实验编号： 2 **四川师大《IOS》实验报告 2018** 年 **9** 月 **12** 日

### **计算机科学学院** 2016 级 4 班 实验名称： 闭包、扩展、泛型、协议 \_

姓名：\_\_樊琳\_\_ 学号：\_2016110408\_ 指导老师：\_\_李贵洋\_\_ 实验成绩:\_\_\_\_\_

**实验\_二\_ \_\_\_\_\_\_**闭包、扩展、泛型、协议**\_\_\_\_\_\_\_\_**

1. 实验目的
2. 掌握闭包、扩展、泛型的定义;
3. 掌握排序等采用闭包方法的实现；
4. 掌握类的派生、协议；
5. 掌握版本控制git的进阶使用；
6. 实验内容

* 作业1（闭包、扩展、泛型）：

1. 给定一个Dictionary，Dictionary包含key值name和age，用map函数返回age字符串数组;
2. 给定一个String数组，用filter函数选出能被转成Int的字符串
3. 用reduce函数把String数组中元素连接成一个字符串，以逗号分隔
4. 用 reduce 方法一次求出整数数组的最大值、最小值、总数和
5. 新建一个函数数组，函数数组里面保存了不同函数类型的函数，要求从数组里找出参数为一个整数，返回值为一个整数的所有函数；
6. 扩展Int，增加sqrt方法，可以计算Int的Sqrt值并返回浮点数，进行验证；
7. 实现一个支持泛型的函数，该函数接受任意个变量并返回最大和最小值，分别传入整数值、浮点数值、字符串进行验证。
8. 掌握版本控制git的进阶使用
   1. git reset 恢复到之前修改的版本；
   2. git log 看提交记录；
   3. git branch 新建分支；
   4. git checkout 切换分支；
   5. git branch -d 删除分支；

* 作业2:（枚举、类、派生、协议）(红色字体为新增内容)

1. 实现Person类：
   1. 要求具有firstName, lastName，age，gender等存储属性,fullName计算属性；其中gender是枚举类型（male，female）；
   2. 具有指定构造函数和便利构造函数；
   3. 两个Person实例对象可以用==和!=进行比较；
   4. Person实例可以直接用print输出；
   5. Person增加run方法(方法里面直接print输出Person XXX is running;
2. 从Person分别派生Teacher类和Student类：
   1. Teacher类增加属性title，实例可以直接用print输出；
   2. Student类增加属性stuNo，实例可以直接用print输出；
   3. Teacher和Student重载run方法(方法里面直接print输出Teacher XXX is running和Student XXX is running)
3. 分别构造多个Person、Teacher和Student对象，并将这些对象存入同一个数组中；
4. 新建一个协议SchoolProtocol，协议包括一个department属性(Enum，自己实现enum的定义)和lendBook方法（随便写点内容，能区隔即可）；
5. 修改Teacher和Student，让这两个类实现该协议；
6. 对数组执行以下要求：
   1. 分别统计Person、Teacher和Student对象的个数并放入一字典中，统计完后输出字典内容；
   2. 对数组按以下要求排序并输出：age、fullName、gender+age；
   3. 对数组进行穷举，调用每个对象的run方法，同时调用满足协议SchoolProtocol对象的lendBook方法；
7. 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）

* 作业1（闭包、扩展、泛型）：

1. 给定一个Dictionary，Dictionary包含key值name和age，用map函数返回age字符串数组;
2. 给定一个String数组，用filter函数选出能被转成Int的字符串
3. 用reduce函数把String数组中元素连接成一个字符串，以逗号分隔
4. 用 reduce 方法一次求出整数数组的最大值、最小值、总数和
5. 新建一个函数数组，函数数组里面保存了不同函数类型的函数，要求从数组里找出参数为一个整数，返回值为一个整数的所有函数；
6. 扩展Int，增加sqrt方法，可以计算Int的Sqrt值并返回浮点数，进行验证；
7. 实现一个支持泛型的函数，该函数接受任意个变量并返回最大和最小值，分别传入整数值、浮点数值、字符串进行验证。
8. 掌握版本控制git的进阶使用
   1. git reset 恢复到之前修改的版本；
   2. git log 看提交记录；
   3. git branch 新建分支；
   4. git checkout 切换分支；
   5. git branch -d 删除分支；

