**集美大学计算机工程学院实验报告**

|  |  |  |
| --- | --- | --- |
| **课程名称**：Java程序设计 | **班级**：软件1912 | **实验成绩**： |
| **指导教师**：李　旺 | **姓名**：游嘉升 |  |
| **实验项目名称**：继承与多态 | **学号**：201921122050 | **上机实践日期**：10月16日 |
| **实验项目编号**： | **组号**： | **上机实践时间**： 2 学时 |

**一、目的**

熟悉类的继承；掌握多态性编程；了解反射机制。

**二、实验内容与设计思想**

1. 抽象类Shape与泛型数组列表

(1)编写一抽象类(Shape)，有一个PI属性(final static)用于存放圆周率、用于求周长的抽象方法public double getPerimeter();和用于求面积的抽象方法public double getArea();。

(2)长方形类、三角形类与圆形类均为其子类，并各有各的属性，并均有获得其周长、面积的方法。还需要生成相应的setter/getter方法。

(3)在一测试类中，分别建立若干个子对象（对象类型自定，尽量用到所定义的类），并放入数组或者ArrayList中，然后将各种形状的对象的所有面积与长度累加输出。

(4)针对上面的各子类编写相应的equals,hashCode,toString(可自动生成，但需看懂代码，注意里面对double类型属性是如何进行比较的)。

**注意：所有的代码都应该放在合适的包中（包名自定）。**

**思考：为什么PI属性要使用final进行修饰？**

2. Staff, Teacher, SecurityGuard, Dean

(1)定义Staff类(职工)，添加如下属性(name, address, age, sex, salary, dateHired)，类型自定，其中salary为工资，dateHired为雇佣日期(java.util.Date或java.time.LocalDateTime类型)。生成相应的setter/getter方法。

(2)编写Teacher类(教师)，继承自Staff类，包含属性：department(系), speciality(专业), postAllowance(岗位津贴)。

(3)编写SecurityGuard类(保安)，继承自Staff类，包含属性：skills(专技), dangerousAllowance(高危津贴)。

(4)编写Teacher的一个子类Dean(院长)，包含属性：adminAward(行政奖金)。

(5)定义上述各类的getter/setter方法，并添加合适的构造方法。

(6)编写一个测试类，在测试类中添加若干个Staff, Teacher, SecurityGuard, Dean实例(个数及内容自定)，并在测试类中定义并测试如下方法：①编写一个方法private static void printName(Staff[] persons)打印出每个人的名字；②编写一个方法private static void printSalary(Staff[] staffs)打印出Staff类或者其子类对象的薪水(**注意：Staff的薪水只有salary，Teacher的薪水为salary+postAllowance，SecurityGuard的薪水为salary+dangerousAllowance，而Dean的薪水则为salary+postAllowance+adminAward**)；③编写一方法private static void sortBySalary(Staff[] staffs)，支持对Staff类及其子类按照各自的薪水降序排序；④编写一方法private static void sortByAge(Staff[] staffs)，对Staff对象按照年龄升序排序，再编写一个方法按name升序进行排序；⑤(选做)编写一方法sortByDateHired，支持对Staff类及其子类按照各自的dateHired升序排序，可以使用java.util.Date类的getTime方法或者java.time.LocalDateTime的compareTo方法。

**说明：排序暂时不要使用比较器（Comparable、Comparator等）！**

3. Account, CheckingAccount, SavingAccount

(1)定义Account类，模拟一个银行帐户，包括帐户名、余额、年利率、开户日期等成员。同时Account类拥有存款(deposit)和取款(withdraw)两个方法，**方法参数及返回值自定，但要综合考虑后续测试类中的调用规范**。

(2)创建Account类的两个子类：支票帐户(CheckingAccount)、储蓄帐户(SavingAccount)。其中支票帐户有一个透支限定额，但储蓄帐户不能透支。

(3)编写测试程序，创建若干Account、CheckingAccount、SavingAccount，打印帐号信息，同时测试存款、取款方法。

提示：这里要注意“取款”方法的设计，在逻辑上，支票帐户允许一定额度的透支，而普通帐户和储蓄帐户不能透支。可以合理使用覆盖设计该方法。

4. Rect, PlainRect

(1)定义一个矩形类Rect，包含两个protected属性：矩形的宽width和高height。一个带有两个参数的构造方法，用于将width和height属性初始化。一个不带参数的构造方法，将矩形的宽和高都初始化为10。还有两个普通方法：求矩形面积的方法getArea()和求矩形周长的方法getPerimeter()。

