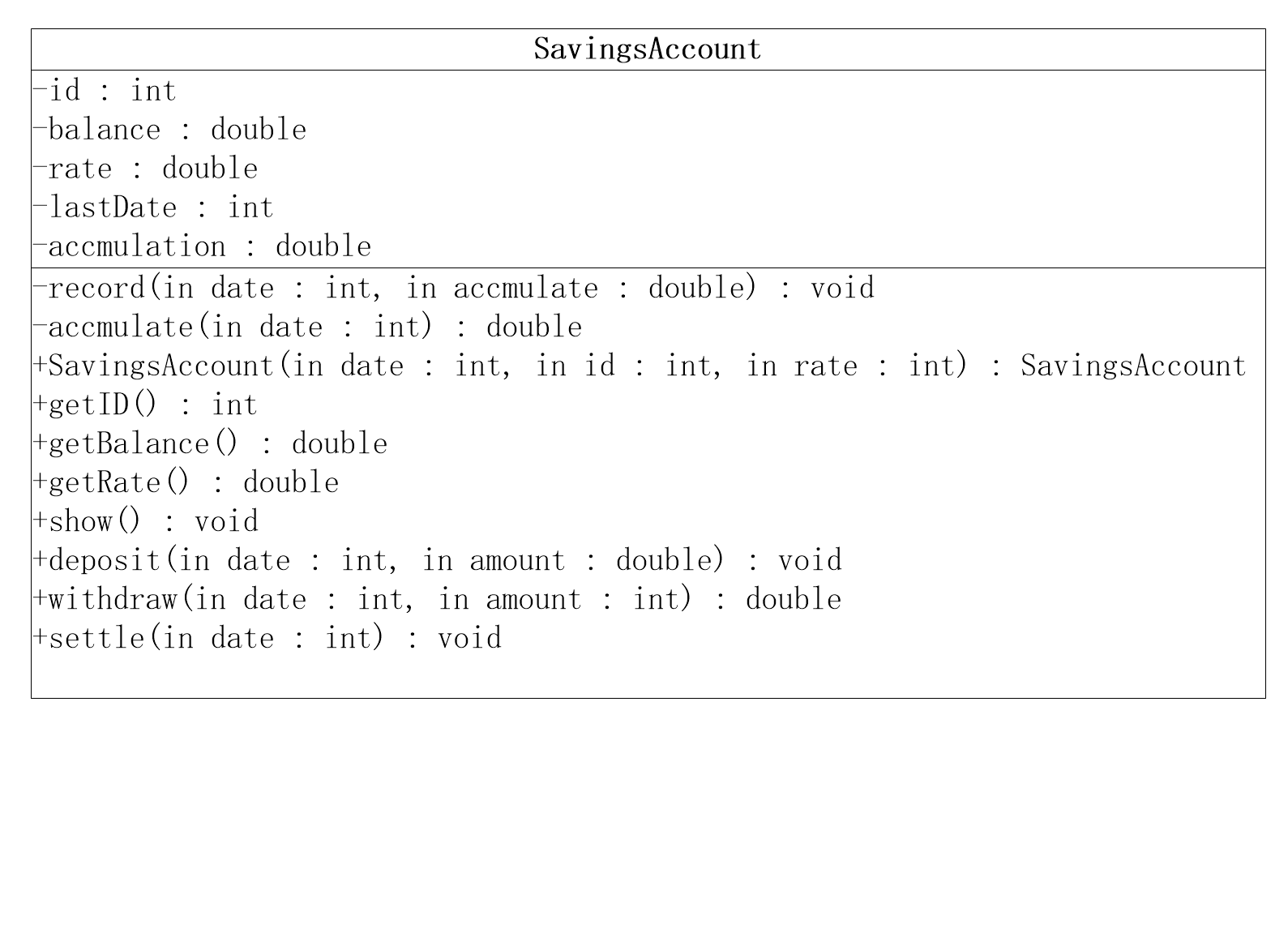
课程设计报告一：个人银行账户管理系统

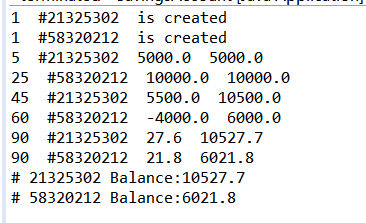
* 课程设计要求与目的
* 模仿个人银行账户管理系统的C++版本（第4章-第9章），使用Java语言重新实现该系统，比较C++与Java在实现上的异同，熟练掌握Java基础及语法。
* 根据系统需求的演化，逐步完善个人银行账户管理系统的功能，改进代码，体会面向对象思想的封装、继承、多态特性在实际系统中的应用，初步掌握使用Java编写可复用、可扩展、可维护代码的基本技能。
* 课程设计进展记录
* 个人银行管理系统版本0.1（对应第4章记录）
* 系统需求

