实验编号： 1 **四川师大《IOS》实验报告 2018** 年 **9** 月 **5** 日

计算机科学学院 2017 级 4 班 实验名称： Swift类和子类 \_

姓名： 李晋 学号：2017110517 指导老师：李贵阳 实验成绩:

### **实验2 闭包、扩展、泛型、协议**

**1、实验目的**

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

**2、实验要求**

1. 认真填写实验报告，要求附加部分运行界面和主要代码；
2. 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；

**3、实验内容**

* 作业1（闭包、扩展、泛型）：

1. 给定一个Dictionary数组，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；

**4.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数组，Dictionary包含key值name和age，用map函数返回age字符串数组;

**let** theDic1:[String: String] = ["name": "mary", "age": "15"]

**let** theDic2:[String: String] = ["name": "lucy", "age": "16"]

**let** theDic3:[String: String] = ["name": "liu", "age": "18"]

**let** theArray = [theDic1, theDic2, theDic3]

print(theArray)

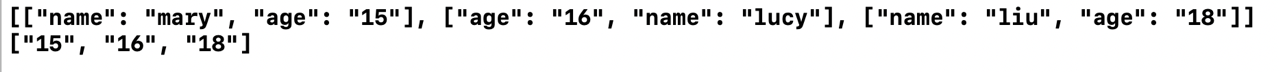
**let** overArray = theArray.map(){myDic -> String **in**

**return** myDic["age"]!

}

print(overArray)

* 运行结果



1. 给定一个String数组，用filter函数选出能被转成Int的字符串

**let** strArray = ["li", "8", "98", "kjidnf"]

print(strArray)

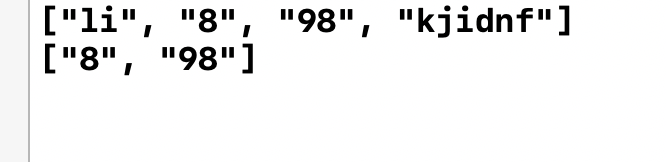
**let** numberArr = strArray.filter{

Int($0) != **nil**

}

print(numberArr)

* 运行结果



1. 用reduce函数把String数组中元素连接成一个字符串，以逗号分隔

**let** strArray1 = ["li", "8", "", "kjidnf",""]

print(strArray1)

**var** theString = strArray1.reduce(""){

**if** $0 == ""{

**return** $1

}**else** **if** $1 == ""{

**return** $0

}**else**{

**return** $0 + "," + $1

}

}

print(theString)

* 运行结果



1. 用 reduce 方法一次求出整数数组的最大值、最小值、总数和

**let** intArray = [12,55,-88,22,168,1]

print(intArray)

//最大值

**let** maxValue = intArray.reduce(intArray[0],{

**return** $0 > $1 ? $0 : $1

})

print("最大值为\(maxValue)")

//最小值

**let** minValue = intArray.reduce(intArray[0],{

**return** $0 < $1 ? $0 : $1

})

print("最小值为\(minValue)")

//总数和

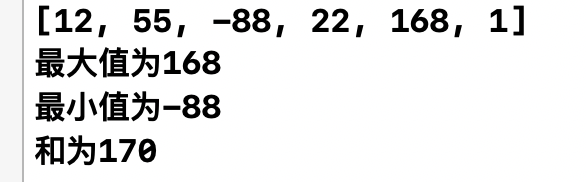
**let** sum = intArray.reduce(0,{

**return** $0 + $1

})

print("和为\(sum)")

* 运行结果



1. 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；

**func** func1(a: Int) -> Int{

**return** 1

}

**func** func2(b: String) -> Int{

**return** 1

}

**func** func3(c: Float) -> Int{

**return** 1

}

**func** func4(d: Int) -> Int{

**return** 2

}

**var** funcArray = [func1,func2,func3,func4] **as** [**Any**]

print(funcArray)

**var** theFunc = funcArray.filter(){

$0 **is** (Int) -> Int

}

print(theFunc **as**! [(Int) -> Int])

