
}
```

Custom TableViewCells







Filling UITableView with data

- You have to set UITableView instance dataSource property with object implement UITableViewDataSource protocol

- That object provides all the needed data for table view

UITableViewDataSource

```
public protocol UITableViewDataSource : NSObjectProtocol {


    public func tableView(tableView: UITableView,
numberOfRowsInSection section: Int) -> Int

    public func tableView(tableView: UITableView,
cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCell



    optional public func numberOfSectionsInTableView(tableView:
UITableView) -> Int

    …
}
```

Sections

numberOfSectionsInTableView(_:)

```
func numberOfSectionsInTableView(tableView: UITableView) -> Int {
  return 2;
}
```

tableView(_:numberOfRowsInSection:)

```
func tableView(tableView: UITableView, numberOfRowsInSection
section: Int) -> Int {
        switch section {
        case 0:
            return allTitles.count
        case 1:
            return allNames.count
        }
    }
```

tableView(_:cellForRowAtIndexPath:)

```
func tableView(tableView: UITableView, cellForRowAtIndexPath
indexPath: NSIndexPath) -> UITableViewCell {
        let cell =
tableView.dequeueReusableCellWithIdentifier("NamesCellIdentifier",
forIndexPath: indexPath)
        switch indexPath.section {
        case 0:
            let name = allTitles[indexPath.row]
            cell.textLabel?.text = name as? String
        case 1:
            let name = allNames[indexPath.row]
            cell.textLabel?.text = name as? String
        default:
            break
        }
        return cell
}
```

Dequeuing

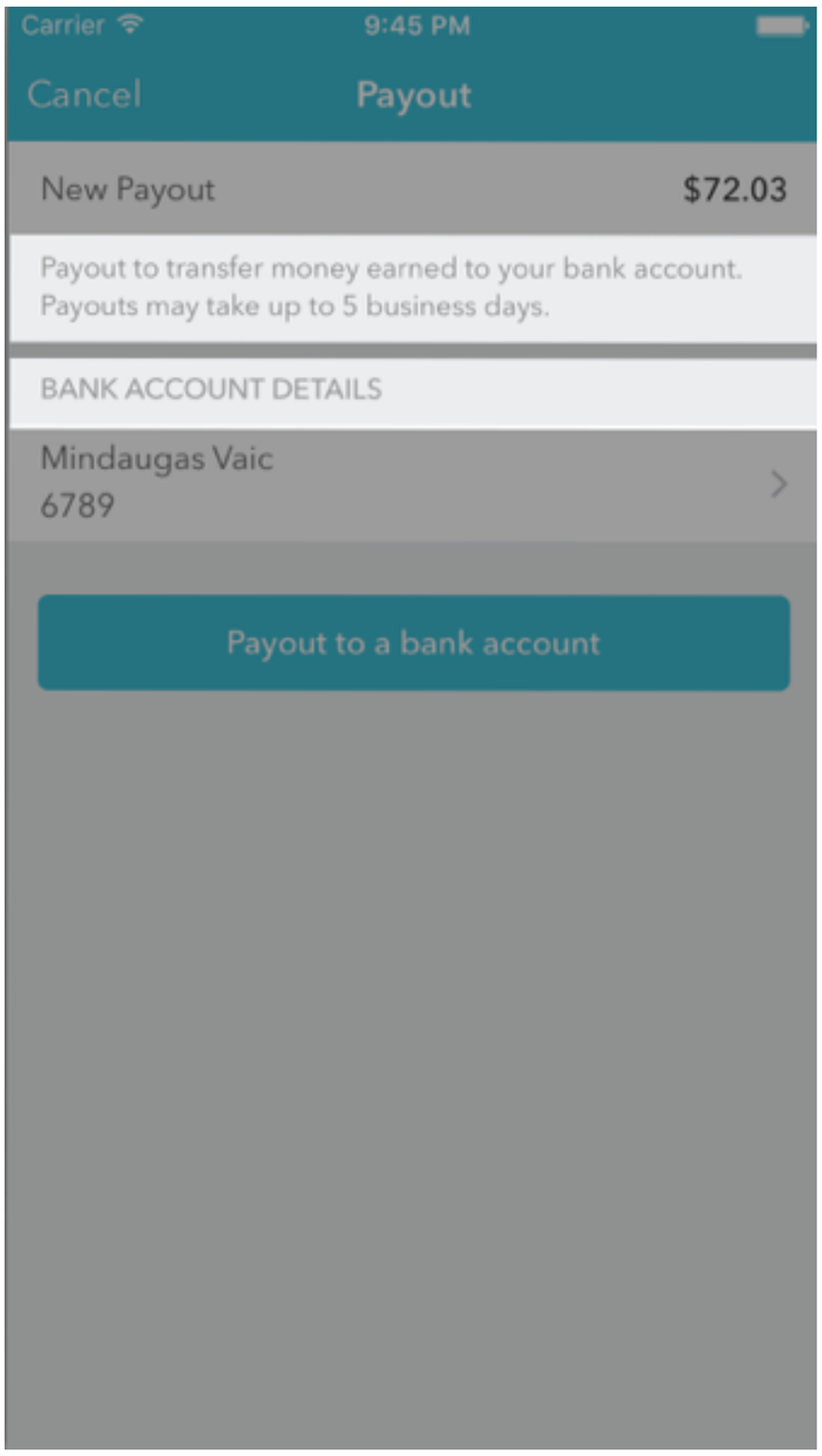
- *UITableView* tries to not create more cells than it is needed
- Each created cell can be reused for another data to represent
- *prapareForReuse()* is called when cell will be reused
- *reuseIdentifier* is used to identify cells for reuse

Register Cells

