实验编号： 8 **四川师大《IOS》实验报告 2018** 年 **10** 月 **31**日

### **计算机科学学院** 2016 级 4 班 实验名称： TextField和TableView \_

姓名：\_樊琳\_\_ 学号：2016110408\_ 指导老师：\_\_李贵洋\_\_ 实验成绩:\_\_\_\_\_

**实验\_一\_ \_\_\_\_\_\_** Swift类和子类**\_\_\_\_\_\_\_\_**

1. 实验目的及要求
2. 掌握TextField和Delegate、Notification的使用；
3. 掌握表视图TableView以及Datasource、Delegate等的使用；
4. 实验内容
5. 将之前作业实现的Person 类和子类导入新项目，新建多个 Student 和 Teacher 对象并放入一数组中；
6. 数组排序后采用 TableView 的形式显示所有Student和Teacher对象的信息，要求每一个对象一个Cell，同时学生和教师显示不同的信息，选择一个Cell 后给出选择反馈；
7. 修改界面，增加添加新学生的部分(键盘要正确显示和消失)，要求实现TableView的增删改;

说明：键盘显示消失需要基于Notification 实现自适应键盘的界面布局;注册KeyboardDidShow和KeyBoardWillHide通知，并处理相应的action，进行界面的布局调整；

1. 实验主要流程、基本操作或核心代码、算法片段（该部分如不够填写，请另加附页）
2. 将之前作业实现的Person 类和子类导入新项目，新建多个 Student 和 Teacher 对象并放入一数组中；
3. 数组排序后采用 TableView 的形式显示所有Student和Teacher对象的信息，要求每一个对象一个Cell，同时学生和教师显示不同的信息，选择一个Cell 后给出选择反馈；
4. 修改界面，增加添加新学生的部分(键盘要正确显示和消失)，要求实现TableView的增删改;

说明：键盘显示消失需要基于Notification 实现自适应键盘的界面布局;注册KeyboardDidShow和KeyBoardWillHide通知，并处理相应的action，进行界面的布局调整；

* 程序代码：

//

// AppDelegate.swift

// mytableview

//

// Created by student on 2018/12/15.

// Copyright © 2018年 fl. All rights reserved.

//

import UIKit

@UIApplicationMain

class AppDelegate: UIResponder, UIApplicationDelegate {

var window: UIWindow?

func application(\_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {

// Overrde point for customization after application launch.

self.window?.rootViewController = UINavigationController(rootViewController: ViewController())

return true

}

func applicationWillResignActive(\_ application: UIApplication) {

// Sent when the application is about to move from active to inactive state. This can occur for certain types of temporary interruptions (such as an incoming phone call or SMS message) or when the user quits the application and it begins the transition to the background state.

// Use this method to pause ongoing tasks, disable timers, and invalidate graphics rendering callbacks. Games should use this method to pause the game.

}

func applicationDidEnterBackground(\_ application: UIApplication) {

// Use this method to release shared resources, save user data, invalidate timers, and store enough application state information to restore your application to its current state in case it is terminated later.

// If your application supports background execution, this method is called instead of applicationWillTerminate: when the user quits.

}

func applicationWillEnterForeground(\_ application: UIApplication) {

// Called as part of the transition from the background to the active state; here you can undo many of the changes made on entering the background.

}

func applicationDidBecomeActive(\_ application: UIApplication) {

// Restart any tasks that were paused (or not yet started) while the application was inactive. If the application was previously in the background, optionally refresh the user interface.

}

func applicationWillTerminate(\_ application: UIApplication) {

// Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:.

}

}

import UIKit

class ViewController: UIViewController, UITableViewDelegate, UITableViewDataSource {

//学生数组

var students = [Student]()

//教师数组

var teachers = [Teacher]()

//定义表头数组

var tableTitle = ["Teacher", "Student"]

//定义一个表视图

var table: UITableView!

//右边按钮

var rightItem: UIBarButtonItem!

