------------BELOW IS ITEMLISTVC CLASS-----

Import UIKit

Class ItemListVC: UIViewController

{  
@IBOutlet var dataProvider: ~~ItemListDataProvider! (~~UITableViewDataSource & UITableViewDelegate)! <-protocol

@IBOutlet var tableView: UITableView!  
 var itemManager: ItemManager?

Override func viewDidLoad()

{

Super.viewDidLoad()

tableView.dataSource = dataProvider

tableView = UITableView()  
 tableView.delegate = dataProvider

}

}  
------ABOVE IS ITEMLISTVC CLASS,

---------BELOW IS ITEMCELL CLASS----------

Import UIKit  
  
class ItemCell: UITableViewCell

{

func configCell(with item: ToDoItem) {

}

}

------ABOVE IS ITEMCELL CLASS---------

BELOW IS ITEMLISTDATAPROVIDER CLASS--------  
enum Section: Int {

case toDo

case done

}  
  
class ItemListDataProvider: NSObject, UITableViewDataSource, UITableViewDelegate

{

**func tableView(\_ tableView: UITableView,**

**numberOfRowsInSection section: Int) -> Int {**

~~return 0~~

~~return itemManager?.toDoCount ?? 0~~

guard let itemManager = itemManager else { return 0 }

guard let itemSection = Section(rawValue: section) else {

fatalError()

}

let numberOfRows: Int

switch itemSection {

case .toDo:

numberOfRows = itemManager.toDoCount

case .done:

numberOfRows = itemManager.doneCount

}

return numberOfRows

} }  
  
**func tableView(\_ tableView: UITableView,**

**cellForRowAt indexPath: IndexPath) -> UITableViewCell {**

return ~~UITableViewCell() ItmeCell()~~ ~~let cell = tableView.dequeueReusableCell(~~

~~withIdentifier: "ItemCell",~~

~~for: indexPath) as! ItemCell //added so class has member configCell  
 cell.configCell(with: ToDoItem(title: ""))//test for cell.configCellGotCalled is not true unless we create this~~

~~return cell~~

~~let cell = tableView.dequeueReusableCell(~~

~~withIdentifier: "ItemCell",~~

~~for: indexPath) as! ItemCell~~

~~if let item = itemManager?.item(at: indexPath.row) {~~

~~cell.configCell(with: item)~~

~~}~~

~~return cell~~

~~}~~

let cell = tableView.dequeueReusableCell(

withIdentifier: "ItemCell",

for: indexPath) as! ItemCell

guard let itemManager = itemManager else { fatalError() }

guard let section = Section(rawValue: indexPath.section) else

{

fatalError()

}

let item: ToDoItem

switch section {

case .toDo:

item = itemManager.item(at: indexPath.row)

case .done:

item = itemManager.doneItem(at: indexPath.row)

}

cell.configCell(with: item)

return cell  
  
 **func numberOfSections(**

**in tableView: UITableView) -> Int** {

return 2

}  
  
**func tableView(**

**\_ tableView: UITableView,**

**titleForDeleteConfirmationButtonForRowAt indexPath:**

**IndexPath) -> String?** {

return "Check"

}

**func tableView(**

**\_ tableView: UITableView,**

**titleForDeleteConfirmationButtonForRowAt indexPath:**

**IndexPath) -> String?**

{

guard let section = Section(rawValue: indexPath.section) else

{

fatalError()

}

let buttonTitle: String

switch section {

case .toDo:

buttonTitle = "Check"

case .done:

buttonTitle = "Uncheck"

}

return buttonTitle

}

**func tableView(\_ tableView: UITableView,**

**commit editingStyle: UITableViewCellEditingStyle,**

**forRowAt indexPath: IndexPath)** {

itemManager?.checkItem(at: indexPath.row)

tableView.reloadData()

}

---------ABOVE IS ITEMLISTDATAPROVIDER CLASS-----

--------BELOW IS ITEMLISTVIEWCONTROLLER TESTS-

@testable import ItemListVC

Var sut: ItemListViewController!

