**“设计模式”课程**

**大作业**

**软件设计规格说明书**

**目 录**

[1 引言 1](#_Toc28794758)

[1.1 团队成员及分工 1](#_Toc28794759)

[1.2 课题的来源、意义和目标 1](#_Toc28794760)

[1.3 完成的内容与设计结果 1](#_Toc28794761)

[1.3.1 设计内容 1](#_Toc28794762)

[1.3.2 设计结果 1](#_Toc28794763)

[2.1 总体设计 5](#_Toc28794764)

[2.1.1 设计目标 5](#_Toc28794765)

[2.1.2 系统结构 5](#_Toc28794766)

[2.2 实施方案 5](#_Toc28794767)

[2.2.1 开发平台和工具的选择 5](#_Toc28794768)

[2.2.2 方案实施步骤 5](#_Toc28794769)

[2.3 本章小结 6](#_Toc28794770)

[3 关键技术介绍 7](#_Toc28794771)

[3.1 面向对象技术概述 7](#_Toc28794772)

[3.2 面向对象设计原则概述 7](#_Toc28794773)

[3.3 软件设计模式概述 7](#_Toc28794774)

[3.4 软件设计模式介绍 8](#_Toc28794775)

[3.4.1工厂方法模式 8](#_Toc28794776)

[3.4.2 命令模式 8](#_Toc28794777)

[3.4.3 模板方法模式 8](#_Toc28794778)

[3.4.4 桥接模式 8](#_Toc28794779)

[3.4.5 适配器模式 8](#_Toc28794780)

[3.4.6 单例模式 8](#_Toc28794781)

[3.4.7 状态模式 9](#_Toc28794782)

[3.4.8 建造者模式 9](#_Toc28794783)

[3.4.9 简单工厂模式 9](#_Toc28794784)

[3.4.10 原型模式 9](#_Toc28794785)

[3.5 本章小结 9](#_Toc28794786)

[4 系统设计与实现 10](#_Toc28794787)

[4.1 数据库设计 10](#_Toc28794788)

[4.1.1数据库需求分析 10](#_Toc28794789)

[4.1.2 数据库逻辑关系 10](#_Toc28794790)

[4.1.3 数据库结构创建 11](#_Toc28794791)

[4.2 软件设计模式应用 14](#_Toc28794792)

[4.2.1 工厂模式应用 14](#_Toc28794793)

[4.2.2 命令模式应用 14](#_Toc28794794)

[4.2.3 模板方法模式应用 15](#_Toc28794795)

[4.2.4 桥接模式应用 16](#_Toc28794796)

[4.2.5 适配器模式应用 17](#_Toc28794797)

[4.2.6 单例模式应用 18](#_Toc28794798)

[4.2.7 状态模式应用 19](#_Toc28794799)

[4.2.8 建造者模式应用 20](#_Toc28794800)

[4.2.9 简单工厂模式应用 20](#_Toc28794801)

[4.2.10 原型模式应用 21](#_Toc28794802)

[4.3 本章小结 22](#_Toc28794803)

[5 总结与展望 22](#_Toc28794804)

[5.1 总结 22](#_Toc28794805)

[5.2 展望 23](#_Toc28794806)

# 1 引言

## 1.1 团队成员及分工

## 1.2 课题的来源、意义和目标

本组的项目是银行ATM机仿真系统。过去，有许多家银行在客户办理业务时都需要到柜台前服务人员的操作，效率很低。为了解决这样的问题，银行开发出了ATM机来实现24小时自动取款，无需人工服务，同时提供了许多的功能，尽可能满足用户的需求。本项目是模拟ATM机的功能而诞生的一个小程序。