//弹出框

var alert: UIAlertController!

override func viewDidLoad() {

super.viewDidLoad()

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

self.title = "table"

self.view.backgroundColor = UIColor.white

//生成3个Teacher对象

for i in 1...3 {

let temp = Teacher(title: "老师", firstName: "王", lastName: "\(i)", age: 40, gender: .female, department: .one)

teachers.append(temp)

}

//生成4个Student对象

for i in 1..<5 {

let temp = Student(stuNo: 2015110100 + i, firstName: "孙", lastName: "\(i)", age: 18, gender: .male, department: .two)

students.append(temp)

}

//按全名排序

teachers.sort { return $0.fullName < $1.fullName }

students.sort { return $0.fullName < $1.fullName }

//创建表视图，并设置代理和数据源

table = UITableView(frame: self.view.bounds)

table.delegate = self

table.dataSource = self

self.view.addSubview(table)

//导航栏控制器右边的按钮

rightItem = UIBarButtonItem(title: "编辑", style: .plain, target: self, action: #selector(edit))

self.navigationItem.rightBarButtonItem = rightItem

//导航栏控制器左边的按钮

let leftItem = UIBarButtonItem(title: "添加", style: .plain, target: self, action: #selector(addStudent))

self.navigationItem.leftBarButtonItem = leftItem

}

/// 添加学生提示框

@objc func addStudent() {

alert = UIAlertController(title: "hh", message: "ss", preferredStyle: .alert)

alert.addTextField { (textField) in

textField.placeholder = "学生学号"

}

alert.addTextField { (textField) in

textField.placeholder = "学生姓"

}

alert.addTextField { (textField) in

textField.placeholder = "学生名"

}

alert.addTextField { (textField) in

textField.placeholder = "学生性别"

}

alert.addTextField { (textField) in

textField.placeholder = "学生年龄"

}

let OKBtn = UIAlertAction(title: "确定", style: .default) { (alert) in

self.add()

}

let cancelBtn = UIAlertAction(title: "取消", style: .cancel, handler: nil)

alert.addAction(OKBtn)

alert.addAction(cancelBtn)

self.present(alert, animated: true, completion: nil)

}

/// 添加学生

func add() {

let no = Int(alert.textFields![0].text!)

let firstName = alert.textFields![1].text!

let lastName = alert.textFields![2].text!

let gender: Gender

switch alert.textFields![3].text! {

case "男":

gender = .male

case "女":

gender = .female

default:

gender = .unknow

}

let age = Int(alert.textFields![4].text!)

let student = Student(stuNo: no!, firstName: firstName, lastName: lastName, age: age!, gender: gender)

students.append(student)

table.reloadData()

}

/// 编辑表视图

@objc func edit() {

if table.isEditing {

rightItem.title = "编辑"

table.isEditing = false

} else {

rightItem.title = "完成"

table.isEditing = true

}

}

// MARK: delegate

func tableView(\_ tableView: UITableView, editingStyleForRowAt indexPath: IndexPath) -> UITableViewCell.EditingStyle {

return .delete

}

func tableView(\_ tableView: UITableView, titleForDeleteConfirmationButtonForRowAt indexPath: IndexPath) -> String? {

return "删除"

}

func tableView(\_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {

let category = tableTitle[indexPath.section]

let name: String

if indexPath.section == 0 {

name = teachers[indexPath.row].fullName

} else {

name = students[indexPath.row].fullName

}

let message = "you selected \(category), name: \(name)"

let alert = UIAlertController(title: "系统提示", message: message, preferredStyle: .alert)

let OKBtn = UIAlertAction(title: "确定", style: .default, handler: nil)

alert.addAction(OKBtn)

self.present(alert, animated: true, completion: nil)

}

// MARK: data source

private func tableView(\_ tableView: UITableView, commit editingStyle: UITableViewCell.EditingStyle, forRowAt indexPath: IndexPath) {

if editingStyle == UITableViewCell.EditingStyle.delete {

if indexPath.section == 0 {

teachers.remove(at: indexPath.row)

} else {

students.remove(at: indexPath.row)

}

tableView.deleteRows(at: [indexPath], with: .left)