(2)继承Rect类编写一个具有确定位置的矩形类PlainRect，其位置用矩形的左上角坐标来标识，有两个属性：矩形左上角坐标startX和startY。两个构造方法：一个是带4个参数的构造方法，用于对startX、startY、width和height属性初始化。一个是不带参数的构造方法，将矩形初始化为左上角坐标、长和宽都为0的矩形。

(3)为PlainRect类添加一个方法：public boolean isInside(double x, double y)，用于判断某个点(x,y)是否在矩形内部。若点在矩形内，则返回true, 否则，返回false。

(4)编写测试类，测试上述功能。

**思考：考虑如果三角形、圆形都需要添加isInside方法，你会怎么扩展代码，以达到代码重用的目的（当然也要符合类本身的设计准则及封装性原则）？**

**三、实验使用环境**

Java version "13.0.2"

IntelliJ IDEA 2020.2.2 x64

**四、实验步骤和调试过程**

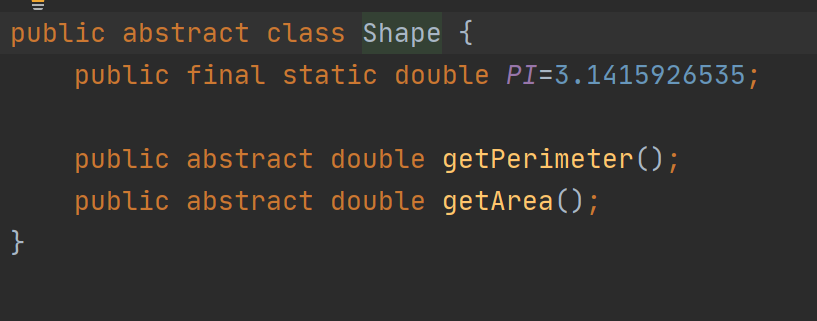
1. 抽象类Shape与泛型数组列表（代码具体见附件shape）

1. 设计思想：

定义一个抽象类Shape包含PI属性，抽象方法getPerimeter和getArea，再定义三个图形子类，具体实现两个抽象方法和equals,hashCode和toString方法。测试时定义Shape型数组列表，再生成不同的子类对象进行面积与长度的加减。