## 1.3 完成的内容与设计结果

### 1.3.1 设计内容

1.功能设计

客户可以在ATM机上实现登录、取款、存款、转账、余额查询、修改密码、打印凭条退出、注册用户、开卡等功能；

2.数据库设计

本项目数据库设计了四个表：银行卡信息表、交易信息表、客户信息表、账户表。

银行卡信息表：卡号、密码、客户ID、开户日期、开户金额、余额、是否挂失；

交易信息表：交易ID、交易金额、交易时间、交易卡号、交易类型、交易描述；

客户信息表：账号、身份证号、姓名、电话号码、地址、客户ID；

账户表：账号、密码；

### 1.3.2 设计结果

1.用例图



2.ER图



3.数据库设计



2 总体设计与实施方案

## 2.1 总体设计

### 2.1.1 设计目标

本项目是模拟实现ATM机的功能，另外添加了客户在ATM机上的注册和开卡等功能。

### 2.1.2 系统结构

本系统有客户模块。客户模块的主要功能是取款、存款、转账、修改密码、余额查询、打印凭条、注册和开卡。该系统功能模块图如下图所示：



系统结构图

## 2.2 实施方案

### 2.2.1 开发平台和工具的选择

本项目使用的开发平台有：MyEclipse、PowerDesigner、Visio 2013、IBM Rational Rose、SQL Server 2008。

### 2.2.2 方案实施步骤

（1）系统需求分析：

项目开始时分析系统需求，即模仿ATM机的功能。将整个系统归为一个模块：客户模块，分析该模块的功能，画出用例图。

（2）系统数据库设计：

本项目将数据库设计为四个表：银行卡信息表、交易信息表、客户信息表、账户表。

（3）模式应用：

根据系统要实现的功能，本项目使用的设计模式有：模板方法模式、命令模式、工厂方法模式、适配器模式、桥接模式、单例模式、原型模式、简单工厂模式、建造者模式、状态模式。

（4）代码编写：

见附件

## 2.3 本章小结

本章主要是确定系统的需求设计、数据库设计以及设计模式运用，通过这些对本系统有一个较为清晰的设计方向和思路。

# 3 关键技术介绍

## 3.1 面向对象技术概述

面向对象技术是一种新的软件技术，其概念来源于程序设计，从20世纪60年代提出面向对象的概念，到现在已发展成为一种比较成熟的编程思想，并且逐步成为目前软件开发领域的主流技术。同时，它不仅局限于程序设计方面，已经成为软件开发领域的一种方法论。它对信息科学、软件工程、人工智能和认知科学等都产生了重大影响，尤其在计算机科学与技术的各个方面影响深远。通过面向对象技术，可以将客观世界直接映射到面向对象解空间，从而为软件设计和系统开发带来革命性影响。

## 3.2 面向对象设计原则概述

对于面向对象软件系统的设计而言，在支持可维护性的同时，提高系统的可复用性是一个至关重要的问题，如何同时提高一个软件系统的可维护性和可复用性是面向对象设计需要解决的核心问题之一。在面向对象设计中，可维护性的复用是以设计原则为基础的。每一个原则都蕴含一些面向对象设计的思想，可以从不同的角度提升一个软件结构的设计水平。

面向对象设计原则为支持可维护性可复用性而诞生，这些原则蕴含在很多设计模式中，它们是从许多设计方案中总结出的指导性原则。面向对象设计原则也是我们用于评价一个设计模式的使用效果的重要指标之一。

## 3.3 软件设计模式概述

软件设计模式，又称设计模式，是一套反复使用、多数人知晓的、经过分类编目的、代码设计经验的总结。它描述了在软件设计过程中的一些不断重复 发生的问题，以及该问题的解决方案。也就是说，它是解决特定问题的一系列套路，是前辈们的代码设计经验的总结，具有一定的普遍性，可以反复使用。其目的是为了提高代码的可重用性、代码的可读性和代码的可靠性。

## 3.4 软件设计模式介绍

### 3.4.1工厂方法模式

工厂方法模式(Factory Method Pattern)又称为工厂模式，也叫虚拟构造器(Virtual Constructor)模式或者多态工厂(Polymorphic Factory)模式，它属于类创建型模式。在工厂方法模式中，工厂父类负责定义创建产品对象的公共接口，而工厂子类则负责生成具体的产品对象，这样做的目的是将产品类的实例化操作延迟到工厂子类中完成，即通过工厂子类来确定究竟应该实例化哪一个具体产品类。

### 3.4.2 命令模式

命令模式(Command Pattern)：将一个请求封装为一个对象，从而使我们可用不同的请求对客户进行参数化；对请求排队或者记录请求日志，以及支持可撤销的操作。命令模式是一种对象行为型模式，其别名为动作(Action)模式或事务(Transaction)模式。

