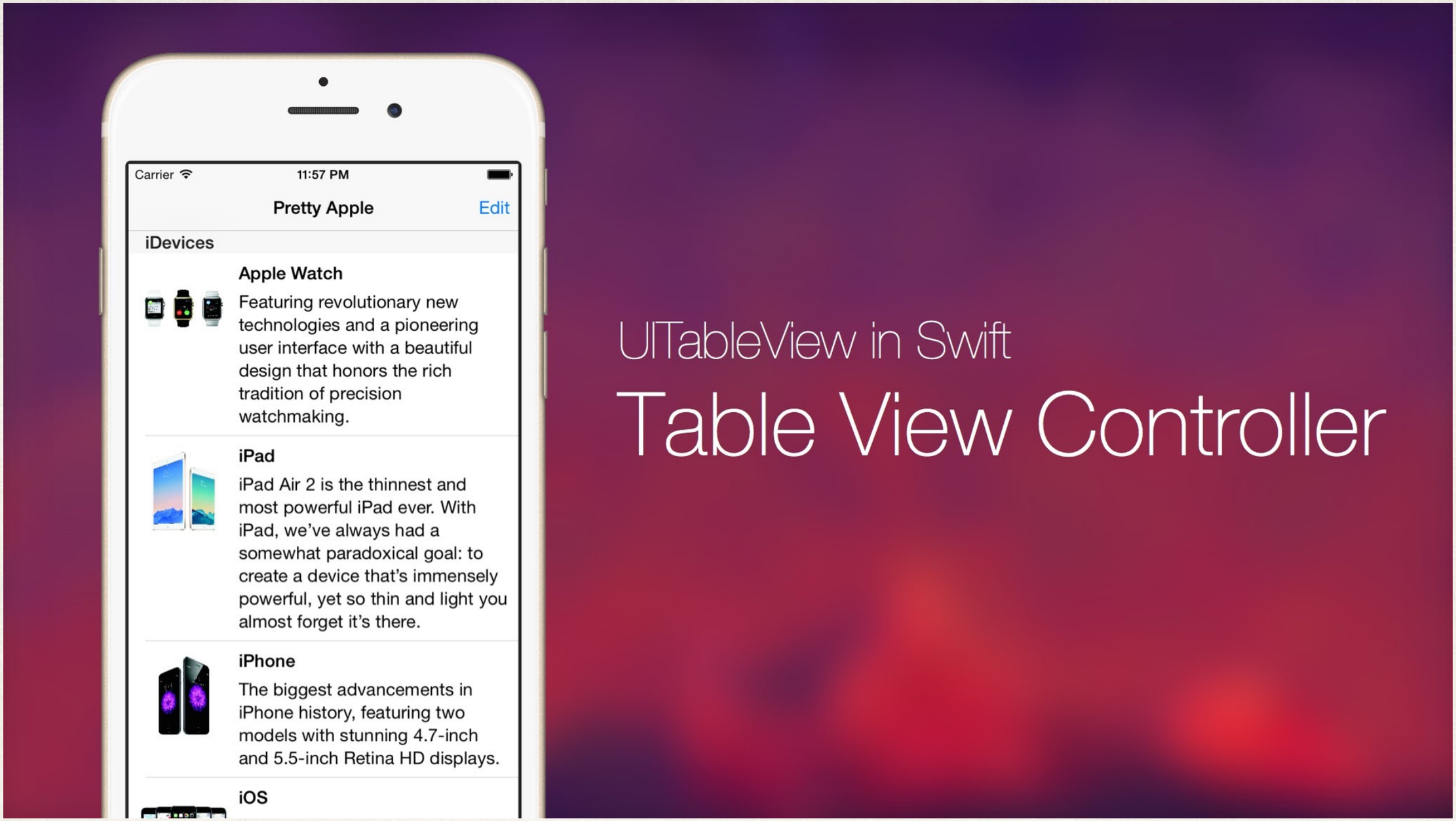


Table View Controllers

Dr. Sarah Abraham

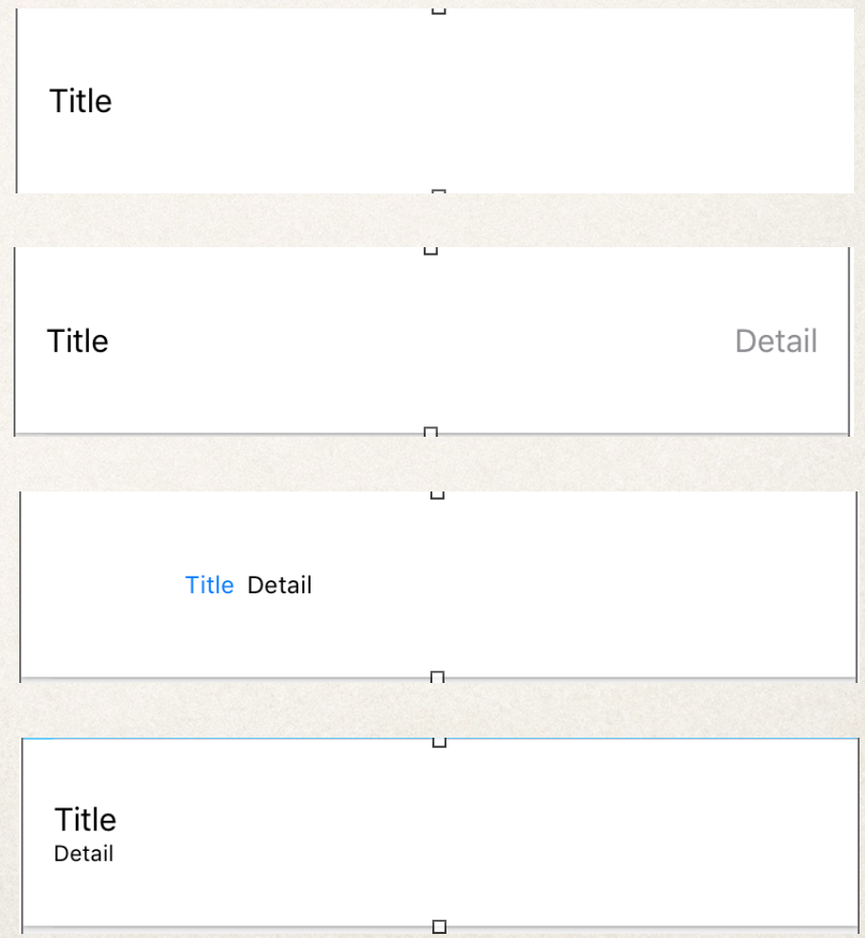
University of Texas at Austin
CS329e
Spring 2020

Using TableView Controllers

- Displays scrollable list of information in cells
- Useful for displaying compiled lists of data
- Information can come from a backend server or be stored in a data structure on the phone
- iOS handles the data list, programmer focuses on data display

Table View Cells

- Pre-defined table view cells allow for minimum coding to display basic data:
  - Basic
  - Right detail
  - Left detail
  - Subtitle

