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： 郭周倩 学号： 2016110413 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. 实验要求
7. 认真填写实验报告，要求附加部分运行界面和主要代码；
8. 对设计好的程序，检查输出是否符合预期，如有错请分析错误原因并解决；
9. 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：

import Darwin

import UIKit

//1.

print("第1题")

var dicArr :[[String:Any]] = [["name":"Alice","age":18],["name":"David","age":19]]

//方法一：

var nameArr = dicArr.map({ $0["age"] }) //map不能自动解包

print(nameArr)

var nameArr1 = dicArr.map({ $0["age"]! }) //感叹号解包

print(nameArr1)

//方法二：

//var nameArr2 = dicArr.flatMap { $0["name"] }

//print(nameArr2)

//2.

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

print("第2题")

var myStr = ["123","12w","ttt"]

//方法一：

var res\_int1 = myStr.filter{ Int($0) != nil }

print(res\_int1)

//方法二：

//var res\_int2 = myStr.flatMap { Int($0) }

//print(res\_int2)

//3.

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

print("第3题")

var strArr = ["li","gui","yang"]

var res\_str = strArr.reduce("",{ $0+$1+"," })

print(res\_str)

//4、

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

print("第4题")

var str1 = [10,4,2,1,24,45]

let temp = str1.reduce((max:Int.min,min:Int.max,sum:0))

{ (temp,i) in

return (max(temp.max,i),min(temp.min,i),temp.sum+i)

}

print("max:\(temp.max)")

print("min:\(temp.min)")

print("sum:\(temp.sum)")

//5、筛选函数数组

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

print("第5题")

func one()->Void

{

print("返回值为空")

}

func two()->Int

{

return 2

}

func three(max:Int)->Int

{

return 3

}

func four(max:Int)->Void

{

print("返回值为空，参为整数")

}

func five(max:Int,min:Int)->Void

{

print("返回值为空，参为整数")

}

var arr6:[Any]=[one,two,three,four,five]

for (key,value) in arr6.enumerated()

{

if value is (Int)->Int

{

print("下标为\(key)的函数是参数为整数，返回值也是整数")

}

}

//6.

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

print("第6题")

extension Int{

func Sqrt() -> Double{

return Darwin.sqrt(Double(self))

}

}

var Num:Int = 9

print(Num.Sqrt())

//7.

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

print("第7题")

func returnValue<T :Comparable>(arr :[T]) -> (min:T,max:T){

var min = arr[0]

var max = arr[0]

for i in arr{

if i > max{

max = i

}else if i < min{

min = i

}

}

return (min,max)

}

print("输入整数：")

var arr1:[Int] = [123,12,-1,-67,222,0]

print(returnValue(arr: arr1))

print("输入字符串：")

var arr2 = ["123","12","-1","-67","222","0"]

print(returnValue(arr: arr2))

print("输入浮点数：")

var arr3 = [123.0,12.0,-1.0,-67.0,222.0,0.0]

print(returnValue(arr: arr3))

//8

git reset

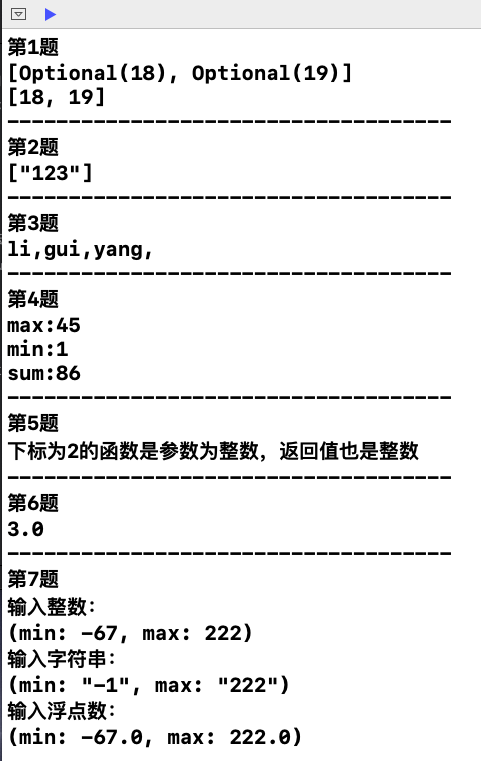
git log

git branch

git checkout

git branch -d

* 运行结果：



* 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;

* 程序代码：

//性别的枚举类型

enum Gender:String{

case female

case male

}

class Person{

//存储属性

var firstName : String

var lastName : String

var age : Int

var gender:Gender

//计算属性

var fullName : String{

get{

return firstName+""+lastName

}

}

//指定构造函数

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

self.firstName = firstName

self.lastName=lastName

self.age = age

self.gender = gender

}

//便利构造函数

convenience init(name:String){

self.init(firstName:name,lastName:"",age:18,gender:Gender.female)

}

var description:String{

return"Name:\(fullName) Age:\(age) Gender:\(gender)"

}

}

let p1 = Person(firstName: "person", lastName: "One", age: 20, gender: Gender.female)// 使用指定构造函数构造的对象

let p2 = Person(name: "personTwo")//使用便利构造函数构造的对象

print(p1.description)

print(p2.description)

//重新方法如下：

func == (left:Person,right:Person)->Bool{

return left.firstName==right.firstName && left.lastName==right.lastName && left.age == right.age && left.gender == right.gender

}

func != (left:Person,right:Person)->Bool{

return left.firstName != right.firstName || left.lastName != right.lastName || left.age != right.age || left.gender != right.gender

}

//比较一

if p1 == p2 {

print("两个对象比较相等")

}else{

print("两个对象比较不相等")