Func setup(){

let storyboard = UIStoryboard(name: "Main",

bundle: nil)

let viewController =

storyboard.instantiateViewController(

withIdentifier: "ItemListViewController")

sut = viewController as! ItemListViewController

sut.loadViewIfNeeded()

}

func test\_TableViewIsNotNilAfterViewDidLoad()

{

XCTAssertNotNil(sut.tableView)

}  
  
func test\_LoadingView\_SetsTableViewDataSource() {

XCTAssertTrue(sut.tableView.dataSource is ItemListDataProvider)

}

func test\_LoadingView\_SetsTableViewDelegate() {

XCTAssertTrue(sut.tableView.delegate is ItemListDataProvider)

}  
  
func test\_LoadingView\_DataSourceEqualDelegate() {

XCTAssertEqual(sut.tableView.dataSource as? ItemListDataProvider,

sut.tableView.delegate as? ItemListDataProvider)

}

ABOVE IS ITEMLISTVIEWCONTROLLER TESTS------  
-----------BELOW IS ITEMLISTDATAPROVIDER TESTS

var controller: ItemListViewController!

var sut: ItemListDataProvider!

var tableView: UITableView!

Func setup()

{

~~sut = ItemListDataProvider()~~

~~sut.itemManager = ItemManager()~~

~~tableView = UITableView()~~

~~tableView.dataSource = sut~~

super.setUp()

sut = ItemListDataProvider()

sut.itemManager = ItemManager()

let storyboard = UIStoryboard(name: "Main", bundle: nil)

controller = storyboard.instantiateViewController(

withIdentifier: "ItemListViewController") as!

ItemListViewController

controller.loadViewIfNeeded()

tableView = controller.tableView

tableView.dataSource = sut

tableView.delegate = sut

}

func test\_NumberOfSections\_IsTwo() {

let numberOfSections = tableView.numberOfSections

XCTAssertEqual(numberOfSections, 2)

}  
  
func test\_NumberOfRows\_Section1\_IsToDoCount() {

sut.itemManager?.add(ToDoItem(title: "Foo"))  
XCTAssertEqual(tableView.numberOfRows(inSection: 0), 1)

sut.itemManager?.add(ToDoItem(title: "Bar"))

XCTAssertEqual(tableView.numberOfRows(inSection: 0), 2)

}

func test\_NumberOfRows\_Section2\_IsToDoneCount() {

sut.itemManager?.add(ToDoItem(title: "Foo"))

sut.itemManager?.add(ToDoItem(title: "Bar"))

sut.itemManager?.checkItem(at: 0)

XCTAssertEqual(tableView.numberOfRows(inSection: 1), 1)

sut.itemManager?.checkItem(at: 0)

tableView.reloadData()

XCTAssertEqual(tableView.numberOfRows(inSection: 1), 2)

}  
  
func test\_CellForRow\_ReturnsItemCell(){

{

sut.itemManager?.add(ToDoItem(title: "Foo"))

tableView.reloadData()

let cell = tableView.cellForRow(at: IndexPath(row: 0,

section: 0))

XCTAssertTrue(cell is ItemCell)

}

**func test\_CellForRow\_DequeuesCellFromTableView()**

{

~~let mockTableView = MockTableView.mockTableView(withDataSource: sut)  
mockTableView.dataSource = sut~~

~~mockTableView.register(ItemCell.self,~~

~~forCellReuseIdentifier: "ItemCell")~~

sut.itemManager?.add(ToDoItem(title: "Foo"))

mockTableView.reloadData()

\_ = mockTableView.cellForRow(at: IndexPath(row: 0, section: 0))

XCTAssertTrue(mockTableView.cellGotDequeued)