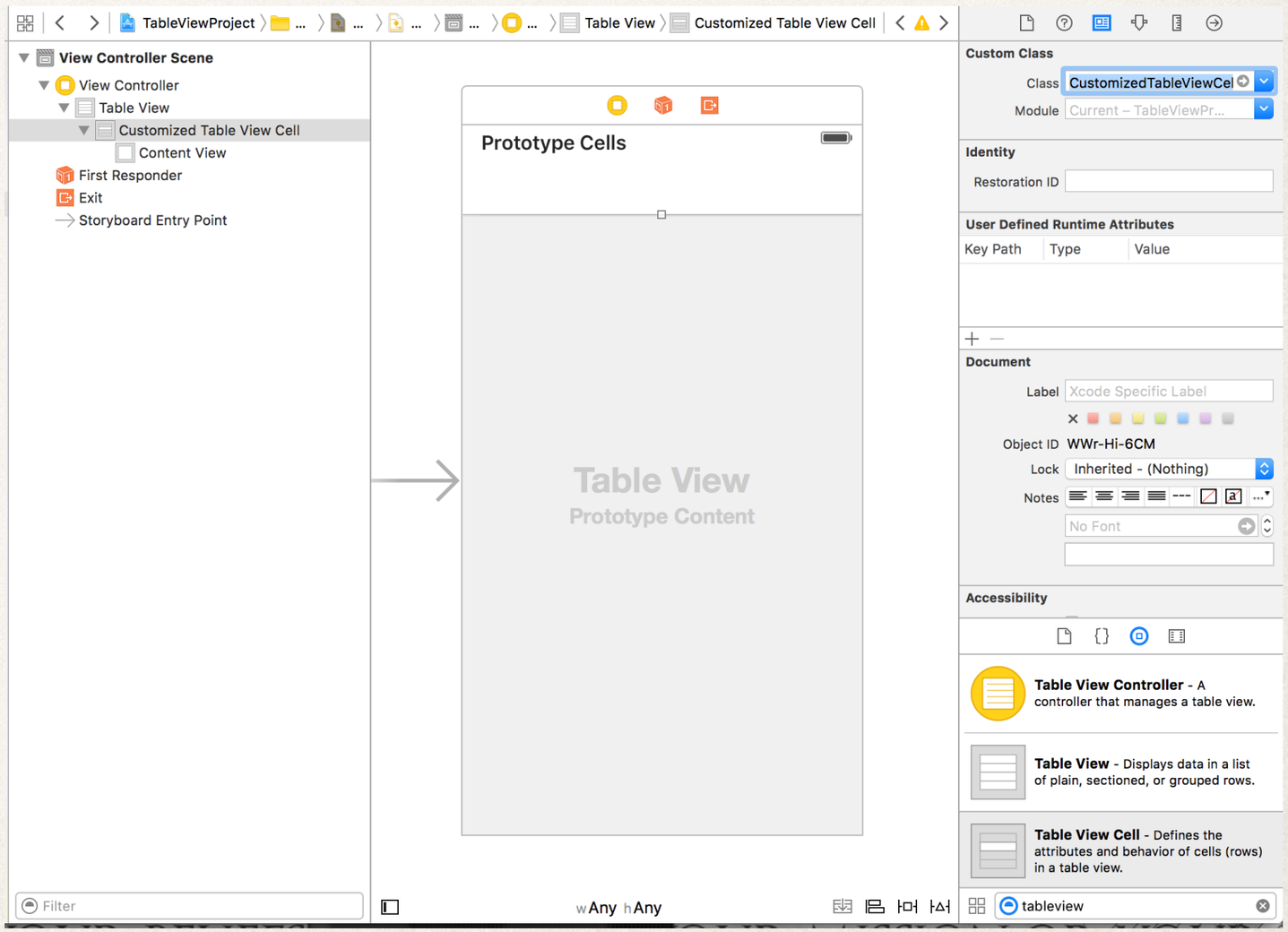


Custom Cells

- Complete customization possible
- View can contain any interface object
  - Labels, Buttons, Images, etc
- Multiple types of cells within a table also possible
- Main limitation is screen real estate

UITableViewCell

- Create subclass of UITableViewCell for custom cell
- Set table view cell to hold custom cell
- Build content of custom cell