### 3.4.3 模板方法模式

模板方法模式(Template Method Pattern)：定义一个操作中算法的骨架，而将一些步骤延迟到子类中，模板方法使得子类可以不改变一个算法的结构即可重定义该算法的某些特定步骤。模板方法是一种类行为型模式。

### 3.4.4 桥接模式

桥接模式(Bridge Pattern)：将抽象部分与它的实现部分分离，使它们都可以独立地变化。它是一种对象结构型模式，又称为柄体(Handle and Body)模式或接口(Interface)模式。

### 3.4.5 适配器模式

适配器模式(Adapter Pattern) ：将一个接口转换成客户希望的另一个接口，适配器模式使接口不兼容的那些类可以一起工作，其别名为包装器(Wrapper)。适配器模式既可以作为类结构型模式，也可以作为对象结构型模式。

### 3.4.6 单例模式

单例模式确保某一个类只有一个实例，而且自行实例化并向整个系统提供这个实例，这个类称为单例类，它提供全局访问的方法。

### 3.4.7 状态模式

状态模式(State Pattern) ：允许一个对象在其内部状态改变时改变它的行为，对象看起来似乎修改了它的类。其别名为状态对象(Objects for States)，状态模式是一种对象行为型模式。

### 3.4.8 建造者模式

建造者模式(Builder Pattern)：将一个复杂对象的构建与它的表示分离，使得同样的构建过程可以创建不同的表示。

### 3.4.9 简单工厂模式

简单工厂模式(Simple Factory Pattern)：又称为静态工厂方法(Static Factory Method)模式，它属于类创建型模式。在简单工厂模式中，可以根据参数的不同返回不同类的实例。简单工厂模式专门定义一个类来负责创建其他类的实例，被创建的实例通常都具有共同的父类。

### 3.4.10 原型模式

原型模式(Prototype Pattern)：原型模式是一种对象创建型模式，用原型实例指定创建对象的种类，并且通过复制这些原型创建新的对象。原型模式允许一个对象再创建另外一个可定制的对象，无须知道任何创建的细节。

## 3.5 本章小结

本章主要是讲述一些基础概念，让我们了解到面向对象技术、面向对象设计原则，软件设计模式的概念和具体模式的概念，对后面将软件设计模式运用到系统中有很大帮助。

# 4 系统设计与实现

## 4.1 数据库设计

### 4.1.1数据库需求分析

客户在登录ATM机时需要的账号密码信息保存在账户表中；已注册客户的信息保存在客户表中；卡号信息保存在银行卡信息表、银行卡的交易信息保存在交易信息表中。

账户表（账号、密码）

客户信息表（客户ID、账号、身份证号、姓名、电话号码、地址）

银行卡信息表（卡号、密码、客户ID、开户日期、开户金额、余额、是否挂失）

交易信息表（交易ID、交易金额、交易时间、交易卡号、交易类型、交易描述）

### 4.1.2 数据库逻辑关系

根据上面的数据项及数据结构，设计的实体有：银行卡信息、交易信息、客户信息、账户。

根据系统设计的功能，按照不同的实体分类可以分为以下几个实体对象：他们架构起了整个银行ATM机系统。E-R图如下图所示：



E-R图

### 4.1.3 数据库结构创建

本系统使用的是SQL Server数据库。根据实体及实体关系在数据库中设计4个表，如下表所示：

整体数据库列表

|  |  |  |
| --- | --- | --- |
| **序 号** | **数据库表** | **数据库表存储内容** |
| 1 | cardInfo | 银行卡信息表 |
| 2 | transInfo | 交易信息表 |
| 3 | userInfo | 客户信息表 |
| 4 | account | 账号表 |

此系统有客户一个对象，根据他的功能，设计数据项和数据结构。数据字典如下表所示：

银行卡信息表，包含的数据项：卡号、密码、客户ID、开户日期、开户金额、余额、余额。表结构如下表所示：

银行卡信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 表名 | cardInfo | | |
| **字段名** | **数据类型（精度范围）** | **字段含义** | **其他** |
| cardID | char(19) | 卡号 | PK |
| cardPassword | char(6) | 密码 | null |
| customerID | char(8) | 客户ID | null |
| openDate | datetime | 开户日期 | null |
| openMoney | money | 开户金额 | null |
| balance | money | 余额 | null |
| IsReportLoss | int | 是否挂失 | null |
| 补充说明 | 银行卡信息表：此表为银行卡信息的表，表中cardID为表的主键；IsReportLoss为0表示该银行卡挂失了，不能继续使用，IsReportLoss为1则代表没有挂失，可以使用, customerID是外键。 | | |