功能：显示账户信息，存款，取款，结算利息。

* 系统设计 
* 系统实现

一个人可以有多个账户，每个账户包含账号（id）、余额（balance）、年利率（rate）等数据成员。为保险起见，数据成员全部设为私有。并提供相应方法可使外界完成查看余额、存钱、取钱等操作。操作方法包括：显示账户信息（show）、存款（deposit）、取款（withdraw）、结算利息（rate）等操作。当银行卡里的钱小于要取款的数目时，会提示错误信息。可以实现一个简单的银行账户管理系统。

* 系统测试



运行结果：

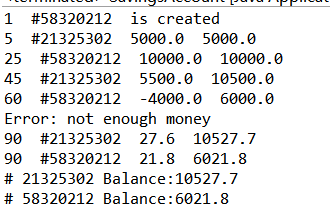


样例：

运行结果与样例相比，运行结果大体相同，只是保留小数点位数不同。但本次测试并没有把所有分支测试到。比如取钱时，银行卡里的钱少于所取的钱的情况。所以，测试用例优化如下：

在主函数中添加：sa1.withdraw(61,7000);

得到的测试结果为：



可以证明，当银行卡里的钱少于所取的钱的情况时，会提示错误。与设计相符。

1.5体会心得

通过此次实验，我理解了Java中创建对象和C++不同。Java创建对象的一句话实际上包含了三个内容：创建对象、声明对象的引用、初始化引用。引用使用起来比指针方便，易懂。学会了Java中的格式化输出例如保留两位小数。对编译时的报错信息有了大致的了解。C++中可以将函数的具体实现写到类的外面（成员函数）同样也有内联成员函数（将具体实现写在类中），但Java中只能将成员函数写在类中。Java中的System.out.println()使用’+’可以完成字符串的连接，因为在内部已经实现了对+运算符的重载。Java中的final和C++中的const等价。