1. 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；

**extension** Int{

**func** sqrt() -> Double{

**return** Darwin.sqrt(Double(**self**))

}

}

1. 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。

**protocol** Addable {

**static** **func** < (x: Self, y: Self) -> Bool

**static** **func** > (x: Self, y: Self) -> Bool

}

**func** searchMaxAndMin<T: Addable>(data: [T]) -> (max:T,min:T) {

**var** max = data[0]

**var** min = data[0]

**for** i **in** data {

**if** max < i {

max = i

}**else** **if** min > i{

min = i

}

}

**return** (max,min)

}

**extension** Int: Addable {}

**extension** Double: Addable {}

**extension** Float: Addable {}

**extension** String: Addable {}

**let** tempInt = searchMaxAndMin(data:[8,10,9,11])

print("传入整数时最大值为：\(tempInt.max) 最小值为：\(tempInt.min)")

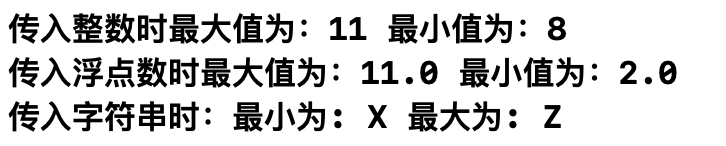
**let** tempFloat = searchMaxAndMin(data:[11.0,2.0,5.0,7.0])

print("传入浮点数时最大值为：\(tempFloat.max) 最小值为：\(tempFloat.min)")

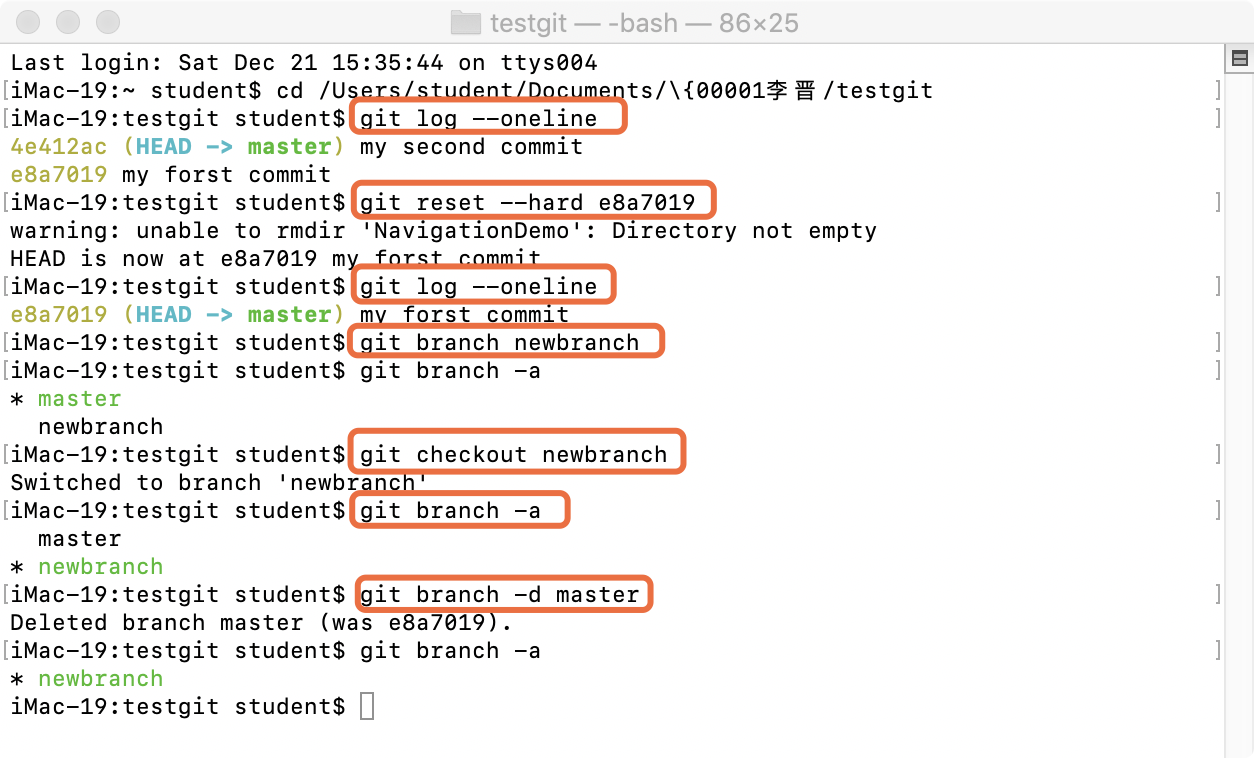
**let** tempString = searchMaxAndMin(data:["X","Y","Z"])