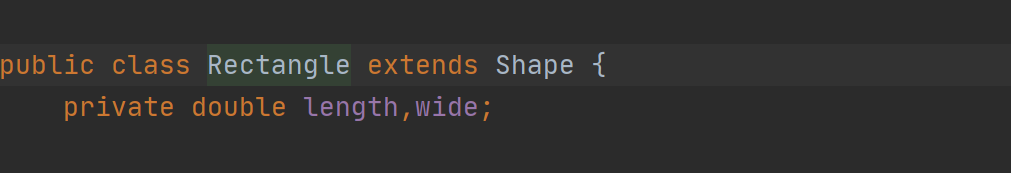
1. 关键代码：

Shape类的定义：

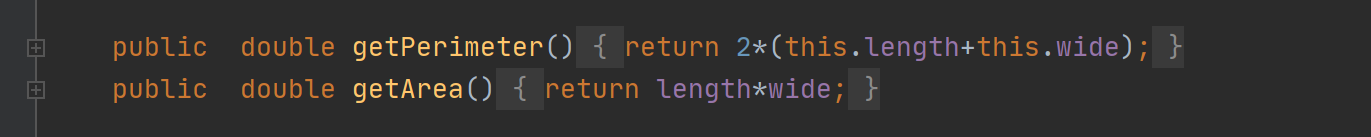


继承子类选择Rectangle作为示范

子类的定义：



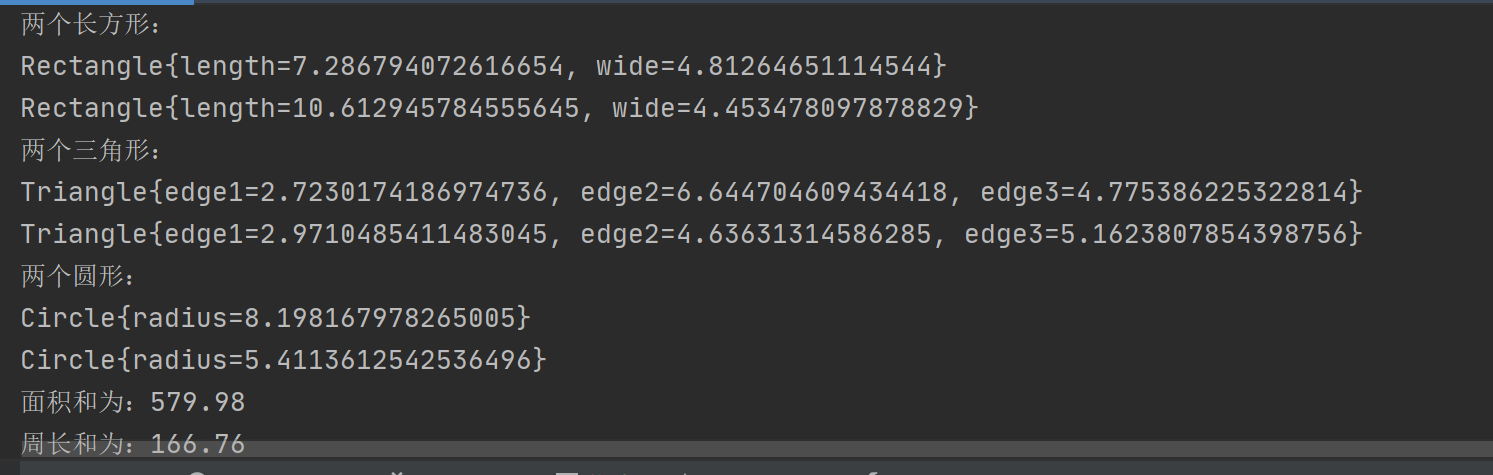
抽象方法的实现：



泛型数组列表的使用：



1. 测试数据：一种子类图形生成两个，具体参数由随机数方法生成。
2. 测试结果：



思考：为什么PI属性要使用final进行修饰？

使用关键词final修饰使其不可以被轻易修改或继承，可以明确禁止该变量不会改变。

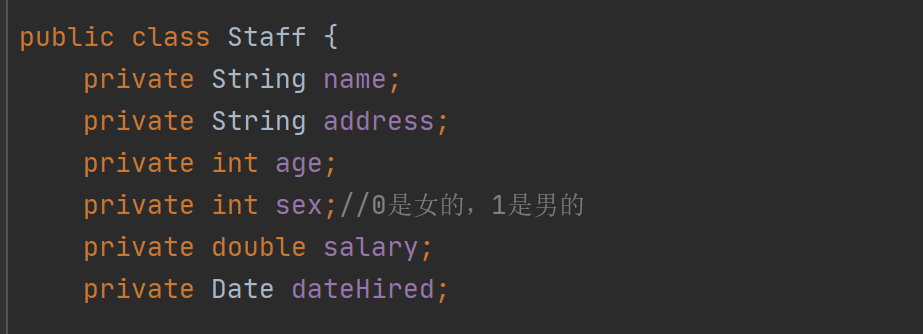
2. Staff, Teacher, SecurityGuard, Dean（代码具体见附件staff）

1. 设计思想：

定义一个父类Staff包含员工基本属性，并定义getSalary方法。定义三个子类继承Staff的员工基本属性，并在不同的子类中重写getSalary方法。在Test类中定义多种输出对象属性的printf方法以及根据对象属性来排序的多种sort方法，在主函数中生成一个类生成两个对象作为测试。

