BAM银行账户管理系统（ATM管理系统）

本系统采用JAVA语言并在eclipse环境下编写测试完成，涉及类的概念，以及面向对象的几大特性（继承，封装，多态，抽象），也有异常处理机制，基本可以满足大多数BAM系统的相关实现，且代码内标注大量注释，读者可以很轻松地理解相关逻辑，大家可以开心参考。

系统简介：

1、JAVA类的面相对象的应用，拥有异常处理机制，不会因为输入错误而导致程序崩溃 2、主要有5个类，即

①Account（账户类）

②SaveAccount（储蓄账户类）：不能透支

③CreditAccount（信用账户类）：可以透支

④Bank（银行类） ⑤ATM（ATM类）

类的具体属性级行为见代码

3、各个类之间的相互关系，涉及继承、封装、多态、抽象，在多态中又涉及重载和重 写，请读者注意相关联系（关注注释）

4、可以实现数据保存功能，数据将保存在文件中（即当你注册了一个账户，下次再登 陆系统时，可以实现与上次最后的操作相衔接）

5、账户号自动生成，比较符合现实

6、主要功能有：

1).开户2).查询账户余额 3).存款 4).取款5).转账（一个账户到另一个账户）等……

7、运行时界面简示

1.初始界面（账户登录）

2.账户登录后界面

注意事项：

1、本系统采用的编程环境是JDK1.7，jer7。所以，运行代码需要保持电脑上所装的JDK为1.7以上版本，如有报错，只需换个高一点的版本即可。注意：第一次装JDK，要配置环境变量（请查阅相关资料，比较简单）

2、本系统代码涉及到包，所以如果报名不一致就会报错，解决方法：修改一下包名即可

3、建议把各个类写在同一个包下面，且每一个类单独写一个java文件，如下图：

4、在运行程序前，需要在项目下面新建一个account.txt（用来保存数据）文件（如上图），并在其中写入至少一个账户信息，（如下图，其中每项代表的意思，请读者参照代码内的注释），否则在初始化的时候会因为找不到账户信息，从而产生异常。

系统源码：

Account类

package com.qx;//包名 /\*\*

\* 账户类:包含两种账户类型-->1.储蓄账户 2.信用账户 \*/

public abstract class Account { //属性

protected long id;

protected String password; protected String name;

protected String personId; protected int accountType;

protected double balance;

//构造方法

public Account(){ super(); }

public Account(long id, String password, String name, String personId,

int accoutType,double balance) { super();

this.id = id;

this.password = password; this.name = name;

this.personId = personId;

this.accountType = accountType; this.balance = balance; }

//getXxx,setXxx方法 public long getId() { return id; }

public void setId(long id) { this.id = id; }

public String getPassword() { return password; }

public void setPassword(String password) { this.password = password; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public String getPersonId() { return personId; }

public void setPersonId(String personId) { this.personId = personId; }

public int getAccountType() { return accountType; }

public void setAccountType(int accountType) { this.accountType = accountType; }

public double getBalance() { return balance; }

public void setBalance(double balance) { this.balance = balance; } /\*\*

\* 存款 \*/

public void deposit(double money){

balance += money; } /\*\*

\* 取款（取款方式由账户类型决定，所以设为抽象方法，相应的Account类应设为抽象类） \*/

public abstract void withdraw(double money); }

SavingAccount类

package com.qx; /\*\*

\* 储蓄账户类 \*/

public class SavingAccount extends Account { }

//构造函数

public SavingAccount() { super(); }

public SavingAccount(long id, String password, String name, String personId,int accountType, double balance) {

super(id, password, name, personId, accountType, balance); }

//对父类的withdraw()实现

public void withdraw(double money){ if(balance < money){ }

System.out.println("对不起，账户余额不足！"); } else {

balance -= money; }

CresitAccount类

package com.qx; /\*\*

\* 信用账户类，增加一个信用额度ceiling属性 \*/

public class CreditAccount extends Account{ private int ceiling;

//构造函数

public CreditAccount(){ super(); }

public CreditAccount(long id, String password, String name, String personId,int accountType, double balance, int ceiling){

super(id, password, name, personId, accountType, balance); this.ceiling = ceiling; } }

//getXxx,setXxx方法

public int getCeiling() { return ceiling; }

public void setCeiling(int ceiling) { this.ceiling = ceiling; }

//实现父类的withdraw()

public void withdraw(double money){ if((balance + ceiling) < money){ }

System.out.println("对不起，已超出您的信用额度！"); } else {

balance -= money; }

Bank类

package com.qx;

import java.io.BufferedReader; import java.io.BufferedWriter; import java.io.File;

import java.io.FileNotFoundException; import java.io.FileReader; import java.io.FileWriter; import java.io.IOException; import java.util.Properties; /\*\*

\* Bank类

\* 编写Bank类,属性:

1.当前所有的账户对象的集合,存放在数组中

2.当前账户数量 方法:

1.用户开户,需要的参数:id,密码,密码确认,姓名,身份证号码,账户类型,返回新创建的Account对象的账号,

提示:用s1.equals(s2) 可以比较s1,s2两个字符串的值是否相等.账户类型是一个整数,为0的时候表示储蓄账户,为1的时候表示信用账户

2.用户登录,参数:id,密码 返回登录账户的账号

3.用户存款,参数:id,存款数额,返回void

4.用户取款,参数:id,取款数额,返回void

5.查询余额,参数:id，返回该账户的余额 double

用户会通过调用Bank对象以上的方法来操作自己的账户,请分析各个方法需要的参数 \*/

public class Bank { private Account[] accounts = new Account[20]; private int number;//账户数目 private int id = 1001;//确定银行账号从1001开始生成，即第一个账户的账号是1001 //构造函数 public Bank(){ accounts=new Account[20];//以后不足时扩容。 number = 0; BufferedReader bufReader = null; Properties props=System.getProperties(); String path=props.getProperty("user.dir"); try { bufReader=new BufferedReader(new FileReader(new File(path,"account.txt"))); String s = bufReader.readLine();

String[] str = s.split(","); if(str[4].equals("0")) { Account savingAcc = new SavingAccount(Long.parseLong(str[0]), str[1].toString(), str[2].toString(), str[3].toString(),Integer.parseInt(str[4]), Double.parseDouble(str[5])); accounts[number] = savingAcc; } else { Account creditAcc = new CreditAccount(Long.parseLong(str[0]), str[1].toString(), str[2].toString(), str[3].toString(),Integer.parseInt(str[4]), Double.parseDouble(str[5]),5000); accounts[number] = creditAcc; } number ++; id++; s = bufReader.readLine(); } } catch (NumberFormatException e) { // TODO Auto-generated catch block e.printStackTrace(); } catch (FileNotFoundException e) { // TODO Auto-generated catch block e.printStackTrace(); } catch (IOException e) { // TODO Auto-generated catch block e.printStackTrace(); }finally{ try { if(bufReader != null) { bufReader.close(); } } catch (IOException e) { // TODO Auto-generated catch block e.printStackTrace(); } } }

//getXxx,setXxx

return accounts; }

public void setAccounts(Account[] accounts) { this.accounts = accounts; }

public int getNumber() { return number; }

public void setNumber(int number) { this.number = number; }

public int getId() { return id; }

public void setId(int id) { this.id = id; } /\*\*

\* 开户 \*/

public Account openAccount(String passwd1, String passwd2, String name, String personId, int type){

//创建一个新账户 Account account = null; //判断两次密码是否一致 if(passwd1.equals(passwd2)){ //若一致，再判断账户类型（根据type的值） if(type == 1){

//可令开始余额为10,信用额度为5000 account = new CreditAccount(id, passwd1, name, personId, type, 10, 5000); } else { account = new SavingAccount(id, passwd1, name, personId, type, 10); }

//将账户存入账户数组accounts[]中 //判断是否超出存储空间 if(number >= accounts.length) { //扩容 Account[] newAccounts = new Account[accounts.length\*2]; //copy原来的相关数据 System.arraycopy(accounts, 0, newAccounts, 0, accounts.length);

//将newAccounts[]赋给accounts[] accounts = newAccounts; accounts[number] = account; } else { accounts[number] = account; }

System.out.println("开户成功！！！账户信息见下"); System.out.println("您的卡号为："+id+"\n"+"您的密码为："+passwd1+"\n"+"您的户名为："+name+"\n" +"您的身份证号为："+personId+"\n"+"您的账户类型为："+type+"\n"); account.accountType = type; number++; id++;

return account;//此时开户成功 } else {

System.out.println("对不起！您两次密码输入不匹配，开户失败！！！"); return null;//此时开户失败

} } /\*\*

\* 保存数据 \*/

public void saveAccountDate(){ BufferedWriter bufWriter=null; try { Properties props=System.getProperties(); String path=props.getProperty("user.dir"); bufWriter=new BufferedWriter(new FileWriter(new File(path,"account.txt"))); for(int i = 0;i < accounts.length;i++) { //若存在账户 if(accounts[i] != null) { //写入账户信息到account.txt bufWriter.write(accounts[i].id+","); bufWriter.write(accounts[i].getPassword()+","); bufWriter.write(accounts[i].getName()+","); bufWriter.write(accounts[i].getPersonId()+","); bufWriter.write(accounts[i].getAccountType()+","); bufWriter.write(Double.toString(accounts[i].getBalance()));

} /\*\*

} else { break; } } bufWriter.flush();//清空缓存中的内容 } catch (IOException e) { // TODO Auto-generated catch block e.printStackTrace(); }finally{ try { if(bufWriter!=null){ bufWriter.close(); } } catch (IOException e) { // TODO Auto-generated catch block e.printStackTrace(); } }

\* 登录验证 \*/

public Account verifyAccount(long id, String password){ Account account = null; for(int i = 0;i < accounts.length;i++) { //若存在账户 if(accounts[i] != null) { //验证id号和password if(id == accounts[i].getId() && password.equals(accounts[i].getPassword())) { account = accounts[i]; break; } } else { break; } } return account;

}

/\*\*

\* 转账验证（方法的重载） \*/

public Account verifyAccount(long id){ Account account = null; for(int i = 0;i < accounts.length;i++) { //若存在账户 if(accounts[i] != null) } /\*\*