```
public func registerNib(nib: UINib?, forCellReuseIdentifier
identifier: String)


public func registerClass(cellClass: AnyClass?,
forCellReuseIdentifier identifier: String)
```

Section Footer and Header views



- Same as UITableViewCell
- Dequeued with reuse identifiers
- Can be created custom or just passed titles for them
- Each section can have different one or none

Register Section Footer and Header Views

```
public func registerNib(nib: UINib?,
forHeaderFooterViewReuseIdentifier identifier: String)
```

```
public func registerClass(aClass: AnyClass?,
forHeaderFooterViewReuseIdentifier identifier: String)
```

Table Footer and Header views

- *UITableView*'s can have footer and header views
- They are shown above or below table content
- Any *UIView* can be assigned to *tableHeaderView* and *tableFooterView*

Static table view content

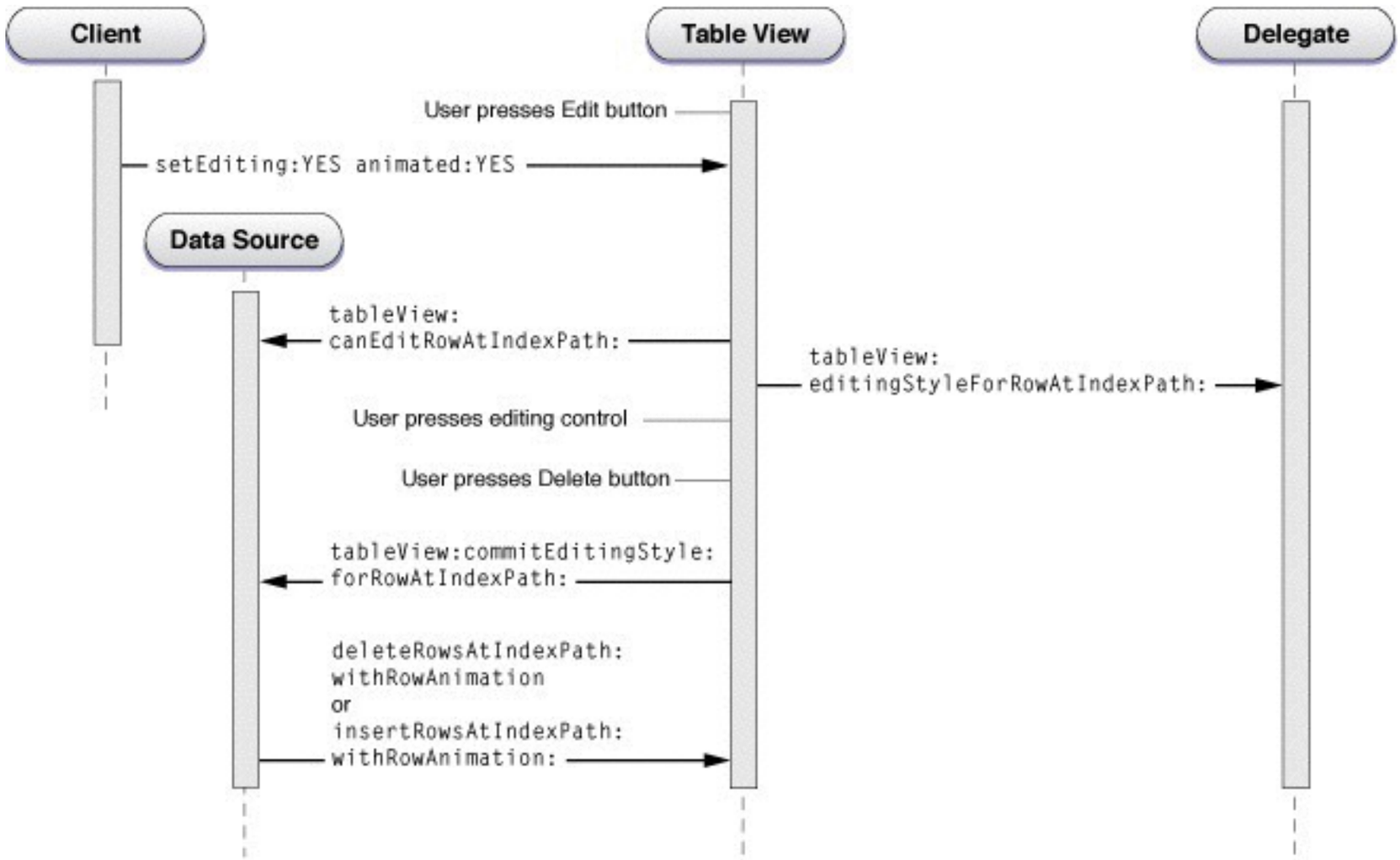
- Everything is configured in Interface builder
- You can link outlets directly from cells

Reacting to events in UITabelView

- All actions to which we can respond are sent to *UITableViewDelegate*

- Mainly we handle ***didSelect*** action

Edit mode


Client
Table View
Delegate
User presses Edit button
setEditing:YES animated:YES
Data Source
tableView:
canEditRowAtIndexPath:
tableView:
editingStyleForRowAtIndexPath:
User presses editing control
User presses Delete button
tableView:commitEditingStyle:
forRowAtIndexPath:
deleteRowsAtIndexPath:
withRowAnimation
or
insertRowsAtIndexPath:
withRowAnimation:

UITableViewController

- UITableView can be created in UIViewController or in UITableViewController.

- In UITableViewController view is UITableView.

- UITableViewController allows more possibilities in Interface builder and gives less setup out of the box.