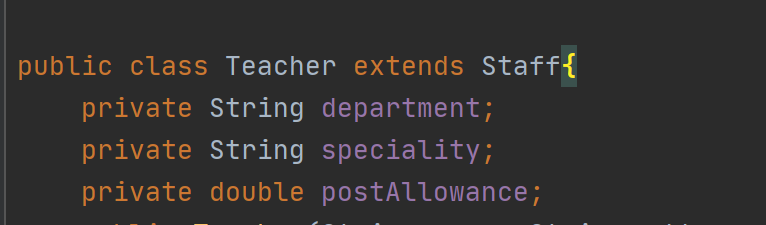
1. 关键代码：

Staff类的定义：

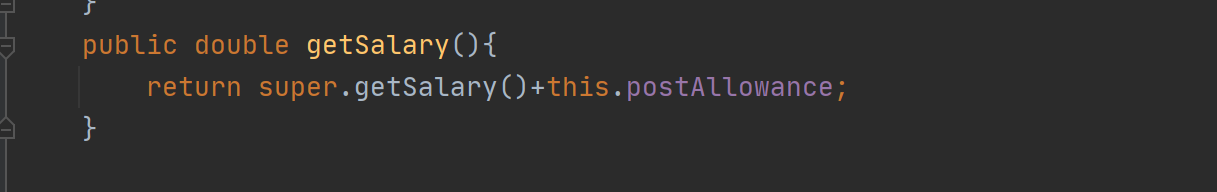


继承子类选择Teacher作为示范

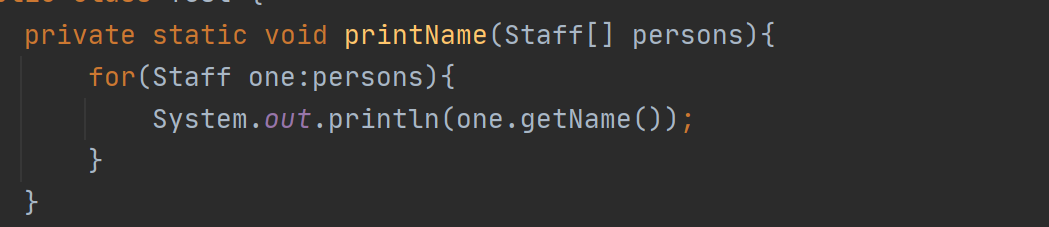
子类的定义：



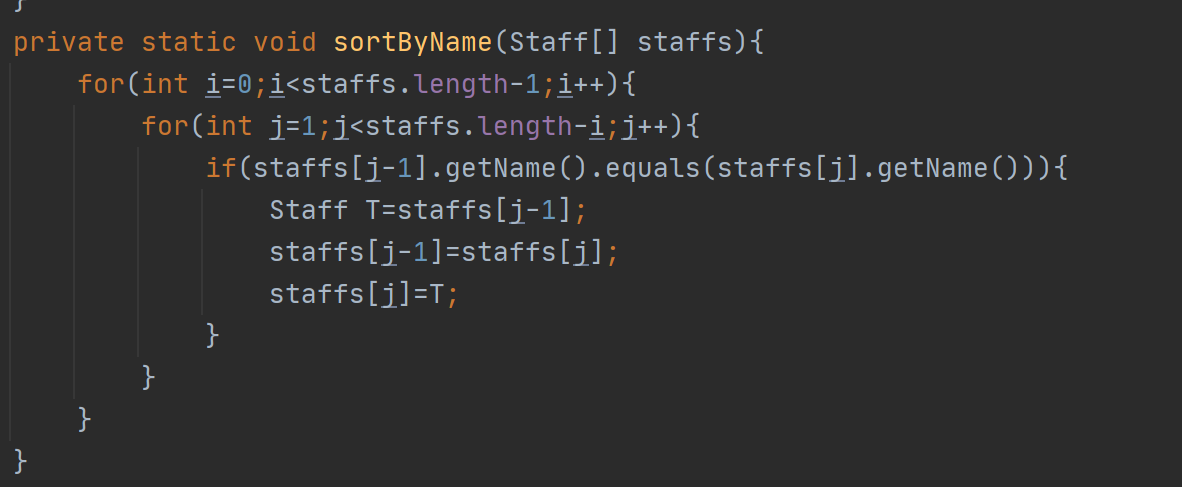