{ //验证id号和password if(id == accounts[i].getId()) { account = accounts[i]; break; } } else { break; } }

return account;

\* 转账 \*/

public void transferAccount(Account account1, Account account2, double money){ account1.withdraw(money); account2.deposit(money); } /\*\*

\* 存款 \*/

public void deposit(Account account, double money){ account.deposit(money); } /\*\*

\* 取款 \*/

public void withdraw(Account account, double money){ account.withdraw(money); }

ATM类

package com.qx;

import java.io.BufferedReader; import java.io.BufferedWriter; import java.io.File;

import java.io.FileReader; import java.io.FileWriter; import java.util.Properties; import java.util.Scanner; /\*\*

\* ATM类，提供用户界面操作 \*/

public class ATM {

//属性

private Bank bank; //构造函数 public ATM() { bank = new Bank(); }

//main方法

public static void main(String[] args){

ATM atm = new ATM();//实例化ATM Bank bank = atm.bank; //标号，判断是否退出一级菜单 boolean firstFlag = true; while(firstFlag) { //一级菜单 System.out.println("\*\*\*\*\*\*欢迎使用XXX银行模拟ATM系统，请按如下步骤操作\*\*\*\*\*\*"); System.out.println(" \*1.用已有账户登录"); System.out.println(" \*2.没有账户，开户"); System.out.println(" \*3.退出"); Scanner scanner = new Scanner(System.in); System.out.print("请选择："); try { int choice1 = scanner.nextInt(); switch(choice1) { case 1: scanner = new Scanner(System.in); System.out.print("请输入银行卡号：");

long id = scanner.nextInt();

scanner = new Scanner(System.in);

System.out.print("请输入银行密码："); String password = scanner.next();

Account account = bank.verifyAccount(id, password); if(account != null)

{ //标号，判断是否退出二级菜单 boolean secondFlag = true; while(secondFlag)

{ //二级菜单

System.out.println("\*\*\*\*\*\*欢迎使用XXX银行模拟ATM系统，请按如下步骤操作\*\*\*\*\*\*"); System.out.println(" \*1.查询账户余额"); System.out.println(" \*2.存款"); System.out.println(" \*3.取款"); System.out.println(" \*4.转账"); System.out.println(" \*5.退卡"); scanner = new Scanner(System.in); System.out.print("请选择："); try { int choice2 = scanner.nextInt(); switch(choice2) { case 1: "+account.getBalance());

System.out.println("您账户的当前余额为：

break; case 2: scanner = new Scanner(System.in); System.out.print("请输入您的存款金额："); double money1 = scanner.nextDouble(); bank.deposit(account, money1); break; case 3: scanner = new Scanner(System.in); System.out.print("请输入您的取款金额："); double money2 = scanner.nextDouble(); bank.withdraw(account, money2); break; case 4: scanner = new Scanner(System.in); System.out.print("请输入您要转入账户的卡号："); long id2 = scanner.nextLong(); Account account2 = bank.verifyAccount(id2);

额："); money);

if(account2 != null) { scanner = new Scanner(System.in); System.out.print("请输入您要转入账户的金

double money = scanner.nextLong(); if(money <= account.balance) { bank.transferAccount(account, account2, System.out.println("转账成功！！！"); } else {

System.out.println("抱歉，您账户没有足

够的金额！请查看后重新选择输入！");

}

} else {

System.out.println("抱歉，没有找到您要转入的账户信息！请核对后重新选择输入！"); } break; case 5: secondFlag = false; break; default: System.out.println("没有该选项，请重新选择！"); break; } } catch (Exception e) {

System.out.println("选择输入不合法，请重新选择！"); } } } else { System.out.println("没有此账户，请先开户！"); } break;

case 2://账号id由银行自动提供（从1001递增） scanner = new Scanner(System.in); System.out.print("请输入账户密码：");

}

}

String passwd1 = scanner.next(); scanner = new Scanner(System.in);

System.out.print("请再输入账户密码："); String passwd2 = scanner.next(); scanner = new Scanner(System.in); System.out.print("请输入户主姓名："); String name = scanner.next();

scanner = new Scanner(System.in);

System.out.print("请输入户主身份证号码："); String personId = scanner.next(); scanner = new Scanner(System.in); System.out.print("请输入账户类型（0储蓄，1信用）："); int type = scanner.nextInt(); bank.openAccount(passwd1, passwd2, name, personId, type); break; case 3: bank.saveAccountDate(); firstFlag = false; break; default :

System.out.println("没有该选项，请重新选择！"); break; }

} catch (Exception e) {

System.out.println("选择输入不合法，请重新选择！"); } }

System.out.println("谢谢使用！");

本文档下载自文档之家，www.doczj.com-免费文档分享平台，众多试卷、习题答案、公务员考试、英语学习、法语学习、人力资源管理、电脑基础知识、学习计划、工作计划、工作总结、活动策划、企业管理等文档分类免费下载；乐于分享，共同进步，转载请保留出处:http://www.doczj.com/doc/3fd7b6c026fff705cd170a03.html