MAP523/DPS923 Table View Introduction

Introduction to Table Views (adding sections):

Modify the project you created earlier, except replace the original ViewController.swift file with the following:

```
//
//  ViewController.swift
//
//  Created by Danny Abesdris on 2016-02-10.
//  Copyright  2016 Danny Abesdris. All rights reserved.
//

// Modified ViewController.swift that adds private methods and allows
// table indexing by section.

import UIKit
// The "MARK :" "comments" (actually preprocessor features) below are used to organize
// your code.
// These preprocessor features allow you to bring some structure to the function drop
// down box of the source code editor.

// MARK:
// MARK: Connecting Data Source and Delegate
/* A table view is supposed to have a data source and a delegate.
   The data source and delegate objects of the table view need to conform to the
   UITableViewDataSource and UITableViewDelegate protocol respectively.
   Protocols are listed after the superclass of the class, and multiple protocols are
   separated by commas (see line 23 below)
*/
class ViewController: UIViewController, UITableViewDataSource, UITableViewDelegate {

   var langs: [String] = [ ]                       // an array of String objects
   let cellIdentifier = "CellIdentifier"           // identifier of prototype cell
   var alphabetizedLangs = [String: [String]]( )   // a dictionary of String arrays
                                                   // (i.e. key=String, value=[String])

   override func viewDidLoad( ) {

      /* The UIViewController class, the superclass of the ViewController class,
         also defines a viewDidLoad( ) method. The ViewController class overrides the
         method defined by the UIViewController class. This is indicated by the overri
         In order to ensure that the base class viewDidLoad( ) method is invoked, the
         is used to call the super class's viewDidLoad( ) method.
      */

      super.viewDidLoad( )

      // initializing the "langs" array with hardcoded data
      langs = [ "C", "Perl", "Python", "C++", "Java", "Pascal", "Ruby", "Swift", "Javas
```

```
                "PHP", "Assembler", "Processing.js", "Prolog", "Smalltalk", "Objective-
                "Rust", "Basic", "Logo", "Lisp", "D", "Simula", "Haskell", "Go", "Forti
                "Ada", "Bash", "Algol 68", "AppleScript", "ActionScript", "COBOL", "Del

    // calling the private alphabetize( ) function to sort the array "langs" and
    // return a dictionary consisting of keys of uppercase letters with their
    // corresponding languages (i.e. A=>"Algol 68", AppleScript"; P="PHP, Prolog, Pyt

    alphabetizedLangs = alphabetize(langs)

    // Do any additional setup after loading the view, typically from a nib.
}

// Private utility function used to return the keys in the alphabetized languages pr
// Sort keys in the dictionary alphabetically (keys are: A, B, C, D, etc.)
private func getLangKeys( ) -> [String] {
    return alphabetizedLangs.keys.sort({ (a, b) -> Bool in a.lowercaseString < b.lowe
}

// Private utility function used to sort and return a dictionary containing an array
// of Strings where each dictionary key is an alphabetic character and each dictiona
// value is an array of languages beginning with that letter.
private func alphabetize(array: [String]) -> [String: [String]] {
    var result = [String: [String]]( )

    for item in array {
        let index = item.startIndex.advancedBy(1)
        let firstLetter = item.substringToIndex(index).uppercaseString

        if result[firstLetter] != nil {
            result[firstLetter]!.append(item)
        }
        else {
            result[firstLetter] = [item]
        }
    }

    for (key, value) in result {
        result[key] = value.sort({ (a, b) -> Bool in a.lowercaseString < b.lowercaseSt
    }
    return result
}

override func didReceiveMemoryWarning() {
    super.didReceiveMemoryWarning()
    // Dispose of any resources that can be recreated.
}

// MARK: -
// MARK: Table View Data Source Protocol Methods
/*
    Because the view controller was used as the data source object of the table view
    view gets its data from the view controller. The first piece of information the t
    requires from its data source is the number of sections it should display.
    The table view does this by invoking the numberOfSectionsInTableView(_:) method o
    This is an optional method of the UITableViewDataSource protocol. If the table vi
    doesn't implement this method, the table view assumes that it needs to display on
    But, "what is a table view section?" A table view section is a group of rows (whi
    by the first letter of name in each row for example).

    The numberOfSectionsInTableView(_:) method accepts one argument, tableView, which
```

```
    that sent the message to the data source object. This is important, because it a
    object to be the data source of multiple table views if necessary.
    In the code below, the numberOfSectionsInTableView(_:) method returns an alphabe
    of keys in the alphabetizedLangs dictionary which consists of the capital letter
    group of languages beginning with that letter.
*/
func numberOfSectionsInTableView(tableView: UITableView) -> Int {
    let keys = alphabetizedLangs.keys
    return keys.count
}


/*
    Now that the table view knows how many sections it needs to display, it asks its
    many rows each section contains. For each section in the table view, the table v
    source a message of tableView(_:numberOfRowsInSection:). This method accepts two
    view sending the message and the table view's section index containing its speci