getSalary方法的重写：



Print方法选择pirntName作为示范



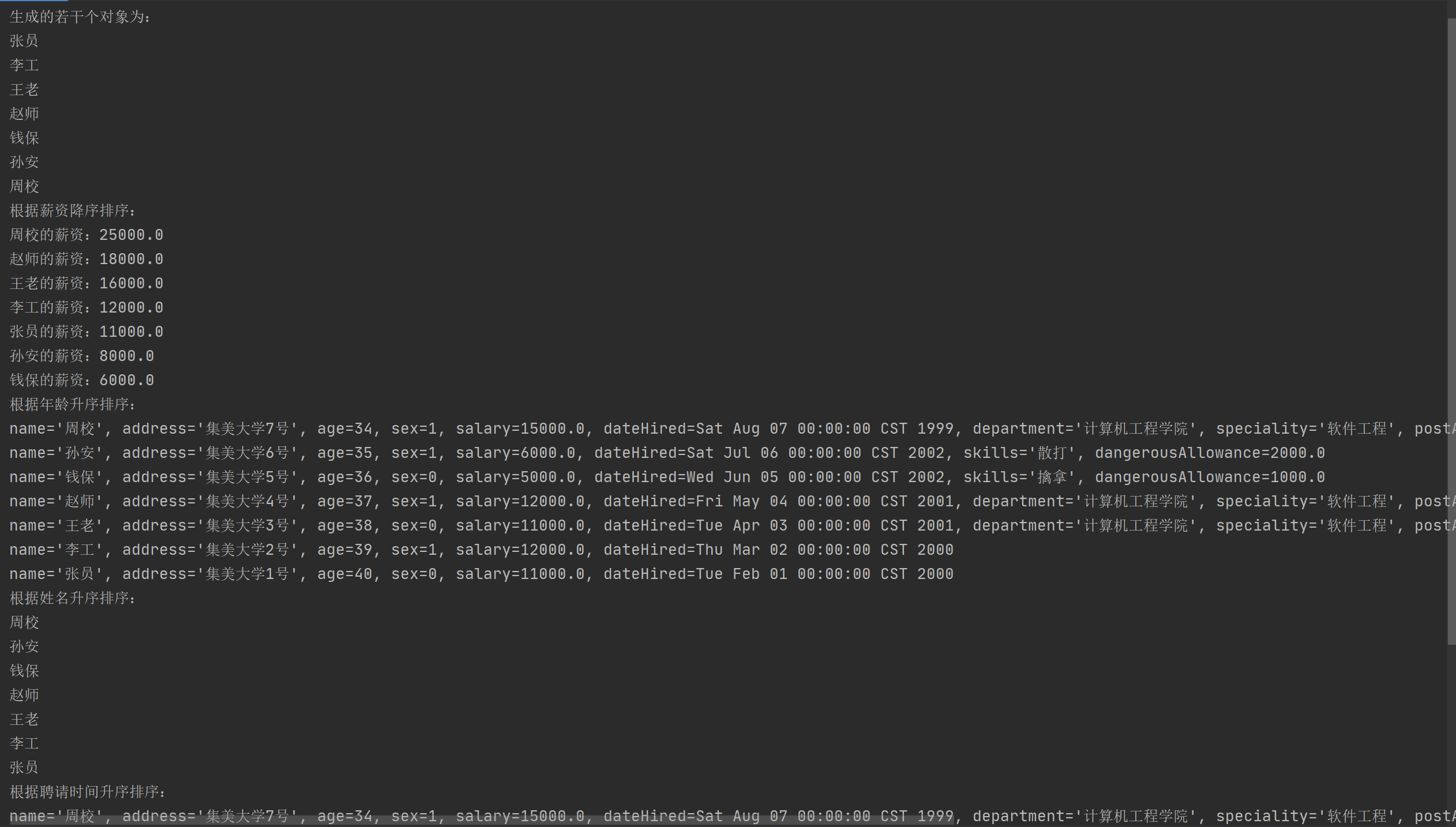
Sort方法选择sortByName作为示范（采用冒泡排序）



1. 测试数据：



1. 测试结果：



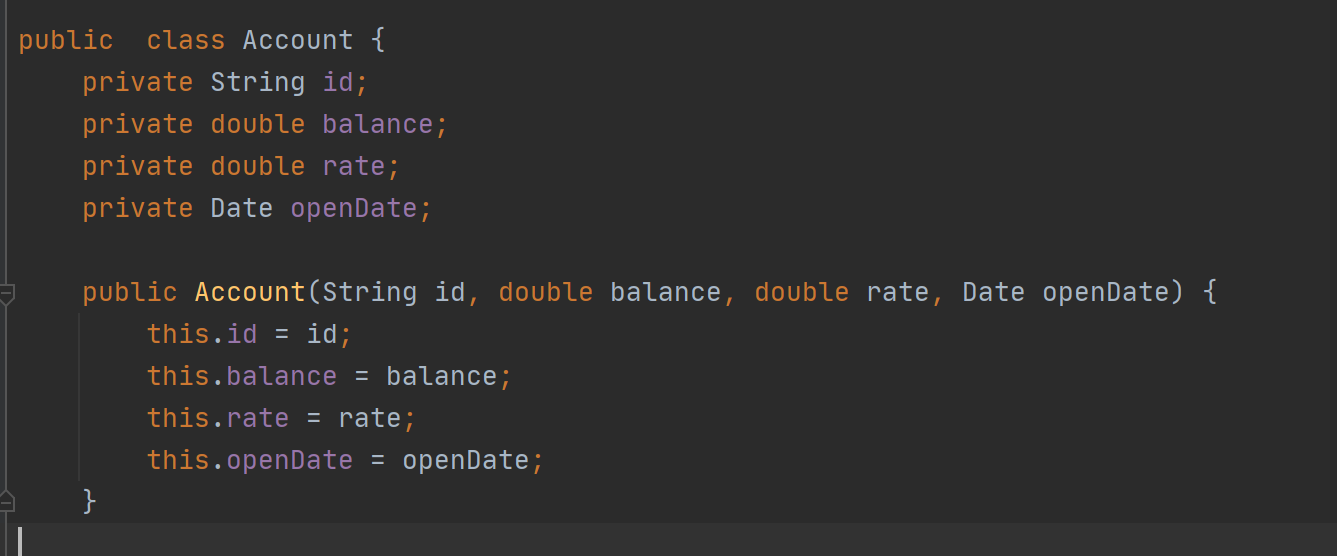
3. Account, CheckingAccount, SavingAccount