}

}

func tableView(\_ tableView: UITableView, moveRowAt sourceIndexPath: IndexPath, to destinationIndexPath: IndexPath) {

if sourceIndexPath.section != destinationIndexPath.section {

tableView.reloadData()

} else {

if sourceIndexPath.section == 0 {

teachers.insert(teachers.remove(at: sourceIndexPath.row), at: destinationIndexPath.row)

} else {

students.insert(students.remove(at: sourceIndexPath.row), at: destinationIndexPath.row)

}

}

}

func numberOfSections(in tableView: UITableView) -> Int {

return tableTitle.count

}

func tableView(\_ tableView: UITableView, titleForHeaderInSection section: Int) -> String? {

return tableTitle[section]

}

func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {

if section == 0 {

return teachers.count

} else {

return students.count

}

}

func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

let identifier = tableTitle[indexPath.section]

var cell = tableView.dequeueReusableCell(withIdentifier: identifier)

if cell == nil {

let style: UITableViewCell.CellStyle = (identifier == "Teacher") ? .subtitle : .default

cell = UITableViewCell(style: style, reuseIdentifier: identifier)

cell?.accessoryType = .disclosureIndicator

}

switch identifier {

case "Teacher":

cell?.textLabel?.text = teachers[indexPath.row].fullName

cell?.detailTextLabel?.text = teachers[indexPath.row].title

case "Student":

cell?.textLabel?.text = students[indexPath.row].fullName

default:

break

}

return cell!

}

override func didReceiveMemoryWarning() {

super.didReceiveMemoryWarning()

// Dispose of any resources that can be recreated.

}

}

//

// Person.swift

// mytableview

//

// Created by student on 2018/12/15.

// Copyright © 2018年 fl. All rights reserved.

//

import Foundation

//性别的枚举

enum Gender: Int {

case male //男性

case female //女性

case unknow //未知

//重载>操作符，方便后面排序使用

static func >(lhs: Gender, rhs: Gender) -> Bool {

return lhs.rawValue < rhs.rawValue

}

}

//公寓的枚举

enum Department {

case one, two, three

}

//学校协议

protocol SchoolProtocol {

var department: Department { get set }

func lendBook()

}

//人类

class Person: CustomStringConvertible {

var firstName: String //姓

var lastName: String //名

var age: Int //年龄

var gender: Gender //性别

var fullName: String { //全名

get {

return firstName + lastName

}

}

//构造方法

init(firstName: String, lastName: String, age: Int, gender: Gender) {

self.firstName = firstName

self.lastName = lastName

self.age = age

self.gender = gender

}

convenience init(firstName: String, age: Int, gender: Gender) {

self.init(firstName: firstName, lastName: "", age: age, gender: gender)

}

convenience init(firstName: String) {

self.init(firstName: firstName, age: 0, gender: Gender.unknow)

}

required convenience init() {

self.init(firstName: "")

}

//重载==

static func ==(lhs: Person, rhs: Person) -> Bool {

return lhs.fullName == rhs.fullName && lhs.age == rhs.age && lhs.gender == rhs.gender

}

//重载!=

static func !=(lhs: Person, rhs: Person) -> Bool {

return !(lhs == rhs)

}

//实现CustomStringConvertible协议中的计算属性，可以使用print直接输出对象内容

var description: String {

return "fullName: \(self.fullName), age: \(self.age), gender: \(self.gender)"

}

//输出Person XXX is running

func run() {

print("Person \(self.fullName) is running")

}

}

//教师类

class Teacher: Person, SchoolProtocol {

var title: String //标题

var department: Department //公寓

//构造方法

init(title: String, firstName: String, lastName: String, age: Int, gender: Gender, department: Department) {

self.title = title

self.department = department

super.init(firstName: firstName, lastName: lastName, age: age, gender: gender)

}

init(title: String, department: Department) {

self.title = title

self.department = department

super.init(firstName: "", lastName: "", age: 0, gender: .unknow)

}

convenience required init() {

self.init(title: "", department: Department.one)

}

//重写父类的计算属性

override var description: String {

return "title: \(self.title), fullName: \(self.fullName), age: \(self.age), gender: \(self.gender), department: \(self.department)"

