个人银行账户管理系统课程设计报告

一、课程设计版本记录

1. 个人银行账户管理系统版本 0.1（对应第 4 章记录）

1.1 系统需求

实现存入，取出，结算操作

1.2 系统设计

抽象出储蓄账户类（SavingAccount）

1.3 系统实现

字段：

```
int id;                 //账号
double balance;      //余额
double rate;         //存款的年利率
int lastDate;        //上次变更余额的时期
double accumulation;     //余额按日累加之和
```

函数：

Deposit,withdraw,settled,show

1.4 系统测试

```
1   #21325302 is created
1   #58320212 is created
5   #21325302   5000.005000 5000.005000
25  #58320212   10000.005000    10000.005000
45  #21325302   5500.005000 10500.010000
60  #58320212   -3999.995000    6000.010000
90  #21325302   27.642013   10527.652013
90  #58320212   21.785841   6021.795841
#21325302   Balance: 10527.652013
#58320212   Balance: 6021.795841
```

1.5 体会心得

在修改代码的过程中对文件结构的布置出现了点问题，后来用 IDEA 默认的 java 文件结构后问题解决。

2. 个人银行管理系统版本 0.2（对应第 5 章记录）

2.1 系统需求

计算所有账户的总额

2.2 系统设计

添加静态数据成员和函数计算获取所有账户总额

2.3 系统实现

```
static double total;      //所有账户的总金额
```

```
static double getTotal() { return total; }
```

2.4 系统测试

```
1   #21325302 is created
1   #58320212 is created
5   #21325302   5000.005000 5000.005000
25  #58320212   10000.005000    10000.005000
45  #21325302   5500.005000 10500.010000
60  #58320212   -3999.995000    6000.010000
90  #21325302   27.642013   10527.652013
90  #58320212   21.785841   6021.795841
#21325302   Balance: 10527.652013
#58320212   Balance: 6021.795841
```

2.5 体会心得

通过修改代码，对静态有了新得认识，静态字段能够被子类函数所修改，例如实际问题中求所有账户总额，将账户总额设置为 static 能够很好的实现功能。

3. 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

添加字符串，对象数组

3.2 系统设计

在 Main 类中创建 accounts 数组

3.3 系统实现

```
SavingsAccount[] accounts = new SavingsAccount[2];
```

3.4 系统测试

```
1   #21325302 is created
1   #58320212 is created
5   #21325302   5000.005000 5000.005000
25  #58320212   10000.005000    10000.005000
45  #21325302   5500.005000 10500.010000
60  #58320212   -3999.995000    6000.010000
90  #21325302   27.642013   10527.652013
90  #58320212   21.785841   6021.795841
#21325302   Balance: 10527.652013
#58320212   Balance: 6021.795841
```

3.5 体会心得

使用了数组之后对于大量数据来说可以很做到很快的遍历，但对于项目而言，在开始之初若用数组储存用户信息要一开始就开很大的数组，浪费空间，并且对于数组来说可用的操作类型较少。

4. 个人银行账户管理系统版本 0.4（对应第 4 章记录）

4.1 系统需求

实现类的继承与派生，抽象出父类，增加子类

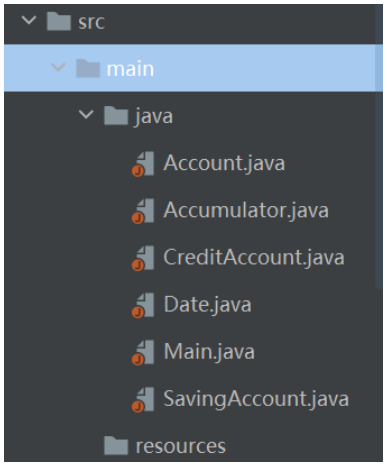
分别计算存钱的利率和欠款后的利率

4.2 系统设计

CreditAccount,Date,SavingAccount 派生于 Account

4.3 系统实现

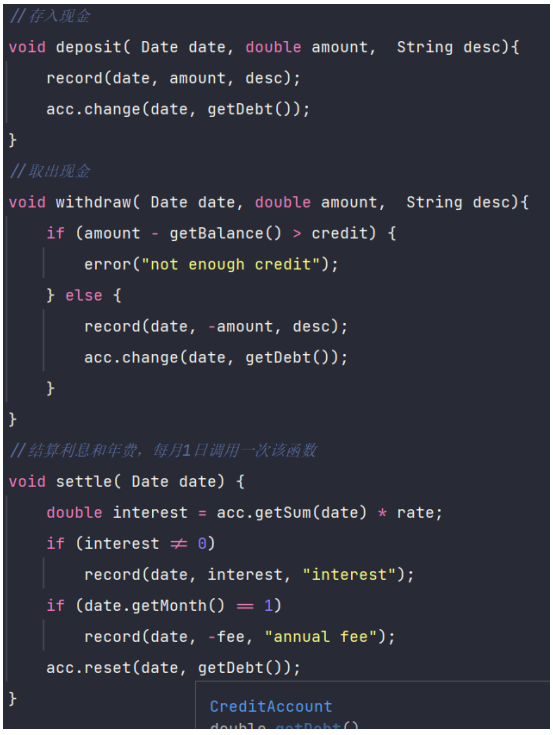
文件结构如图所示



Account 类的实现

```
public class Account {
    private
    String id;  //账号
    double balance; //余额
    static double total; //所有账户的总金额
    protected
    //供派生类调用的构造函数，id为账户
    Account(Date date, String id){
        this.id = id;
        this.balance = 0;
        date.show();
    }
    //记录一笔账，date为日期，amount为金额，desc为说明
    void record( Date date, double amount, String desc){
        amount = (amount * 100 + 0.5) / 100;    //保留小数点后两位
        balance += amount;
        total += amount;
        date.show();
        System.out.printf("\t#%s\t%.2f\t%f\t%s\n", id, amount, balance, desc);
    }
```