    numberOfSectionsInTableView(_:) returns the number of languages contained in eac
    each section is delimited by an alphabetic key ('A', 'B', 'P', etc.).
    NOTE: This first required sorting the dictionary keys which was accomplised by t
          utility function getLangKeys( ).
          Sorting the array of keys is important, because the key-value pairs of a d
          unordered. This is a key difference between arrays and dictionaries and a
          overlooked or misunderstood.
*/
func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int {

    // Calling private utility function to get the array of [String] keys in the dic
    let sortedLangKeys = getLangKeys( )

    // This now sets "key" to the array of Strings in each dictionary key
    // (i.e. A=>"Algol 68", AppleScript" OR P=>"PHP, Prolog, Python", etc.)
    let key = sortedLangKeys[section]

    // Return the number of languages in each "section".
    return alphabetizedLangs[key] != nil ? alphabetizedLangs[key]!.count : 0
}

/*
    The next method required to ensure that the ViewController class conforms to the
    protocol is tableView(_:cellForRowAtIndexPath:). The table view expects the data
    ViewController instance, to return a table view cell for each row in the table v
    The tableView(_:cellForRowAtIndexPath:) method sends a message to its data sourc
    the table view cell of the row specified by indexPath, the second argument of th

    NOTE: This method contains an instance of the NSIndexPath class. The NSIndexPath
    path to a specific node in a tree of nested array collections. An instance of th
    or more indices. In the case of a table view, it holds an index for the section
    row of that item in the section.
    A table view is never more than two levels deep, the first level being the secti
    being the row in the section.
*/
func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath
    let cell = tableView.dequeueReusableCellWithIdentifier(cellIdentifier, forIndexP

    // Get and sort the keys
    let keys = getLangKeys( )

    // Capture languages for section at indexPath
    let key = keys[indexPath.section]
```

```
        if let langs = alphabetizedLangs[key] {
            // Get language
            let sectionLang = langs[indexPath.row]

            // Configure Cell in TableView
            /*
                The text of the textLabel property of the table view cell can now be set t
                the language at a specific row within each section. The UITableViewCell cl
                UIView subclass and it has a number of subviews. One of these subviews is
                UILabel which is used to display the name of the programming language in t
            */
            cell.textLabel?.text = sectionLang
        }
        return cell
    }

    /*
        To add sections to the Table View, the tableView(_: UITableView, titleForHeaderI
        UITableViewDataSource protocol must be implemented. Sections are based on the fi
        proramming language.
        To add sections to the table view, an array of names is not sufficient. Instead,
        broken up into sections with the languages in each section sorted alphabetically
        requires a dictionary of type: [String: [String]] which was declared in the View
        as: var alphabetizedLangs = [String: [String]]( )
        In viewDidLoad( ), the "langs" array is used to create a dictionary of languages
        contains an array of programming languages for each letter of the alphabet. For
        a language, that key is omitted from the dictionary.
        The dictionary is created with the help of a helper method, alphabetize(_:) whic
        "langs" array as an argument.

        In order to tell the table view that it should display a letter in each section
        tableView(_:titleForHeaderInSection:), method that is defined in the UITableView
        is now implemented. Note how it is similar to the implementation of the tableVie
        method in that it simply returns the number of keys in each sectionby first call
        utility method getLangKeys( ) which returns a sorted array of keys in the dictio
    */
    func tableView(tableView: UITableView, titleForHeaderInSection section: Int) -> Str
        // Get and sort keys
        var keys = getLangKeys( )
        /*
        for i in 0..<keys.count {
            keys[i] += " section..."
        }
        */
        return keys[section]
    }

    /*
        Delegation:
        In addition to the UITableViewDataSource protocol, the UIKit framework also defi
        protocol, the protocol to which the table view's delegate needs to conform.
        This was accomplished on the storyboard by setting the view controller as the de

        To implemet a method that respond to touch events (for example, whenever a user
        tableView(_:didSelectRowAtIndexPath:) is implemented and simply displays the tex
        language on the console.
        This is used to demonstrate how the delegate pattern works.
    */

    // MARK: -
```

```
    // MARK: Table View Delegate Methods
    func tableView(tableView: UITableView, didSelectRowAtIndexPath indexPath: NSIndexPat
        // Get and sort keys
        let keys = getLangKeys( )

        // Capture languages for section at indexPath
        let key = keys[indexPath.section]

        if let langs = alphabetizedLangs[key] {
            print(langs[indexPath.row]) // display the language for the specific row selec
                                        // the specific section
        }
    }
}
```

Run the application again.
Do you now notice that the programming languages have been indexed alphabetically and now include sections by first letter?