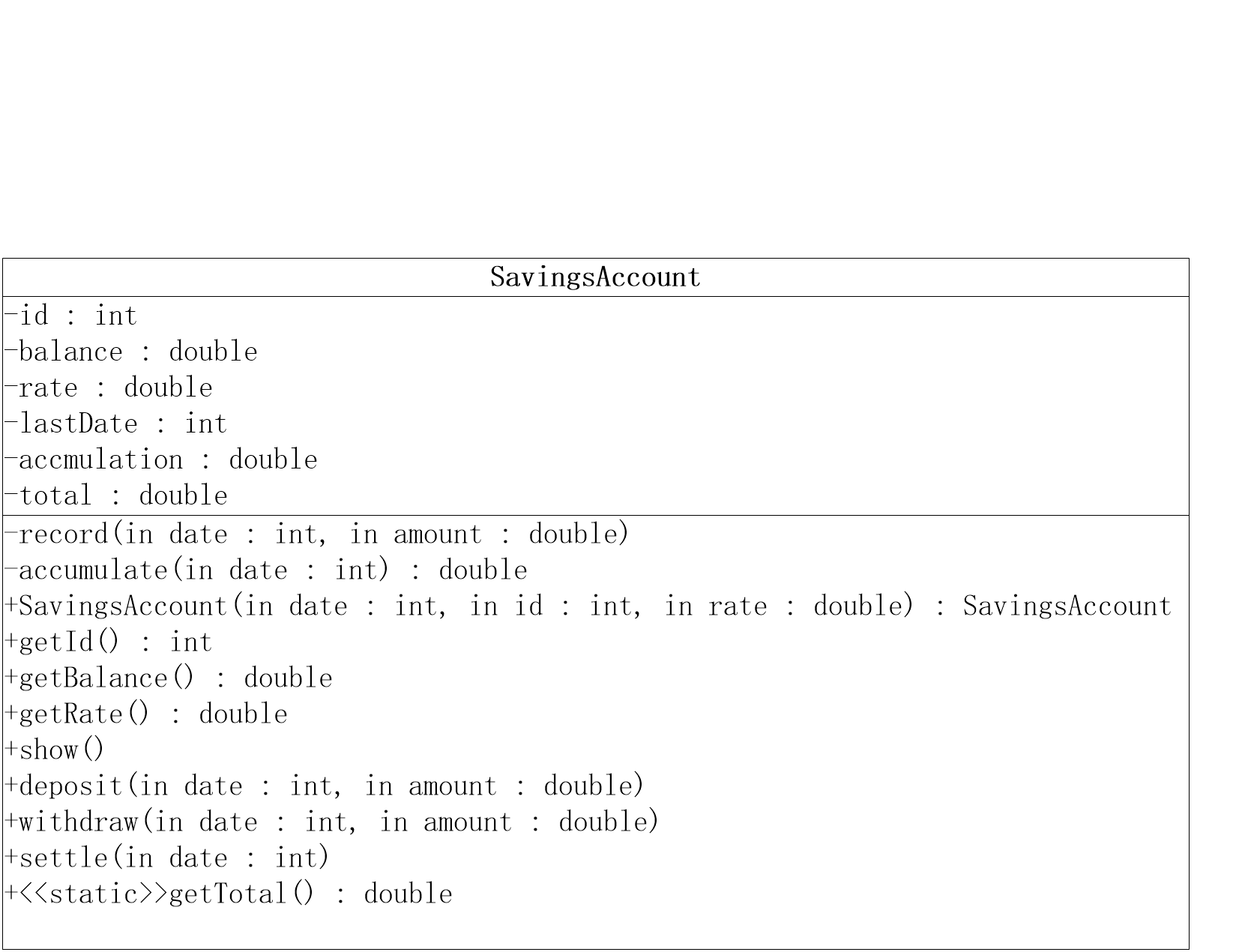
* 个人银行管理系统版本0.2（对应第5章记录）
* 系统需求

需求同系统设计0.1

* 系统实现

将total变为static，其余同0.1

UML图设计如下：



* 系统测试

测试结果同0.1版本结果

* 体会心得

Java与C++的静态数据成员没有太大区别。

* 个人银行管理系统版本0.3（对应第6章记录）

3.1系统需求

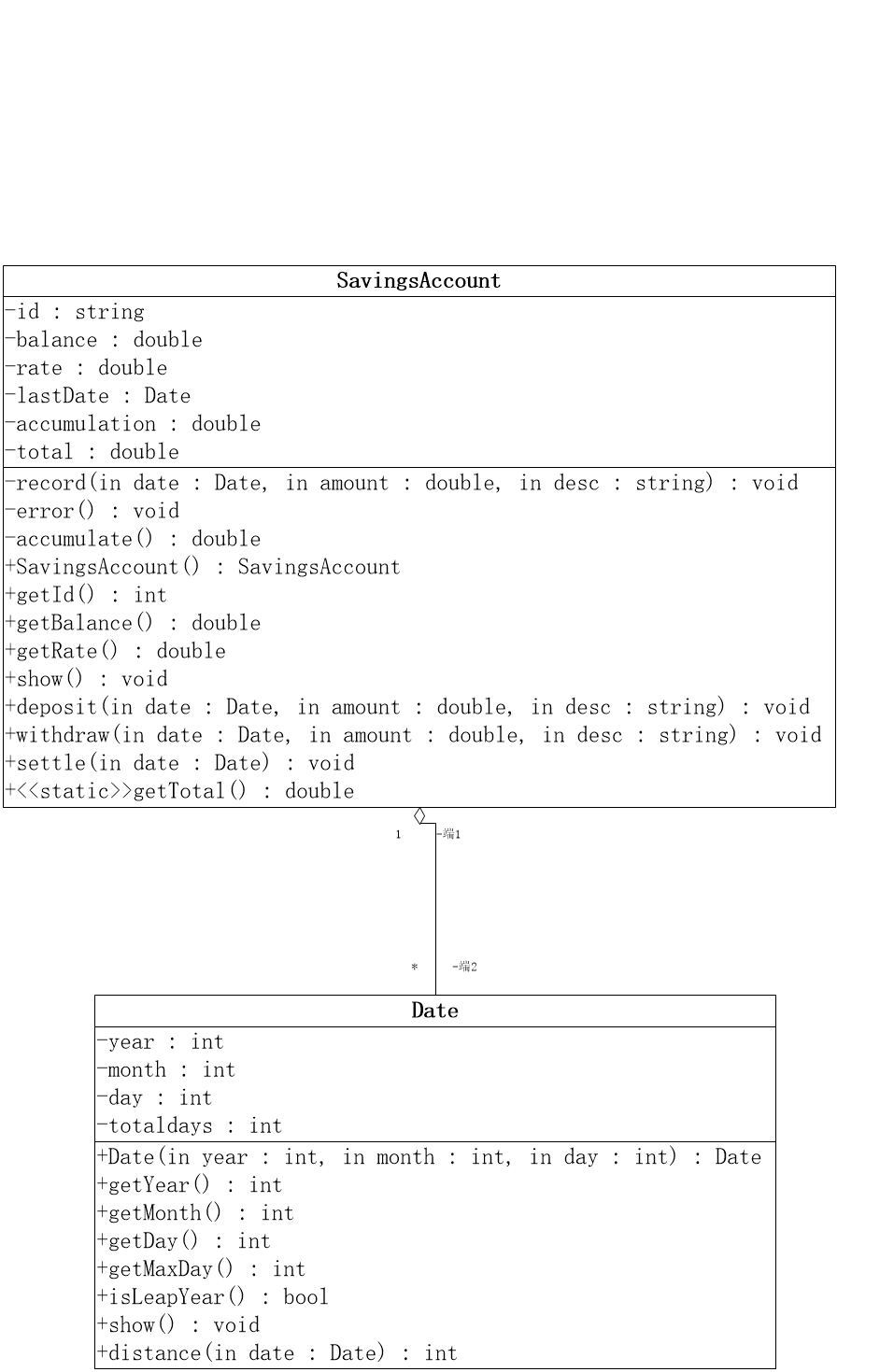
①用字符串来表示银行账号

②在SavingsAccount中专门增加一个方法来报告错误，简化错误信息的输出。

③可以将多个同类型的账户放到对象数组中，在对多个账户进行操作可以使用循环，避免了代码的重复。

④将日期封装，用一个类来表示。这样有利于形成高内聚、低耦合的代码。但也会在计算相对天数上带来不便。

3.2系统设计

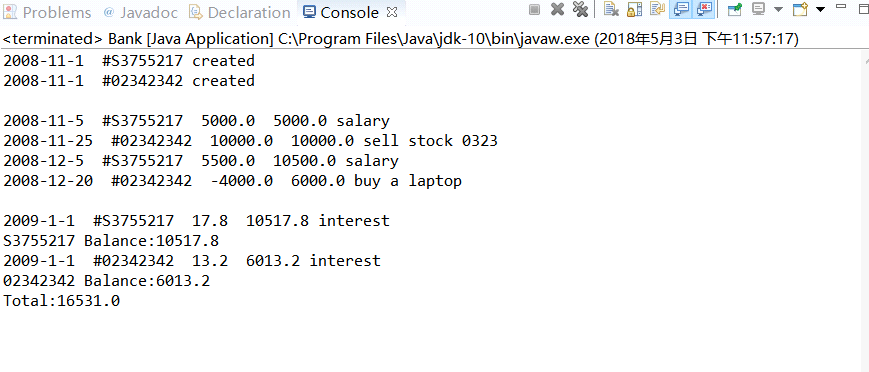