CreditAccount 中关键函数的实现：



```
//存入现金
void deposit( Date date, double amount,  String desc){
    record(date, amount, desc);
    acc.change(date, getDebt());
}
//取出现金
void withdraw( Date date, double amount,  String desc){
    if (amount - getBalance() > credit) {
        error("not enough credit");
    } else {
        record(date, -amount, desc);
        acc.change(date, getDebt());
    }
}
//结算利息和年费，每月1日调用一次该函数
void settle( Date date) {
    double interest = acc.getSum(date) * rate;
    if (interest != 0)
        record(date, interest, "interest");
    if (date.getMonth() == 1)
        record(date, -fee, "annual fee");
    acc.reset(date, getDebt());
}
```

SavingAccount 中关键函数的实现：

```
//存入现金
void deposit(Date date, double amount, String desc){
    record(date, amount, desc);
    acc.change(date, getBalance());
}
//取出现金
void withdraw(Date date, double amount, String desc){
    if (amount > getBalance()) {
        error("not enough money");
    } else {
        record(date, -amount, desc);
        acc.change(date, getBalance());
    }
}
//结算利息，每年1月1日调用一次该函数
void settle(Date date){
    double interest = acc.getSum(date) * rate   //计算年息
            / date.distance(new Date(date.getYear() - 1, 1, 1));
    if (interest != 0)
        record(date, interest, "interest");
    acc.reset(date, getBalance());
}
```

4.4 系统测试

```
1   #21325302 is created
1   #58320212 is created
5   #21325302   5000.005000 5000.005000
25  #58320212   10000.005000    10000.005000
45  #21325302   5500.005000 10500.010000
60  #58320212   -3999.995000    6000.010000
90  #21325302   27.642013   10527.652013
90  #58320212   21.785841   6021.795841
#21325302   Balance: 10527.652013
#58320212   Balance: 6021.795841
```

4.5 体会心得

和 c++实现继承的方法大部分一致。

5.

5.1 系统需求

Account 类的子类的多态性，实现用户输入功能

5.2 系统设计

一 将 Account 覆写的方法标明为 abstract,在子类中实现，如下图

```
abstract void deposit( Date date, double amount, String desc);
//取出现金，date为日期，amount为金额，desc为款项说明
abstract void withdraw( Date date, double amount, String desc);
//结算（计算利息、年费等），每月结算一次，date为结算日期
abstract void settle( Date date);
```

二 用户输入功能使用 switch 语句实现

5.3 系统实现

一 以 deposit 函数为例为例的多态性

```java
@Override
//SavingAccount
void deposit( Date date, double amount, String desc) {
    record(date, amount, desc);
    acc.change(date, getBalance());
}
```

```java
@Override
//CreditAccount
void deposit( Date date, double amount,  String desc){
    record(date, amount, desc);
    acc.change(date, getDebt());
}
```

覆写时养成随手写@Override 的习惯

二 使用 Scanner 实现从键盘读取用户输入

5.4 系统测试

```
Invalid date: 0-0
0-0 #S3755217 created
0-0 #02342342 created
0-0 #C5392394 created
(d)deposit (w)withdraw (s)show (c)change day (n)next month (e)exit
0-0 Total: 0.00 command> d
1
11.1
hhh
0-0 #02342342   11.11   11.105000   hhh
0-0 Total: 11.11    command> 0-0    Total: 11.11    command> c
21
Invalid day0-0  Total: 11.11    command> 0-0    Total: 11.11    command> n
Invalid date: 0-0
0-0 Total: 11.11    command> s
[0] S3755217    Balance: 0.0
[1] 02342342    Balance: 11.105
[2] C5392394    Balance: 0.0    Available credit:10000.0
```

5.5 体会心得

在此版本中可以看出一个项目的雏形，已经实现了用户的键盘输入。

在使用 Scanner 包时，通过查博客了解具体使用方法，阅读 java 源代码文档还是有些困难。

6.

6.1 系统需求

容器代替数组

6.2 系统设计

调用 ArrayList 包

6.3 系统实现

```
ArrayList<Account> accounts = new ArrayList<>();// 创建容器
```

6.4 系统测试

```
Invalid date: 0-0
(a)add account (d)deposit (w)withdraw (s)show (c)change day (n)next month (e)exit
0-0 Total: 0.00 command> a
s
12
1.1
0-0 #12 created
0-0 Total: 0.00 command> 0-0    Total: 0.00 command>
```

6.5 体会心得

在使用 ArrayList 中元素为 Account（父类），在向数组中添加元素是可以添加其子类!!!

二、课程设计总结

具体实现了：

用容器存储用户信息，

在父类 Account 的基础上派生两个子类并实现函数的多态性

实现用户输入的读取

等

在设计过程遇到了很多问题，其中最突出的就是查阅文档的问题，和代码注释不足而导致的查阅耗费过多时间的问题。

在一个个版本迭代的过程中，我对软件的维护升级有了新得认识。