print("传入字符串时：最小为: \(tempString.min) 最大为: \(tempString.max)")

* 运行结果



1. 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；

//协议

**protocol** SchoolProtocol{

**func** lendbook()

**var** department:Department{**get**}

}

**extension** Teacher: SchoolProtocol{

**var** department: Department {

**get** {

Department.teacher

}

}

**func** lendbook() {

print("teacher lendbook")

}

}

**extension** Student: SchoolProtocol{

**var** department: Department {

**get** {

Department.student

}

}

**func** lendbook() {

print("student lendbook")

}

}

1. 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；

**import** UIKit

/\*

作业2:（枚举、类、派生）

(1)实现Person类

a)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）；

b)具有指定构造函数和便利构造函数；

c)两个Person实例对象可以用==和!=进行比较；

d)Person实例可以直接用print输出；

\*/

**enum** Gender: Int{

**case** male

**case** female

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

**return** lhs.rawValue < rhs.rawValue

}

}

**class** Person: CustomStringConvertible, Equatable{

**var** firstName: String = ""

**var** lastName = ""

**var** age = 1

**var** gender = Gender.male

**var** fullName: String{

**get**{

**return** firstName + lastName

}

}

//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**(firstName: String, lastName: String){

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

}

//Person实例对象可以用==和!=进行比较

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

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

}

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

**return** !(lhs == rhs)

}

//直接用print输出

**var** description: String{

**return** "\*Person：全名：\(fullName) 年龄: \(age) 性别: \(gender)\n"

}

//新增run方法

**func** run() {

print("Person \(**self**.fullName) is run!!!")

}

}

//(1)测试

print("\n(1)测试")

**var** onePerson = Person(firstName: "及", lastName: "等", age: 16, gender: Gender.male) //指定构造函数

**var** twoPerson = Person(firstName: "二", lastName: "产生") //便利构造函数

**var** threePerson = Person(firstName: "及", lastName: "等")

**var** fourPerson = Person(firstName: "二", lastName: "产生", age: 18, gender: Gender.male)

print(onePerson)

print(twoPerson)

print(threePerson)

onePerson.run()

**func** equal(**\_** one: Person, **\_** two: Person){

**if**(one == two){

print("\(one.fullName)和\(two.fullName)为同一个人")

}**else**{

print("\(one.fullName)和\(two.fullName)不为同一个人")

}

}

equal(onePerson, twoPerson)

equal(onePerson, threePerson)

equal(twoPerson, fourPerson)

/\*

(2)从Person分别派生Teacher类和Student类：

a)Teacher类增加属性title，实例可以直接用print输出；

b)Student类增加属性stuNo，实例可以直接用print输出；

\*/

**class** Teacher: Person{

**var** title: String = "good"

//指定构造函数

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

**self**.title = title

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

}

//重写父类指定构造函数

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

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

}

//直接用print输出

**override** **var** description: String{

**return** "\*Teacher：全名：\(fullName) 年龄: \(age) 性别: \(gender) title: \(title)\n"

}

//重载run方法

**override** **func** run() {

print("Person \(**self**.fullName) is run!!!")

}

}

**class** Student: Person{

**var** stuNo: String = "20170000"

//指定构造函数

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

**self**.stuNo = stuNo

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

}

//直接用print输出

**override** **var** description: String{

**return** "\*Student：全名：\(fullName) 年龄: \(age) 性别: \(gender) stuNo: \(stuNo)\n"

}

//重载run方法

**override** **func** run() {

print("Person \(**self**.fullName) is run!!!")