3.3系统实现

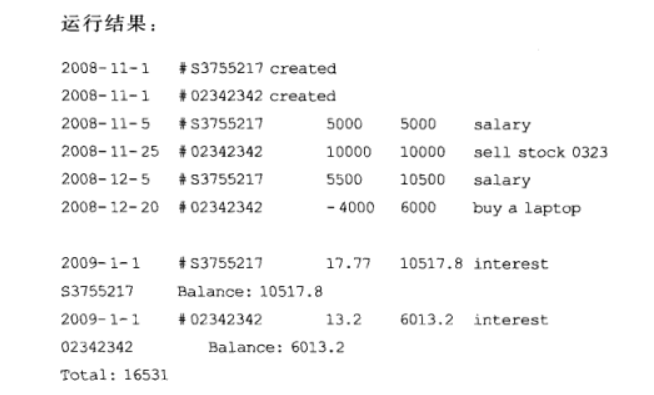
为SavingsAccount类专门增加一个方法，用来报告错误。当需要输出错误信息时，需要调用该方法并为该方法提供相应参数。可以将多个同类型的账户放到对象数组中，在对多个账户进行操作可以使用循环，避免了代码的重复。将日期封装，用一个类来表示。这样有利于形成高内聚、低耦合的代码。假定日期结算时间为1月1日。可以推导出相对日期。在推导时注意闰年的周期。可以创建一个数组专门用来记录本月最后一天到0的天数，计算利息时会用到天数，只需查数组表即可。

3.4系统测试

运行结果：



样例：

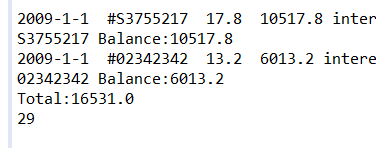


结果分析：该结果针对一般情况是成立的，但对于输入不合法的情况，闰年的366天的情况并没有测试到。

白盒测试：

①2008年为闰年，所以2月有29天。测试结果如下：

在主函数中添加System.out.println(date1.getMaxDay());



