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

1. **实验目的及要求**
2. 用 Swift语言编写类和子类
3. 掌握类和子类的定义及使用
4. 掌握git的基本使用命令
5. **实验要求**
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:（重点：闭包）

1. 找出从1-10000的所有质数，将找出的质数存入可变数组，对数组进行正反排序并输出结果(采用5种以上的方法实现)；
2. 掌握版本控制git的基本使用
   1. git init 初始化版本库
   2. git add . 添加当前目录下的所有文件到暂存区
   3. git commit -m 提交暂存区的文件到版本库

* 作业2:（枚举、类、派生）

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；
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. 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. 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；
5. 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:（重点：闭包）

1. 找出从1-10000的所有质数，将找出的质数存入可变数组，对数组进行正反排序并输出结果(采用5种以上的方法实现)；

* 程序代码：

import UIKit

 /\*

（1）找出从1-10000的所有质数，将找出的质数存入可变数组，对数组进行正反排序并输出结果(采用5种以上的方法实现)；

\*/

var arr = [Int]()

for i in 2...10000{

    var flag = 0

    for j in 2..<i{

        if i == 2{

            print("j的值：\(j)")

        }

        if i % j == 0{

            flag = 1

            break;

        }

    }

    if flag == 0{

        arr.append(i)

    }

}

print(arr)

//--1.排序(闭包函数)

//-------1.1 降序

func changeDesc(varA: Int, varB: Int) -> Bool{

    return varA > varB

}

arr.sort(by: changeDesc)

print("1，降序排列为：\(arr)\n")

//-------1.2 升序

func changeAsc(\_ varA: Int,\_ varB: Int) -> Bool{

    return varA < varB

}

arr.sort(by: changeAsc)

print("1，升序排列为：\(arr)\n")

//--2.排序（内联闭包表达式）

//--------2.1 降序

arr.sort(by: {(varA: Int, varB: Int) -> Bool in

    return varA > varB

})

print("2，降序排列为：\(arr)\n")

//--------2.2 升序

arr.sort(by:{(varA: Int, varB: Int) -> Bool in

    return varA < varB

})

print("2, 升序排列为：\(arr)\n")

//--3.排序（根据上下文推断类型）

//----------3.1 降序

arr.sort(by: {varA, varB in

    return varA > varB

})

print("3.降序排列为: \(arr)\n")

//----------3.2 升序

arr.sort(by: {varA, varB in return varA < varB})

print("3.升序排列为：\(arr)\n")

//--4.排序（单表达式闭包隐式返回，省略return）

//----------4.1 降序

arr.sort(by: {varA, varB in varA > varB})

print("4.降序排列为：\(arr)\n")

//----------4.2 升序

arr.sort(by: {varA, varB in varA < varB})

print("4.升序排列为：\(arr)\n")

//--5.排序（参数名称缩写）

//----------5.1 降序

arr.sort(by: {$0 > $1})

print("5.降序排列为：\(arr)\n")

//----------5.2 升序

arr.sort(by: {$0 > $1})

print("5.升序排列为：\(arr)\n")

//--6.排序（运算符方法）

//----------6.1 降序

arr.sort(by:>)

print("6.降序排列为：\(arr)\n")

//----------6.2 升序

arr.sort(by:<)

//--7.排序（尾随闭包--函数调用作为最后一个参数传递给函数）

//----------7.1 降序

arr.sort(){$0 > $1}

print("7.降序排列为：\(arr)\n")

//----------7.2 升序

arr.sort{$0 < $1}   //当闭包表达式是函数唯一的参数时，可以省略（）

print("7.升序排列为：\(arr)\n")

//--8.自定义的排序函数

func mySort<T>(myArray: [T],compare: (T,T) -> Bool) -> [T]{  //冒泡排序

    var myarr = myArray

    for i in 0..<myarr.count{

        for j in 0..<(myarr.count - i - 1){

            if !compare(myarr[j],myarr[j + 1]){

                let temp = myarr[j]

                myarr[j] = myarr[j + 1]