交易信息表，包含的数据项：交易ID、交易卡号、交易金额、交易时间、交易类型、交易描述。表结构如下表所示：

交易信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 表名 | transInfo | | |
| **字段名** | **数据类型（精度范围）** | **字段含义** | **其他** |
| transID | char(6) | 交易ID | PK |
| cardID | char(19) | 交易卡号 | FK |
| transMoney | money | 交易金额、 | null |
| transDate | datetime | 交易时间 | null |
| transType | char(4) | 交易类型 | null |
| remark | text | 交易描述 | null |
| 补充说明 | 交易信息表：展现每个交易记录的明细，其中cardID是该表的外键。 | | |

客户信息表，包含的数据项：客户ID、姓名、身份证号、电话号、地址、账号。表结构如下表所示：

客户信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 表名 | userInfo | | |
| **字段名** | **数据类型（精度范围）** | **字段含义** | **其他** |
| customerID | char(8) | 客户ID | PK |
| customerName | Varchar(10) | 姓名 | null |
| PID | Char(18) | 身份证号 | null |
| telephone | char(11) | 电话号 | null |
| adress | Varchar(50) | 地址 | null |
| accountID | Varchar(10) | 账号 | FK |
| 补充说明 | 客户信息表：展现每个银行客户的信息，其中accountID是该表的外键。 | | |

账号表，包含的数据项：账号、密码。表结构如下表所示：

账号表

|  |  |  |  |
| --- | --- | --- | --- |
| 表名 | account | | |
| **字段名** | **数据类型（精度范围）** | **字段含义** | **其他** |
| accountID | varchar(10) | 账号 | PK |
| accountPassword | char(6) | 密码 | null |
| 补充说明 | 账号表：展现每个账号的信息。 | | |

## 4.2 软件设计模式应用

### 4.2.1 工厂模式应用

Jdbc的连接使用的是工厂模式



代码见附件中的qpf.FactoryMethod包。



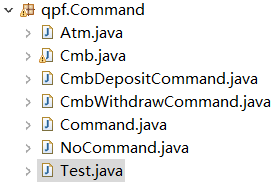
### 4.2.2 命令模式应用

命令模式实现ATM机的存取款功能

命令模式适用于“请求——响应”模式的功能，将用户的请求封装成对象（命令），用户需要执行什么样的操作，就调用什么样的命令，而无需知道命令的执行逻辑是什么。一个有存取款功能的ATM机，它可以向银行的卡里存款，也可以从银行卡里取款。我们不关心银行怎么实现存取款，只要点击ATM上的按钮就行了。银行的存款和取款分别为两个具体的命令实现类（ConcreteCommand）；ATM机充当调用者（Invoker），负责调用银行存款或取款的命令；银行的系统为接收者（Reveiver），处理存款和取款的业务逻辑；使用ATM机的人就是客户端。



代码见附件中的qpf.Command包。

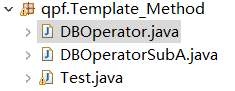


### 4.2.3 模板方法模式应用

命令模式实现数据库连接



代码见附件中的qpf.Template\_Method包。



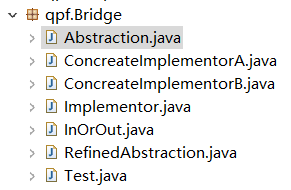
### 4.2.4 桥接模式应用

桥接模式实现转账功能，包括银行内转账和银行间转账。

桥接模式（Bridge)是一种结构型设计模式。Bridge模式基于类的最小设计原则，通过使用封装、聚合及继承等行为让不同的类承担不同的职责。它的主要特点是把抽象(Abstraction)与行为实现(Implementation)分离开来，从而可以保持各部分的独立性以及应对他们的功能扩展。