Connect GUI Cell to Code

- Create outlets for IB objects in Xcode editor

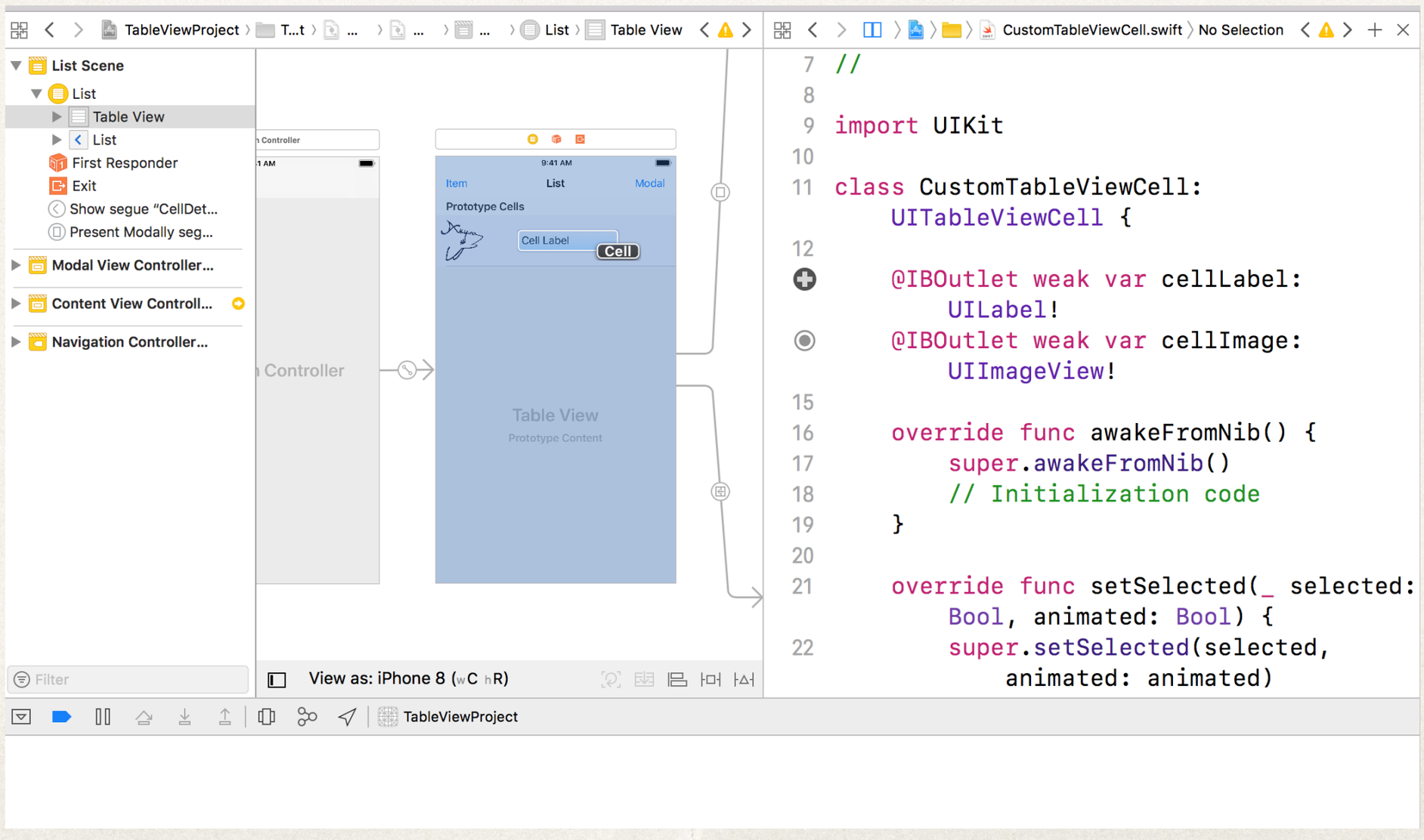


Table View Data Source Methods

- `func numberOfSections(in tableView: UITableView) -> Int`
  - Sets number of sections
- `func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int`
  - Sets number of rows
- `func tableView (_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell`
  - Sets cell content

Set Custom Cell Content

```
override func tableView(_ tableView: UITableView, cellForRowAt
indexPath: NSIndexPath) -> UITableViewCell {

    /* Set cellIdentifier to String that matches name of custom cell */
    let cellIdentifier = “CustomTableViewCell"

    /* Set display cell to custom cell down casting with the as!
    command: this means the program will crash if cast is unsuccessful!
    */
    let cell = tableView.dequeueReusableCell(withIdentifier:
    cellIdentifier, for: indexPath) as! CustomTableViewCell

    /* Perform additional calls to set the content of the cell */

    /* Return the cell for display within the table */
    return cell
}
```

Customize Cell Content

- Content for display in the table cells can come from various sources:
  - Dynamically loaded from external database
  - Placed by designers via plists
  - Hard-coded in the controller (**do not do this in practice!** This only works for toy examples!)
- Build out specialty class to store necessary information
- Index into correct content using `indexPath.row` in function `tableView`'s `cellForRowAt` parameter

Example of Cell Content

```
override func tableView(tableView: UITableView,
cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCell {

    let cellIdentifier = “CustomTableViewCell"
    let cell =
    tableView.dequeueReusableCellWithIdentifier(cellIdentifier,
    forIndexPath: indexPath) as! CustomTableViewCell

    /* content is array that holds model’s data; indexPath.row is
    based on which cell row the table is building out for display */
    cell.cellLabel.text = content[indexPath.row]

   return cell
}
```

UITableViewDelegate Methods

- Examples:
  - Programmatic row configuration
  - Row selection and deselection (`didSelectRowAt`/`willSelectRowAt`)
  - Table View header and footer content and appearance (`viewForHeaderInSection`)
  - Row editing (`willBeginEditingRowAt`)
  - Row reordering and removal (`didEndDisplaying`)
  - Highlighting/Focus (`shouldHighlightRowAt`)

Quiz Question!

- How many methods are required to implement the UITableViewDataSource protocol?
  - 0
  - 1
  - 2
  - 3
  - 4
  - 5

Navigating with Table Views

- Possible to embed TableViewController within a NavigationController
  - Allows navigation controller to manage the table view controller via its navigation stack
- Navigation controller adds a navigation bar
  - Optional two buttons (left and right)
  - Optional title (center)

Embedded Table View Controller

- Editor -> Embed In -> Navigation Controller