}

//重载父类run方法

override func run() {

print("Teacher \(self.fullName) is running")

}

//遵循协议的方法

func lendBook() {

print("Teacher \(self.fullName) lend a book")

}

}

//学生类

class Student: Person, SchoolProtocol {

var stuNo: Int //学号

var department: Department //公寓

//构造方法

init(stuNo: Int, firstName: String, lastName: String, age: Int, gender: Gender, department: Department) {

self.stuNo = stuNo

self.department = department

super.init(firstName: firstName, lastName: lastName, age: age, gender: gender)

}

convenience init(stuNo: Int, firstName: String, lastName: String, age: Int, gender: Gender) {

self.init(stuNo: stuNo, firstName: firstName, lastName: lastName, age: age, gender: gender, department: .one)

}

init(stuNo: Int, department: Department) {

self.stuNo = stuNo

self.department = department

super.init(firstName: "", lastName: "", age: 0, gender: Gender.unknow)

}

required convenience init() {

self.init(stuNo: 0, department: .one)

}

//重写父类的计算属性

override var description: String {

return "stuNo: \(self.stuNo), fullName: \(self.fullName), age: \(self.age), gender: \(self.gender), department: \(self.department)"

}

//重载父类run方法

override func run() {

print("Student \(self.fullName) is running")