}

//比较二

if p1 != p1 {

print("两个对象比较不相等")

}else{

print("两个对象比较相等")

}

//扩展person类

extension Person{

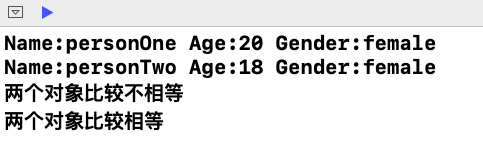
public func run(){

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

}

}

* 运行结果：



1. 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)

* 程序代码：

//部门的枚举类型

enum Department:String{

case jkxy

case wxy

case gxy

}

//协议

protocol SchoolProtocol{

var department:Department{ get }

func lendBook()

}

class Student:Person{

var stuNo:String

//指定构造函数

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

self.stuNo = stuNo

super.init(firstName: firstName, lastName: lastName, age: age, gender: gender)//调用的是指定构造函数

}

//便利构造函数

convenience init(name:String){

self.init(firstName: name,lastName: "",age: 18,gender: Gender.female,stuNo: "001")

}

override var description: String{

return super.description + " stuNo:\(stuNo)"

}

}

class Teacher:Person{

var title:String

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

self.title = title

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

}

convenience init(name:String){

self.init(firstName: name, lastName: "", age: 18, gender: Gender.female, title: "teacher")

}

override var description: String{

return super.description+" title:\(title)"

}

}

extension Student{

//重载--->参数什么的不一样

func run(name:String?){

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

}

}

extension Teacher{

//重载

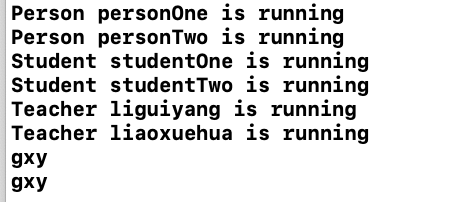
func run(name:String?){

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

}

}

* 运行结果：



1. 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；

* 程序代码：

let s1 = Student(firstName: "student", lastName: "One", age: 20, gender: Gender.female, stuNo: "013")

let s2 = Student(name: "studentTwo")

print(s1.description)

print(s2.description)

let t1 = Teacher(firstName: "li", lastName: "guiyang", age: 40, gender: Gender.male, title: "headTeacher")

let t2 = Teacher(name: "liaoxuehua")

print(t1.description)

print(t2.description)

var persons = [Person]()

persons.append(p1)

persons.append(p2)

persons.append(s1)

persons.append(s2)

persons.append(t1)

persons.append(t2)

var stuNum = 0

var teaNum = 0

var perNum = 0

for i in 0..<persons.count{

if persons[i] is Student{

stuNum += 1

}else if persons[i] is Teacher{

teaNum += 1

}else{

perNum += 1

}

}

//数组排序

persons.sort(by: { $0.age > $1.age}) //age排序

persons.sort(by: { $0.fullName > $1.fullName}) //fullName排序

persons.sort(by: {$0.age > $1.age && $0.gender.rawValue > $1.gender.rawValue })

persons.sort { //gender+age排序

if $0.age == $1.age{

return $0.gender.rawValue > $1.gender.rawValue

}

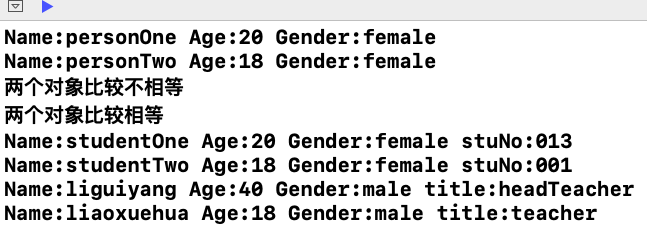
else{

return $0.age > $1.age

}

}

* 运行结果：



1. 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）；
2. 修改Teacher和Student，让这两个类实现该协议；
3. 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：

extension Student:SchoolProtocol{

var department: Department {

return Department.jkxy

}

func lendBook(){

print("Student实现协议SchoolProtocol")

}

}

extension Teacher:SchoolProtocol{

var department: Department {

return Department.gxy

}

func lendBook() {

print("Teacher实现协议SchoolProtocol")

}

}

p1.run()

p2.run()

s1.run(name: "")

s2.run(name: "")

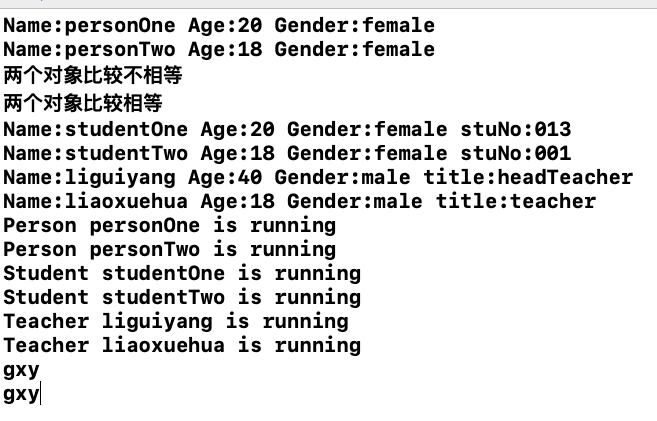
t1.run(name: "")

t2.run(name: "")

print(t1.department)

print(t2.department)

* 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）

文件上传GitHub链接：<https://github.com/gzqxyz/swift-test>

本次实验是在第一次实验的基础上面完成的，作业二只是在实验一的基础上添加了类的扩展和协议，作业一，主要是要学会如何运用map、fitter、reduce方法。运用好这几个方法，对于swift编程有很大的好处，可以节省很多代码。

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。