1. 设计思想：

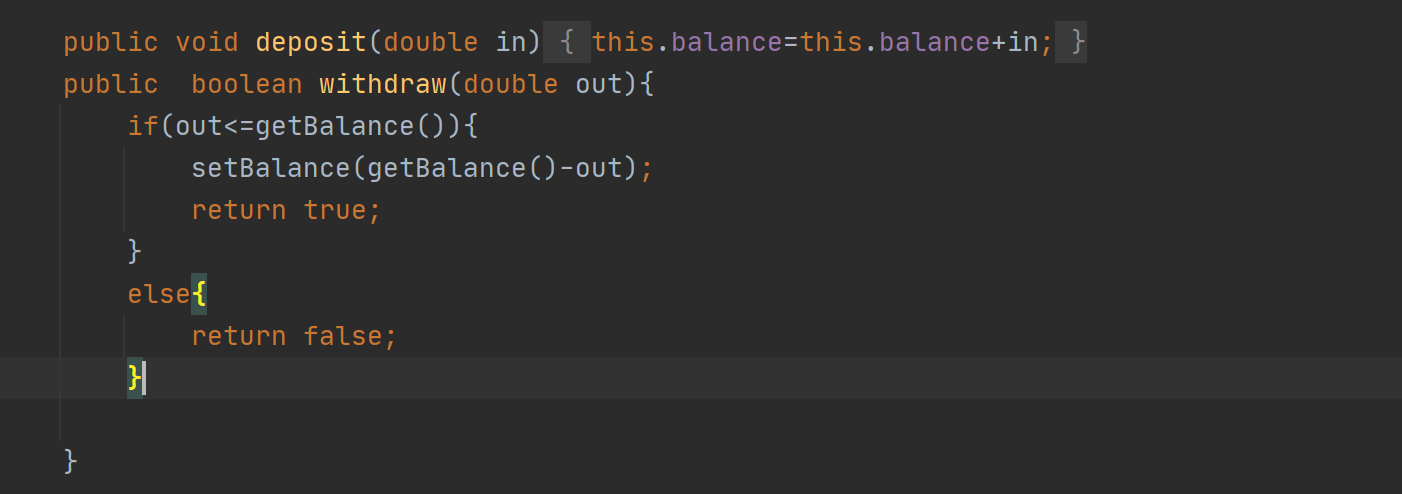
定义一个父类Account包含账户基本属性，以及deposit和withdraw方法，再定义两个子类SavingAccount和CheckingAccount，重写ChecingAccount类中的deposit和withdraw方法，在Test类中测试存取款方法。