闰年判断正确

②对于时间不合法的情况：

在主函数中添加Date date1=**new** Date(2008,2,30);

结果为：

3.5体会心得

Java中的错误退出：System.exit(1)

* 个人银行管理系统版本0.4（对应第7章记录）

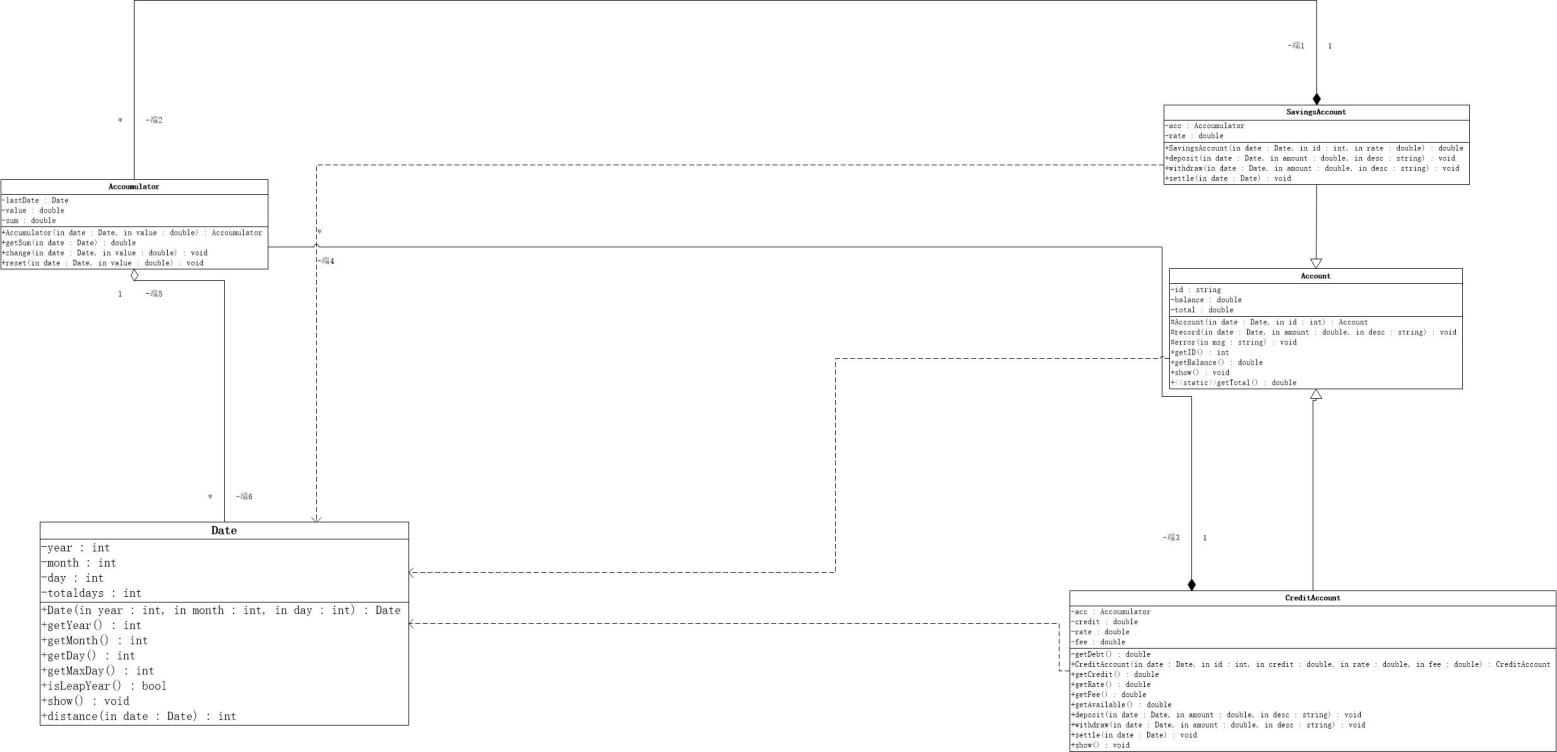
4.1系统需求

（1）进一步明确账户是信用账户，其特点是允许透支，使用信用账户透支需要支付利息；

（2）信用账户的利率以日为单位，假定从透支第一天起开始计算利息；

（3）信用账户每年需交一次年费，假定在每年1月1日结算时扣缴年费。

4.2系统设计



4.3系统实现

1. Account类需要完成：

构造方法

记录一笔账，date为日期，amount为金额，desc为说明

报告错误信息

显示账户信息

1. CreditAccount需要完成：

构造方法

获得欠款额

获得可用信用额度

存入现金

取出现金