}

**func test\_CellForRow\_CallsConfigCell()**

{

~~let mockTableView = MockTableView.mockTableView(withDataSource: sut)~~

~~mockTableView.dataSource = sut~~

~~mockTableView.register(~~

~~MockItemCell.self,~~

~~forCellReuseIdentifier: "ItemCell")~~

let item = ToDoItem(title: "Foo")

sut.itemManager?.add(item)

mockTableView.reloadData()

let cell = mockTableView

.cellForRow(at: IndexPath(row: 0, section: 0)) as! MockItemCell

~~XCTAssertTrue(cell.configCellGotCalled)~~

XCTAssertEqual(cell.catchedItem, item)

}  
  
**func test\_CellForRow\_Section2\_CallsConfigCellWithDoneItem()**

{

~~let mockTableView = MockTableView()~~  
~~let mockTableView = MockTableView(~~

~~frame: CGRect(x: 0, y:0, width: 320, height: 480),~~

~~style: .plain)  
let mockTableView = MockTableView.mockTableView(withDataSource: sut)  
mockTableView.dataSource = sut~~

~~mockTableView.register(MockItemCell.self,~~

~~forCellReuseIdentifier: "ItemCell")~~

sut.itemManager?.add(ToDoItem(title: "Foo"))

let second = ToDoItem(title: "Bar")

sut.itemManager?.add(second)

sut.itemManager?.checkItem(at: 1)

mockTableView.reloadData()

let cell = mockTableView

.cellForRow(at: IndexPath(row: 0, section: 1)) as! MockItemCell

XCTAssertEqual(cell.catchedItem, second)

}  
  
**func test\_DeleteButton\_InFirstSection\_ShowsTitleCheck()**

{

let deleteButtonTitle = tableView.delegate?.tableView?(

tableView,

titleForDeleteConfirmationButtonForRowAt: IndexPath(row: 0,

section: 0))

XCTAssertEqual(deleteButtonTitle, "Check")

}

**~~func test\_DeleteButton\_InSecondSection\_ShowsTitleUncheck()~~** ~~{~~

~~let deleteButtonTitle = tableView.delegate?.tableView?(~~

~~tableView,~~

~~titleForDeleteConfirmationButtonForRowAt: IndexPath(row: 0,~~

~~section: 1))~~

~~XCTAssertEqual(deleteButtonTitle, "Uncheck")~~

~~}~~

**func test\_CheckingAnItem\_ChecksItInTheItemManager()** {

sut.itemManager?.add(ToDoItem(title: "Foo"))

tableView.dataSource?.tableView?(tableView,

commit: .delete,

forRowAt: IndexPath(row: 0,

section: 0))

XCTAssertEqual(sut.itemManager?.toDoCount, 0)

XCTAssertEqual(sut.itemManager?.doneCount, 1)

XCTAssertEqual(tableView.numberOfRows(inSection: 0), 0)

XCTAssertEqual(tableView.numberOfRows(inSection: 1), 1)

}

extension ItemListDataProviderTests {

class MockTableView: UITableView {

var cellGotDequeued = false

override func dequeueReusableCell(

withIdentifier identifier: String,

for indexPath: IndexPath) -> UITableViewCell {

cellGotDequeued = true

return super.dequeueReusableCell(withIdentifier: identifier, for: indexPath)

}

class func mockTableView(

withDataSource dataSource: UITableViewDataSource)

-> MockTableView {

let mockTableView = MockTableView(

frame: CGRect(x: 0, y: 0, width: 320, height: 480),

style: .plain)

mockTableView.dataSource = dataSource

mockTableView.register(MockItemCell.self,

forCellReuseIdentifier: "ItemCell")

return mockTableView

}

} //added because there was duplicate code, the register function provides the cell for each one of these rows

class MockItemCell : ItemCell {

~~var configCellGotCalled = false~~

~~override func configCell(with item: ToDoItem) {~~

~~configCellGotCalled = true~~

var catchedItem: ToDoItem?

override func configCell(with item: ToDoItem) {

catchedItem = item

~~}~~

}

-----------ABOVE IS ITEMLISTDATAPROVIDER TESTS-----------