                myarr[j + 1] = temp

            }

        }

    }

    return myarr

}

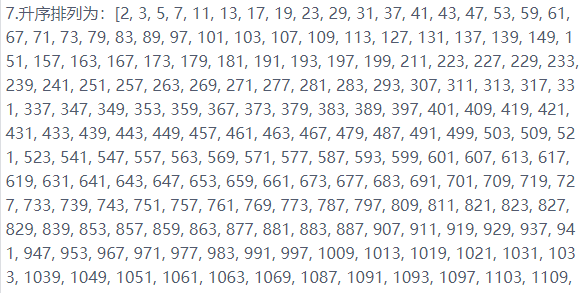
//----------8.1 降序

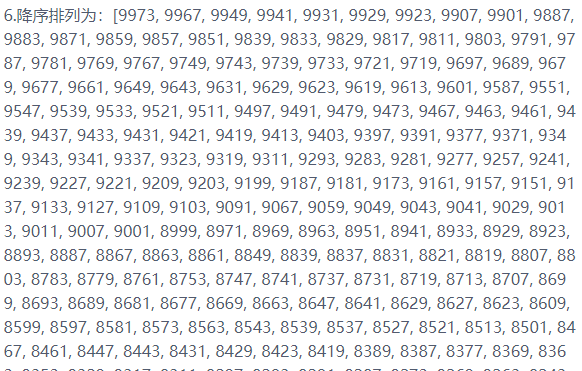
print(mySort(myArray: arr){$0 > $1})

//----------8.2 升序

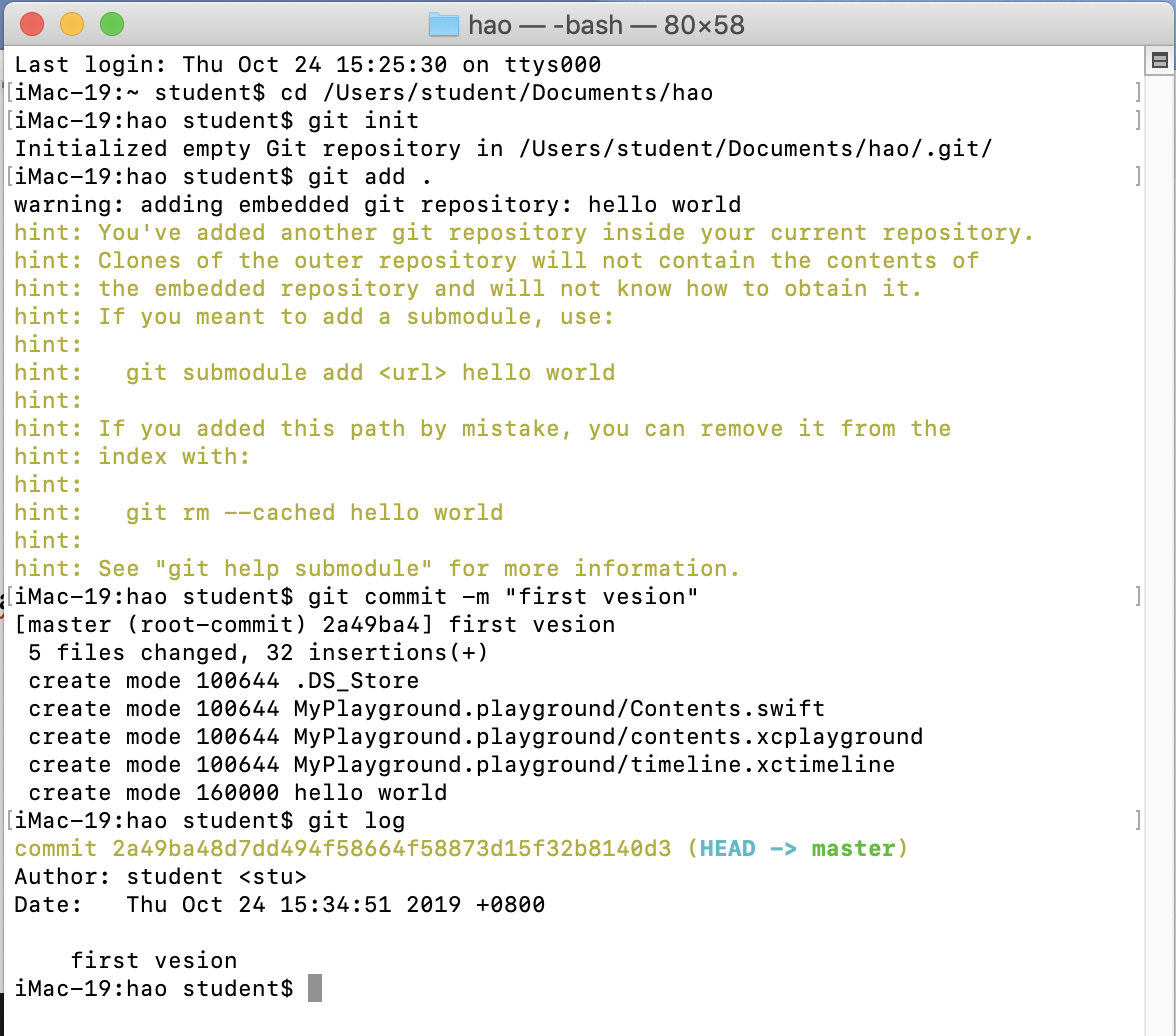
print(mySort(myArray: arr, compare: {$0 < $1}))