1. 关键代码：

Account类的定义：



Deposit和withdraw方法的定义

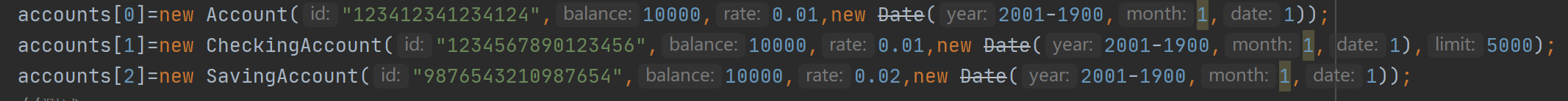


继承子类以CheckingAccount作为示范

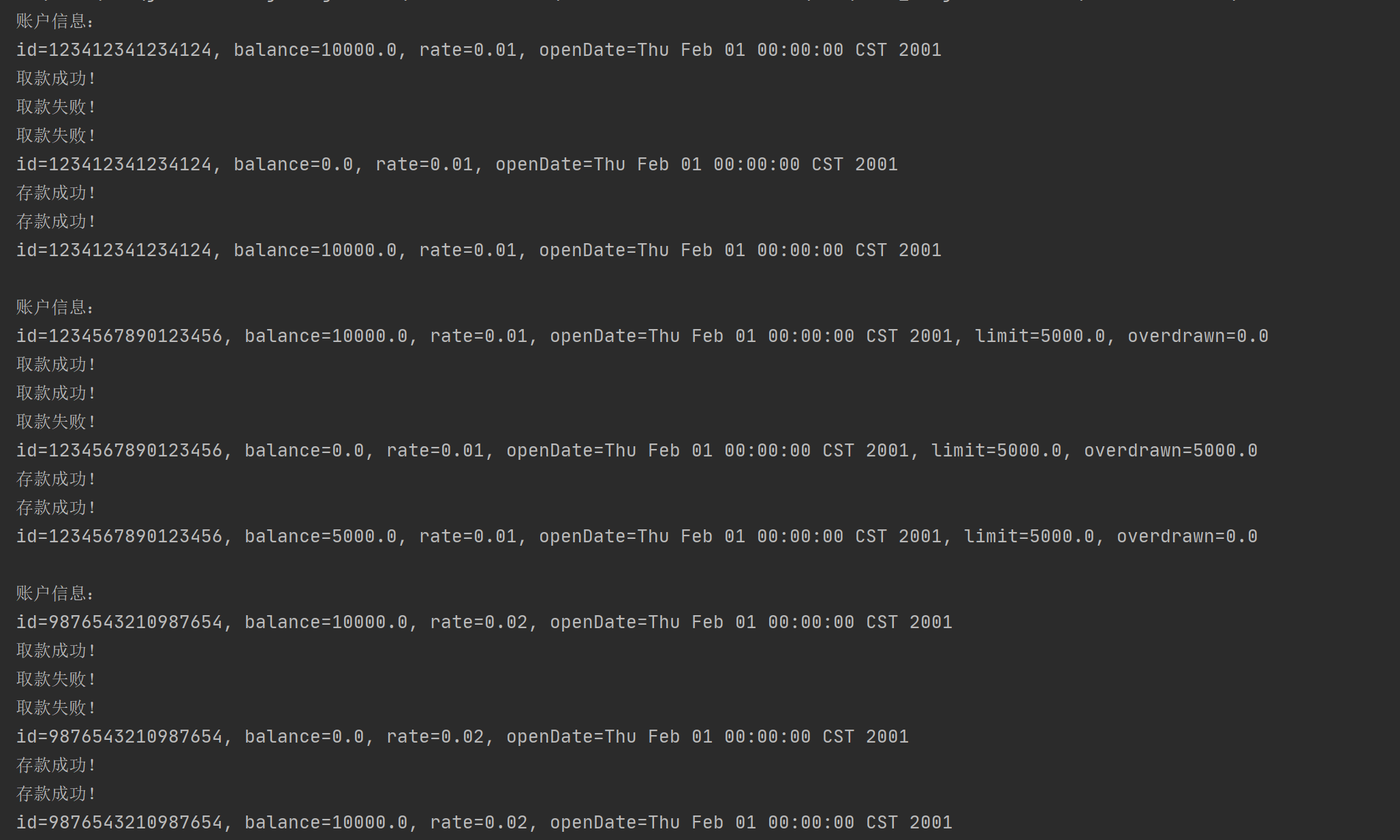
Deposit和withdraw方法的重写



1. 测试数据：



1. 测试结果：

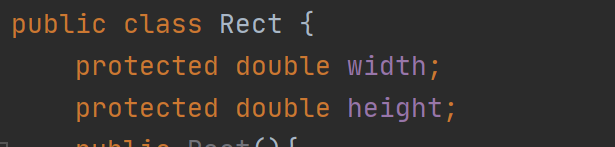


4. Rect, PlainRect

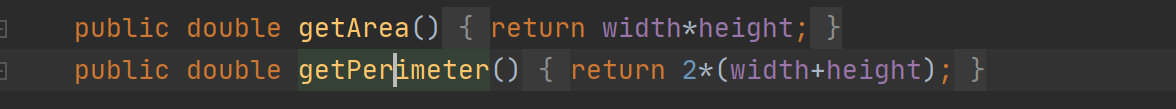
1）设计思想：定义一个Rect类包含矩形类属性，以及getArea和getPerimeter的抽象方法，再定义PlainRect继承于Rect类，以及isInside的抽象方法。

