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：\_谭靖薇\_ 学号：\_2016110437 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. 认真填写实验报告，要求附加部分运行界面和主要代码；
7. 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；
8. 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：

//1

var someDict:[String:Int] = ["tjw":20,"thq":40,"tkj":22]

var after = someDict.map( {"\($1)"} )

print(after)

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

//2

var str2 = ["12","sdfg","34","3"]

var fil2 = str2.filter( {Int($0) != nil} )

print(fil2)

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

//3

var str3 = ["12","sdfg","34","3"]

var redu3 = str3.reduce("",{$0+$1+","})

print(redu3)

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

//4

var int4 = [1,5,2,7,3]

var after4 = int4.reduce((max: int4[0], min: int4[0], sum: 0),

{ (max: max($0.max, $1),

min: min($0.min, $1),

$0.sum + $1) })

print(after4)

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

//5

func f1(a:Int) -> Int {

return a

}

func f2(a:String) -> String {

return a

}

func f3(a:String) -> String {

return a

}

var funArray:[Any] = [f1,f2,f3]

for (index, value) in funArray.enumerated() {

if value is (Int) -> Int {

print(index)

}

}

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

//6

/\*\*extension Int {

func sqrt() -> Double {

return sqrt(Double(self))

}

}

print(4.sqrt())\*/

//7

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")

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

* 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：

enum Gender: Int{

case male

case female

static func >(p1: Gender, p2: Gender) -> Bool {

return p1.rawValue < p2.rawValue

}

}

enum Department: Int{

case cs

case english

}

protocol SchoolProtocol{

var department: Department { get set }

func lendBook() -> String

}

class Person{

var firstName:String

var lastName:String

var age:Int

var gender:Gender

var fullName:String{

get{

return firstName+" "+lastName

}

}

//具有指定构造函数和便利构造函数

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

self.firstName = firstName

self.lastName = lastName

self.age = age

self.gender = gender

}

convenience init(name:String) {

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

}

var description:String {

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

}

//Person增加run方法(方法里面直接print输出Person XXX is running

func run() {

print("Person "+self.firstName+" is running")

}

static func ==(p1:Person,p2:Person) -> Bool {

return p1.description == p2.description

}

static func !=(p1:Person,p2:Person) -> Bool {

return p1.description != p2.description

}

}

* 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：

class Teacher:Person,SchoolProtocol{

var title:String

var department: Department

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

self.title = title

self.department = department

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

}

convenience init(name:String) {

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

}

override var description:String {

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

}

func lendBook() -> String{

return "Teacher "+self.firstName+" lend a book"

}

//2.Teacher重载run

override func run() {

print("Teacher "+self.firstName+" is running")

}

}

class Student: Person,SchoolProtocol{

var stuNo:Int

var department: Department

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

self.stuNo = stuNo

self.department = department

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

}

convenience init(name:String) {

self.init(firstName:name,lastName:"",age:18,gender:Gender.female,stuNo:60,department:Department.cs)

}

override var description:String {

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

}

//2.Student重载run

override func run() {

print("Student "+self.firstName+" is running")

}

func lendBook() -> String{

return "Student "+self.firstName+" lend a book"

}

}

* 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：
* 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：

/\*\*方法一

extension Person: Equatable {

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

return lhs.description == rhs.description

}

}\*/

let p1 = Person(firstName:"li",lastName:"guiyang",age:43,gender:Gender.male)

let p2 = Person(name:"gaoyuexiang")

//Person实例可以直接用print输出

print(p1.description)

print(p2.description)

let s1 = Student(name:"fan")

print(s1.description)

let t1 = Teacher(name:"tan")

print(t1.description)

p1.run()

s1.run()

t1.run()

//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；

var manyPerson = [p1,p2,s1,t1]

for item in manyPerson {

item.run()

if item is Student{

print(Student(item).lendBook())

}else if item is Teacher{

print(Teacher(item).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）

https://github.com/xiongmaobeibei/ios\_homework/

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。