结算利息和年费，每月1日调用一次该方法

1. Accumulator类需要完成：

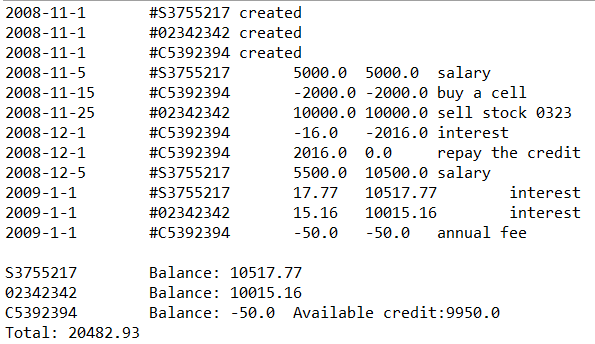
构造方法

获得日期date的累加结果

改变日期date

重置日期、余额

4.4系统测试



4.5体会心得

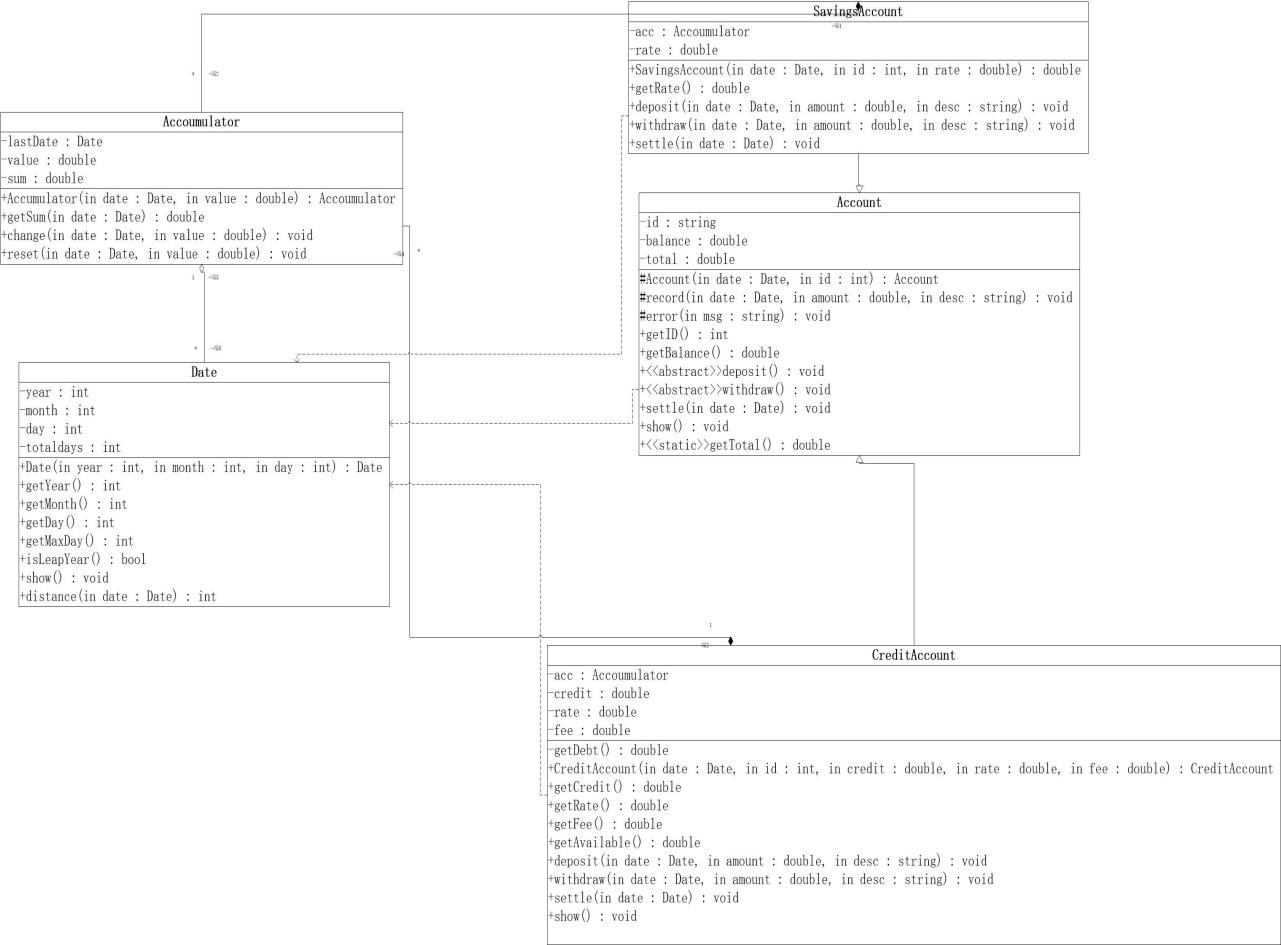
1. 此版本中SavingsAccount类继承了Account类，CreditAccount类也继承了Account类；
2. 基类Account的record方法被派生类继承后，派生类处理每一笔具体账目时可以调用该方法来改变余额并输出账目信息；
3. CreditAccount在show方法中需要输出额外的信息，故该类重新定义了一个同名的方法，该方法是先调用基类的show()后在执行本方法体将额外信息输出的；
4. 本版本将各个账户作为独立的对象来声明，是因为它们是不同派生类的实例，只能分别操作，即分别调用各自的方法。