2）关键代码：

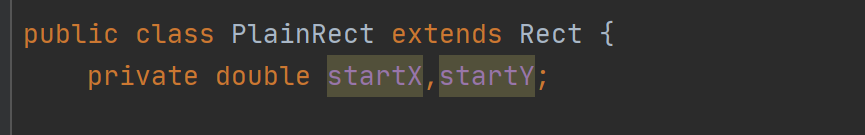
Rect类的定义



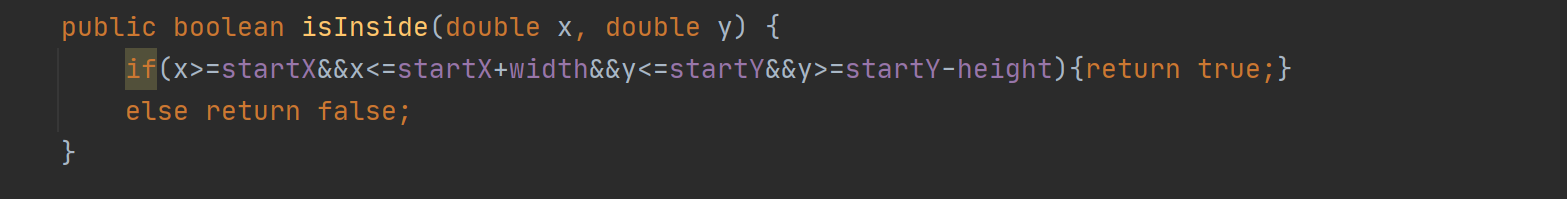
方法getArea和getPerimeter的定义



继承子类PlainRect的定义

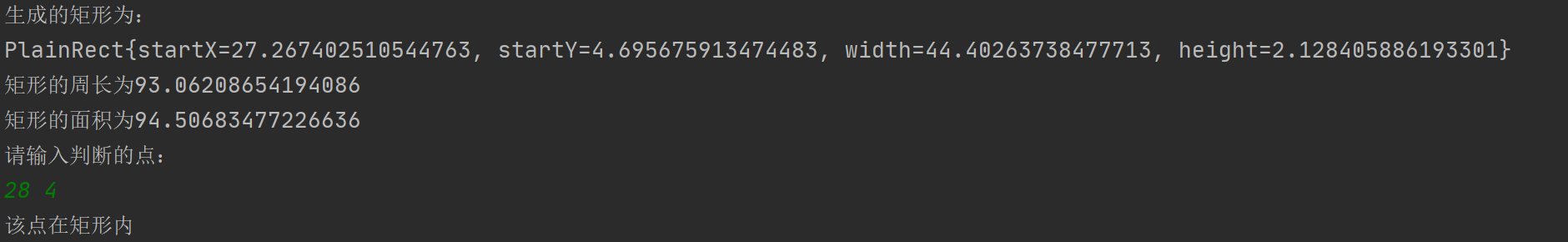


方法isInside的定义



3）测试数据：由随机数方法随机生成

4）测试结果：



思考：考虑如果三角形、圆形都需要添加isInside方法，你会怎么扩展代码，以达到代码重用的目的（当然也要符合类本身的设计准则及封装性原则）？

需要一个抽象类Shape以及抽象方法定义，其他图形为Shape类的子类，在子类中具体实现不同类的不同方法，有相似的部分在抽象类中定义即可。

**五、实验小结**

1．实验中遇到的问题及解决过程

①分不清抽象类与父类以及抽象方法和重写

解决过程：百度了一下，抽象类一定是父类且不能被实例化，需要有具体子类实现抽象方法，重写是父类中有该方法的实体，在子类中根据需求而写的同名方法，而抽象方法在定义时可以不需要实体，但在抽象类的子类中需要有定义。

2．实验中产生的错误及原因分析

对于继承与多态的概念不够熟悉，会分不清或者不会使用一些具体的概念.

3．实验体会和收获。

进一步体会到封装方法的优势，使写的代码更有逻辑性。

**六、附录**

[1][美] 凯.S.霍斯特曼（Cay S. Horstmann）.Java核心技术·卷 I（原书第10版）[M].机械工业出版社