}

}

//(2)测试

print("\n(2)测试")

**var** p = Person(firstName: "李", lastName: "晋", age: 16, gender:Gender.female)

**var** p1 = Person(firstName: "金", lastName: "啦啦")

**var** p2 = Person(firstName: "刘", lastName: "哈")

p.run()

**var** t = Teacher(title: "hello",firstName: "张", lastName: "非得", age: 20, gender: Gender.female)

**var** t1 = Teacher(firstName: "李", lastName: "晋", age: 28, gender:Gender.male)

**var** t2 = Teacher(firstName: "阳", lastName: "杀") //当父类的所有指定初始化器实现，父类的便捷初始化器自动继承

**var** t3 = Teacher(firstName: "光", lastName: "东方")

**var** t4 = Teacher(firstName: "李", lastName: "憨憨", age: 20, gender:Gender.female)

t.run()

**var** s = Student(stuNo: "2017110517",firstName: "何", lastName: "灵", age: 20, gender: Gender.male)

// var s1 = Student(firstName: "李", lastName: "晋") //未实现父类的指定初始化器，所以便利初始化器未继承

**var** s1 = Student(stuNo: "2017110617",firstName: "凯", lastName: "好卡", age: 10, gender: Gender.female)

**var** s2 = Student(stuNo: "2017110517",firstName: "东", lastName: "德", age: 17, gender: Gender.male)

s.run()

print(p)

print(p1)

print(p2)

print(t)

print(t1)

print(t2)

print(t3)

print(t4)

print(s)

print(s1)

print(s2)

/\*

(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；

\*/

print("\n(3)测试")

**var** personArray: [Person] = [p,t,s,s1,s2]

personArray.append(p1)

personArray.append(p2)

personArray += [t1, t2, t3, t4]

personArray.remove(at: 0)

print(personArray)

//department枚举

**enum** Department:String{

**case** teacher

**case** student

}

//协议

**protocol** SchoolProtocol{

**func** lendbook()

**var** department:Department{**get**}

}

**extension** Teacher: SchoolProtocol{

**var** department: Department {

**get** {

Department.teacher

}

}

**func** lendbook() {

print("teacher lendbook")

}

}

**extension** Student: SchoolProtocol{

**var** department: Department {

**get** {

Department.student

}

}

**func** lendbook() {

print("student lendbook")

}

}

/\*

(4)对数组执行以下要求：

a)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；

b)对数组按以下要求排序并输出：age、fullName、gender+age；

\*/

print("\n(4)测试")

**var** personDict: [String : Int] = ["Person": 0, "Teacher": 0, "Student": 0]

//统计

**for** item **in** personArray{

**if** item **is** Teacher{

personDict["Teacher"]! += 1

}**else** **if** item **is** Student{

personDict["Student"]! += 1

}**else**{

personDict["Person"]! += 1

}

}

print(personDict)

//排序

//按age排序

personArray.sort(){

$0.age < $1.age

}

print("----按age排序:\n\(personArray)")

//按fullName排序

personArray.sort(by: {(person1: Person,person2: Person) -> Bool **in**

**return** person1.fullName < person2.fullName

})

print("----按fullName排序:\n\(personArray)")

//按gender+age排序

personArray.sort{

**if** $0.gender < $1.gender{

**return** **true**

}**else** **if** $0.gender == $1.gender && $0.age < $1.age{

**return** **true**

}**else**{

**return** **false**

}

}

print("----按gender+age排序:\n\(personArray)")

**5.实验结果的分析与评价（该部分如不够填写，请另加附页）**

1. 通过此次实验一，更加了解了闭包、扩展、泛型，知道了闭包在filter，map，reduce函数中的使用，学会了使用扩展和泛型。
2. 实验二让我大致掌握了git的分支操作，查看分支，创建分支，切换分支，删除分支；以及如何进行版本回退和查看历史版本
3. 实验三学会了枚举、类、派生、协议等相关知识，进行了方法的重写，与协议的实现。