* 个人银行管理系统版本0.5（对应第8章记录）

5.1系统需求

1. 解决各个账户对象无法通过数组来访问、无法用循环来完成的问题；
2. 解决不同派生类的deposit,withdraw,settle等方法彼此独立只有知道一个实例的具体类型后才能调用这些方法的问题；
3. 由用户手动输入账户编号、对账户的功能操作、以及数值等。

5.2系统设计



5.3系统实现

1. 将show()声明为抽象方法；
2. 在Account类中添加deposit，withdraw，settle这3个方法声明为抽象方法，将Account类声明为抽象类；

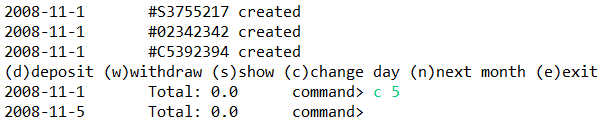
5.4系统测试

本例采用黑盒测试技术，通过等价类划分的方法对程序功能进行覆盖。

1. 有效等价类

* 输入： c 5

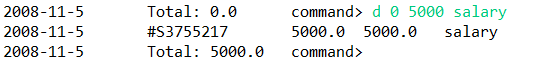
输出：



实现设置本月日期和当前所有账户余额总和。

* 输入： d 0 5000 salary

输出：

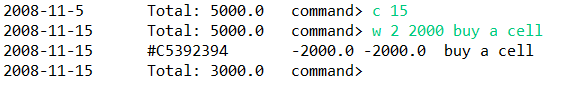


实现对某一个账户存入现金及其信息描述。

* 输入： c 15

w 2 2000 buy a cell

输出：



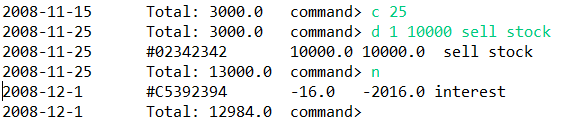
实现对某一个账户取出现金及其信息描述。

* 输入： c 25

d 1 10000 sell stock

n

输出：



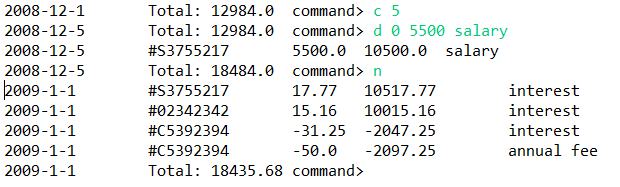
实现当前状态进入下一个月的日期改变以及信用账户的利息扣除。

* 输入： c 5

d 0 5500 salary

n

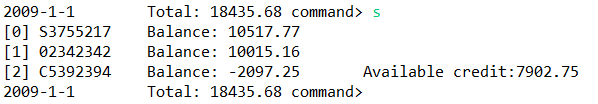
输出：



实现进入第二年时普通账户的年息与本金总和、信用账户利息与本金总和以及年费扣除显示。

* 输入： s

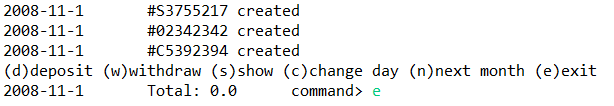
输出：



实现查询当前各账户信息的功能。

⑦ 输入： e

输出：



结束程序，main线程终止。

1. 无效等价类

① 输入： w 0 15000 buy antique

输出：



当取出现金大于当前余额时抛出现金余额不足的异常。

② 输入： c 32

输出：



当设置日期不合理（日期小于0或大于32）时抛出无效日期异常。

5.5体会心得

1. 将Account类声明为抽象类，通过向上转型和动态绑定使得我们可以通过统一的方式来操作各个账户；
2. 输入所执行的操作时采用Scanner方法（扫描器），可以获得多种数据类型。