* 程序代码：

/// question: 1

let dic = [["name": "zhangsan", "age": 18], ["name": "lisi", "age": 19], ["name": "wangwu", "age": 20]]

let str = dic.map( { $0["name"]! } ) //返回字典数组中每个字典元素的"name"对应的值

print(str) //输出["zhangsan", "lisi", "wangwu"]

/// question: 2

let arr1 = ["ajsfhu", "182has8f", "187326478", "123"]

let arr2 = arr1.filter( { Int($0) != nil } ) //先将字符串元素强转为Int?，如果值为nil，则表示该字符串不能转换成Int

print(arr2) //输出["187326478", "123"]

/// question: 3

let arr3 = ["ajsfhu", "182has8f", "187326478"]

var str1 = arr3.reduce("", { $0 + "," + $1 }) //将字符串数组整合

str1.remove(at: str1.startIndex) //因为整合结果的字符串的第一个字符为","，所以需要将这个","移除

print(str1) //输出ajsfhu,182has8f,187326478

/// question: 4

let tuple = intArr.reduce((max: intArr[0], min: intArr[0], sum: 0), { (max: max($0.max, $1), min: min($0.min, $1), $0.sum + $1) })

print(tuple) //输出(44, 11, 99)

/// question: 5

func f1(a: Int) -> Int {

return a

} //函数类型为(Int) -> Int

func f2(a: String) -> Int {

return Int(a)!

} //函数类型为(String) -> Int

func f3() -> Int {

return 2

} //函数类型为() -> Int

func f4(a: Int) {

} //函数类型为(Int) -> Void

func f5(a: Int) -> Int {

return a + 1

} //函数类型为(Int) -> Int

let funArr: [Any] = [f1, f2, f3, f4, f5]

for (index, value) in funArr.enumerated() { //因为循环过程中需要设计到数组的下标，要将数组元素一一列举出来，所以需要调用数组的enumerated()方法。

if value is (Int) -> Int {

print(index) //输出类型为(Int) -> Int的函数在数组中的下标，

}

} //输出结果为0 (换行) 4

/// question: 6

extension Int {

//因为直接使用系统的sqrt(Double)函数会与扩展中定义的函数冲突，所以需要指定系统函数的框架

/// 求平方根

///

/// - Returns: 返回该Int型数据的平方根

func sqrt() -> Double {

return Darwin.sqrt(Double(self))

}

}

print(4.sqrt()) //输出2.0

/// question: 7

//自定义泛型函数，因为需要比较大小，所以该泛型需要遵循Comparable协议，函数参数为可变参数

/// 返回参数中的最大值和最小值

///

/// - Parameter a: 一组可以直接比较大小的值

/// - Returns: 返回一个元组，第一个值为最大值，第二个值为最小值

func getMaxAndMin<T: Comparable>(a: T...) -> (T, T) {

var max = a[0]

var min = a[0]

for item in a {

if item > max {

max = item

} else if item < min {

min = item

}

}

return (max, min)

}

print(getMaxAndMin(a: 1, 2, 3, 9, 2, 88)) //输出(88, 1)

print(getMaxAndMin(a: 1.0, 2.0, 3.0, 9.0, 2.0, 88.0)) //输出(88.0, 1.0)

print(getMaxAndMin(a: "a", "b", "A", "sss")) //输出("sss", "A")

* 运行结果：

已经在代码中注释出来（在线swift写的）

* 作业2:（枚举、类、派生、协议）(红色字体为新增内容)