}

//遵循协议的方法

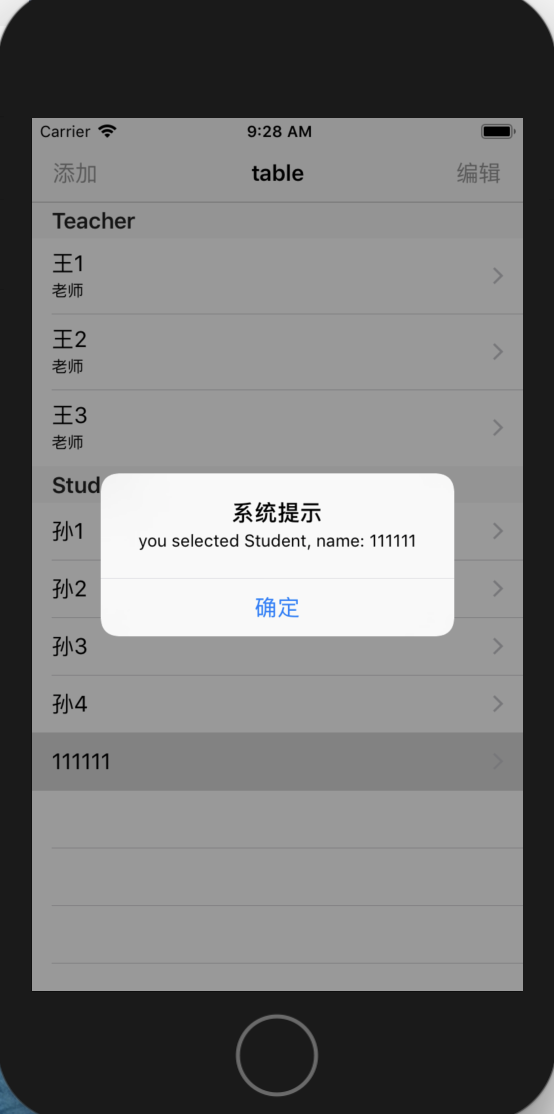
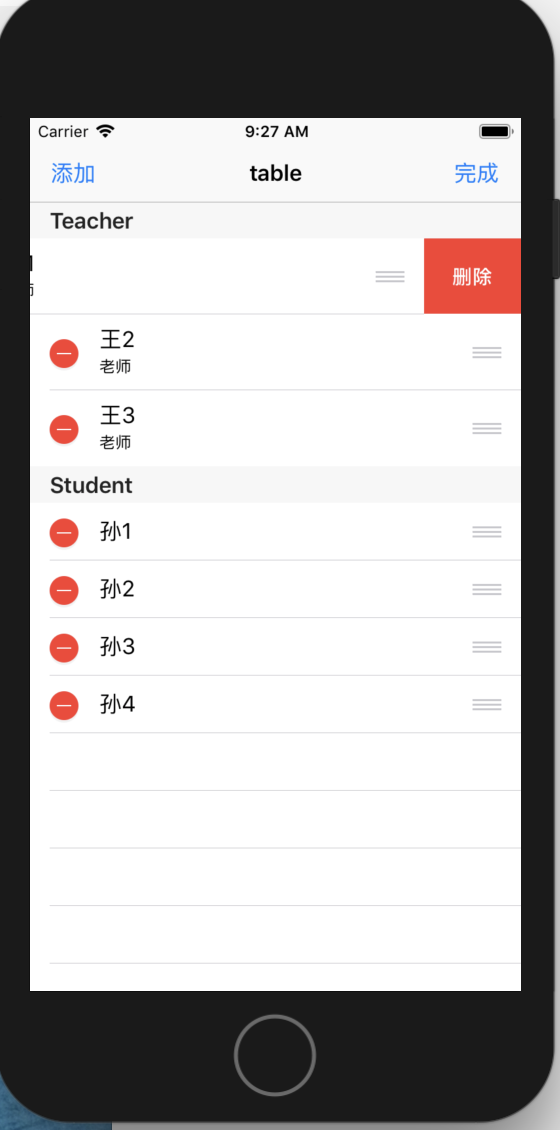
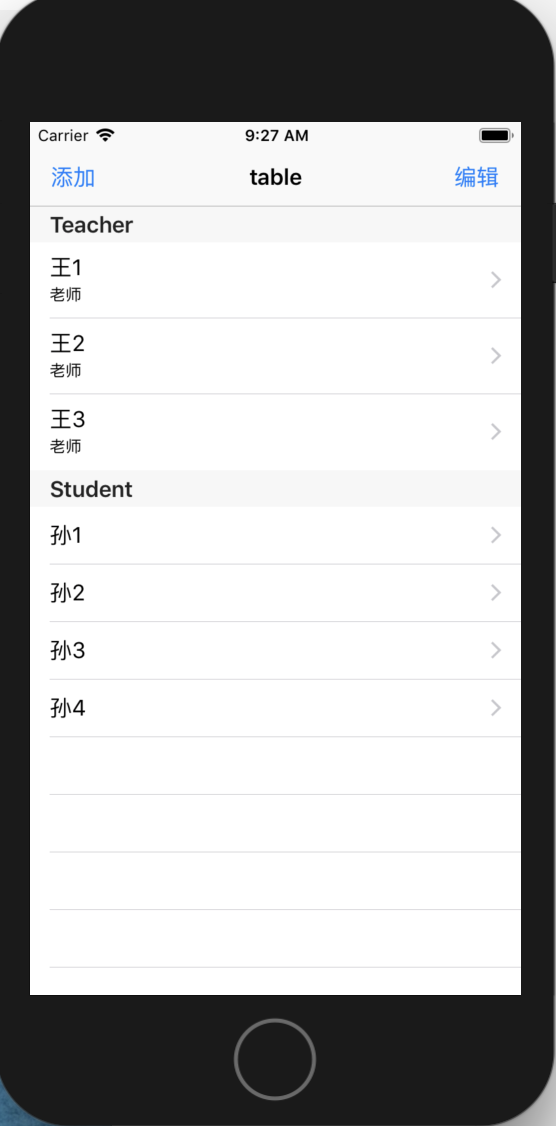
func lendBook() {

print("Teacher \(self.fullName) lend a book")

}

}

* 运行结果：



1. 实验结果的分析与评价（该部分如不够填写，请另加附页）

这一章节的内容在开发中所运用的知识比较多。Tableview是一个很有实用性的界面。我把录屏视频看了很多遍，才慢慢开始写实验要求的内容。在区分老师和学生然后显示在tableview上时出了点问题，我开始以为要两个tableview，最后发现只需要在显示在一个tableview上是判断一下是属于老师还是学生的信息就可以了。

Github地址：

注：实验成绩等级分为（90－100分）优，（80－89分）良，(70-79分)中，（60－69分）及格，（59分）不及格。