* 运行结果：





1. 掌握版本控制git的基本使用
   1. git init 初始化版本库
   2. git add . 添加当前目录下的所有文件到暂存区
   3. git commit -m 提交暂存区的文件到版本库



* 作业2:（枚举、类、派生）

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；

* 程序代码：

/\*

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"

    }

}

//(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)

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")

    }

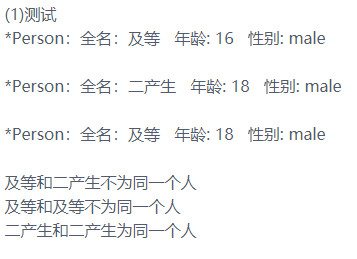
}

equal(onePerson, twoPerson)

equal(onePerson, threePerson)

equal(twoPerson, fourPerson)

* 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；
2. 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；
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；

* 程序代码：

/\*

(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"

    }

}

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"

    }

}

//(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")

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)

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)

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)

/\*

(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

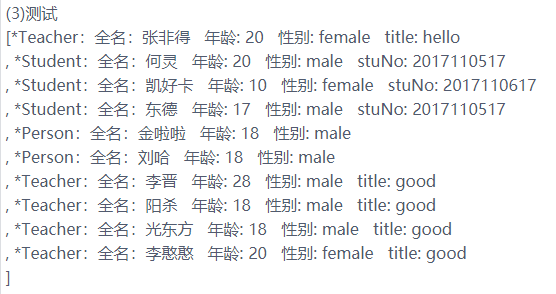
    }

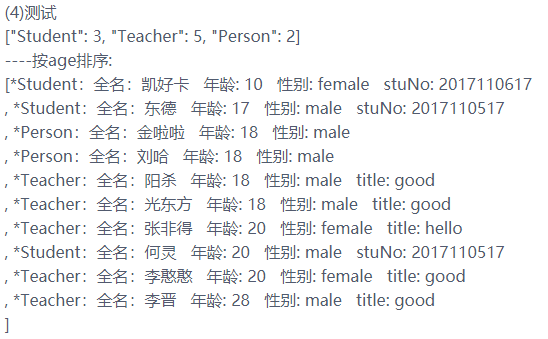
}

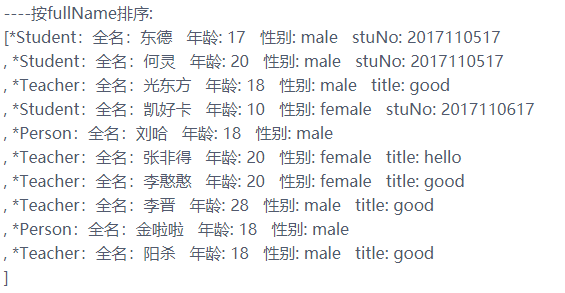
print("----按gender+age排序:\n\(personArray)")

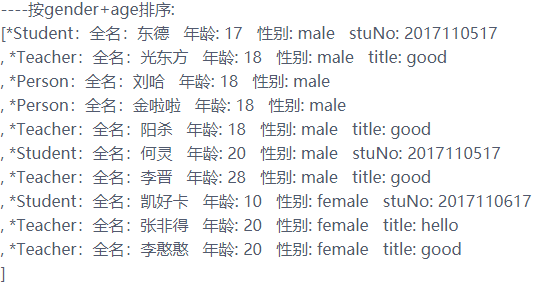
* 运行结果：











1. **实验结果的分析与评价（该部分如不够填写，请另加附页）**

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。