1. 实现Person类：
   1. 要求具有firstName, lastName，age，gender等存储属性,fullName计算属性；其中gender是枚举类型（male，female）；
   2. 具有指定构造函数和便利构造函数；
   3. 两个Person实例对象可以用==和!=进行比较；
   4. Person实例可以直接用print输出；
   5. Person增加run方法(方法里面直接print输出Person XXX is running;
2. 从Person分别派生Teacher类和Student类：
   1. Teacher类增加属性title，实例可以直接用print输出；
   2. Student类增加属性stuNo，实例可以直接用print输出；
   3. Teacher和Student重载run方法(方法里面直接print输出Teacher XXX is running和Student XXX is running)
3. 分别构造多个Person、Teacher和Student对象，并将这些对象存入同一个数组中；
4. 新建一个协议SchoolProtocol，协议包括一个department属性(Enum，自己实现enum的定义)和lendBook方法（随便写点内容，能区隔即可）；
5. 修改Teacher和Student，让这两个类实现该协议；
6. 对数组执行以下要求：
   1. 分别统计Person、Teacher和Student对象的个数并放入一字典中，统计完后输出字典内容；
   2. 对数组按以下要求排序并输出：age、fullName、gender+age；
   3. 对数组进行穷举，调用每个对象的run方法，同时调用满足协议SchoolProtocol对象的lendBook方法；

* 程序代码：

//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")

}

}

var p1 = Person(firstName: "张")

var p2 = Person(firstName: "张", age: 20, gender: .male)

print(p1) //输出fullName: 张, age: 0, gender: male

print(p1 == p2) //输出false

print(p1 != p2) //输出true

//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")

}

}

var t1 = Teacher(title: "hello", department: .one)

print(t1) //输出title: hello, fullName: , age: 0, gender: unknow, department: one

//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)

}

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")

}

}

var s1 = Student(stuNo: 2015110101, department: .two)

print(s1) //输出stuNo: 2015110101, fullName: , age: 0, gender: unknow, department: two

//初始化一个空的Person数组

var array = [Person]()

//生成5个Person对象

for i in 1...5 {

let temp = Person(firstName: "张", lastName: "\(i)", age: 20, gender: .male)

array.append(temp)

}

//生成3个Teacher对象

for i in 1...3 {

let temp = Teacher(title: "hello", firstName: "李", lastName: "\(i)", age: 21, gender: .female, department: .one)

array.append(temp)

}

//生成4个Student对象

for i in 1..<5 {

let temp = Student(stuNo: 2015110100 + i, firstName: "王", lastName: "\(i)", age: 19, gender: .male, department: .two)

array.append(temp)

}

//定义一个字典，用于统计每个类的对象个数

var dict = ["Person": 0, "Teacher": 0, "Student": 0]

for item in array {

if item is Teacher { //是否是Teacher类

dict["Teacher"]! += 1

} else if item is Student { //是否是Student

dict["Student"]! += 1

} else { //Person类

dict["Person"]! += 1

}

}

//输出字典值

for (key, value) in dict {

print("\(key) has \(value) items")

}

//原始数组

print("------------------------------")

for item in array {

print(item)

}

//根据age从大到小排序

print("------------------------------")

array.sort { return $0.age > $1.age}

for item in array {

print(item)

}

//根据全名从前往后排序

print("------------------------------")

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

for item in array {

print(item)

}

//根据gender和age从大往小排序

print("------------------------------")

array.sort { return ($0.gender > $1.gender) && ($0.age > $1.age) }

for item in array {

print(item)

}

//穷举，调用run方法和lendBook方法

print("------------------------------")

for item in array {

item.run()

if let teacher = item as? Teacher {

teacher.lendBook()

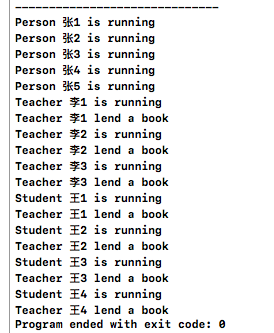
} else if let student = item as? Student {

student.lendBook()

}

}

* 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）

这次实验对我们的要求是熟练掌握闭包,扩展,泛型, 枚举,类,派生,协议的内容.在这些内容里面拓展和协议对于我们来说是新的内容,在以前的学习中我们也没有学习过类似的内容.需要更加努力学习这方面的知识.闭包在上一次的实验中也有涉及到,但是这次是关于那三个函数的运用.

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。