* 个人银行管理系统版本0.6（对应第9章记录）

6.1系统需求

可以动态改变数组的大小，大小随时可变。

6.2系统设计

（同上一版本）

6.3系统实现

通过ArrayList动态数组类进行操作：

ArrayList <Account> accounts=**new** ArrayList <Account>(0);//新建ArrayList数组

accounts.size(); //账户总数

6.4系统测试

1. 储蓄账户基本功能测试：

①基本功能：每个账户都有一个ID，且ID为s开头，可以完成存钱、取钱、利息结算的功能。

②注意事项：每年1月1日结算利息，年利率可以手动输入，不能透支，没有年费，初始日期为2008-11-1。

③测试用例：

a s S001 0.015 //a(add)—添加账户,s储蓄账户,ID:S001,年利率0.015

//初始日期为2008-11-1

d 0 10000 in //给第0个账户存入10000元，in为说明

n //跳到下个月

n //跳到下个月

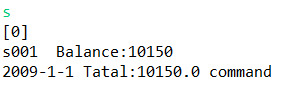
s //显示账户余额等信息

④理论输出结果：

s001 Balance:10150

2009-1-1 Tatal:10150.0 command

⑤实际输出结果：



⑥结果分析：

本例中rate为0.015 本金10000

所以到1月1日，本息和应为10000\*（1+0.015）=10150元

1. 信用账户基本功能测试：

①基本功能：每个账户都有一个ID（ID为C开头），可以完成存钱、取钱、贷款计算年费、利息结算的功能。

②注意事项：每年1月1日结算利息；月利率可以手动输入但要注意存钱不算利息，只有贷款使时才计算利息；可以透支；年费可以手动输入；初始日期为2008-11-1。

③测试用例：

a c C001 10000 0.005 50 //添加ID为C001的信用账户，信用额度为

//10000，年费为50，月利率为0.005，初始日期为2008-11-1

w 1 5000 buy a car

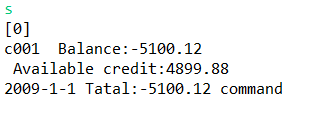
n //跳到下个月

n //跳到下个月

s //显示账户余额等信息

④理论上输出的结果：-5100.12

⑤实际输出结果：



⑥结果分析：

该信用账户的信用额度为10000，借了5000，月利率为0.005

所以，跳转至下个月的时候应还金额变为：5000\*（1+0.005）=5025

再跳转至下个月时，已经到了2009年1月1日，还应该加上50元年费，应还金额变为5025\*（1+0.005）+50=5100.125

1. 异常测试：

①时间异常：

输入：c 31

输出：Invalid day

②信用账户透支金额过多：

输入：a c c001 10000 0.005 50

w 0 10001 buy a car

输出：Error(#c001):not enough credit

③储蓄账户余额不够支付：

输入：a s s002 0.015

d 0 10000 s

w 0 10001 b

输出：Error(#s002):not enough money

6.5 体会心得

使用ArrayList数组可以随时调整数组大小。

保留小数点后两位可以调用 Math.floor

* 课程设计总结

本次课程设计最终实现了一个初级版个人银行账户管理系统，在管理账户方面使用了动态数字ArrayList。在对账户进行操作时，需要通过数组下标进行区别，而不是账户ID，这也是本管理系统的一个缺点。账户分为两种：储蓄账户和信用账户。储蓄账户可以在每年1月1日获得利息，且不需要缴纳年费。随时存钱，只要取钱的金额不大于余额，也可以随时取钱。信用账户可以透支，需要缴纳年费，也需要按月缴纳利息。在面向对象方面，使用了抽象方法、继承、抽象类等技术。在信用账户和储蓄账户中，有一个父类，来对两种账户进行统一管理。在主方法里面，使用向上转型，来对不同账户进行操作。使用Java中的动态数组，以完成账户的增添、删除等操作。遇到的问题：首先是输入。Scanner方法可以完成输入操作。而且可以接受各种类型的输入，例如int double String 等等。第二个主要的问题就是对Scanner中的方法不熟。这个通过上网找资料或查看Java api文档可以解决。第三个主要问题是对ArrayList动态数组及里面的方法不是很熟悉。这个问题同样可以查看Java api得以解决。

在进行整个课程设计的过程中，我收获了很多。其中包括：对输入输出操作有了更深层次的理解。对于不懂的问题可以先上网进行搜索，或者查看Java api。也学会了使用visio画UML图。对Java的引用也有了更深层次的理解。对以后学习新的编程语言掌握了学习的方法。有助于以后的学习。