代码见附件中的qpf.Bridge包。



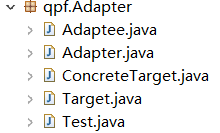
### 4.2.5 适配器模式应用

适配器模式实现修改密码功能。

适配器模式是将一个接口转换成客户希望的另一个接口，使接口不兼容的那些类可以一起工作，其别名为包装器(Wrapper)。适配器模式既可以作为类结构型模式，也可以作为对象结构型模式。在适配器模式中，我们通过增加一个新的适配器类来解决接口不兼容的问题，使得原本没有任何关系的类可以协同工作。



代码见附件中的qpf.Adapter包。

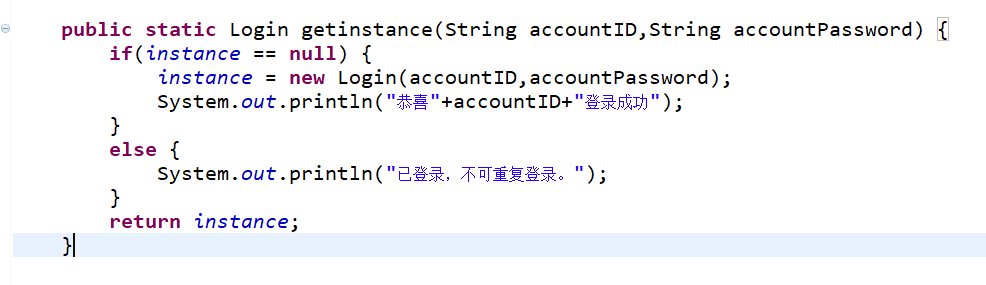


### 4.2.6 单例模式应用

将单例模式运用于用户登录，可以防止用户重复登录。



部分代码如下，具体代码见附件：

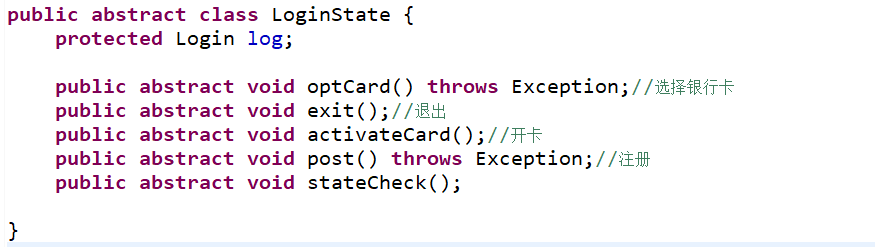


### 4.2.7 状态模式应用

将状态模式应用于登录功能，可以体现登录前后状态下的权限是不同的，未登录能注册和开卡，不能退出登录和选择卡号，而登录后才能选卡和退出，不能注册和开卡。



部分代码如下，具体代码见附件：

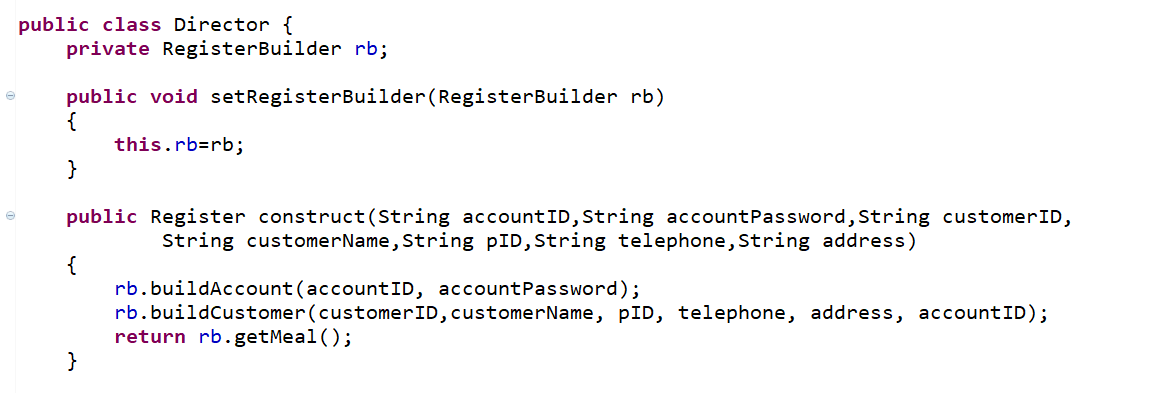


### 4.2.8 建造者模式应用

将建造者模式应用于注册功能。



部分代码如下，具体代码见附件：

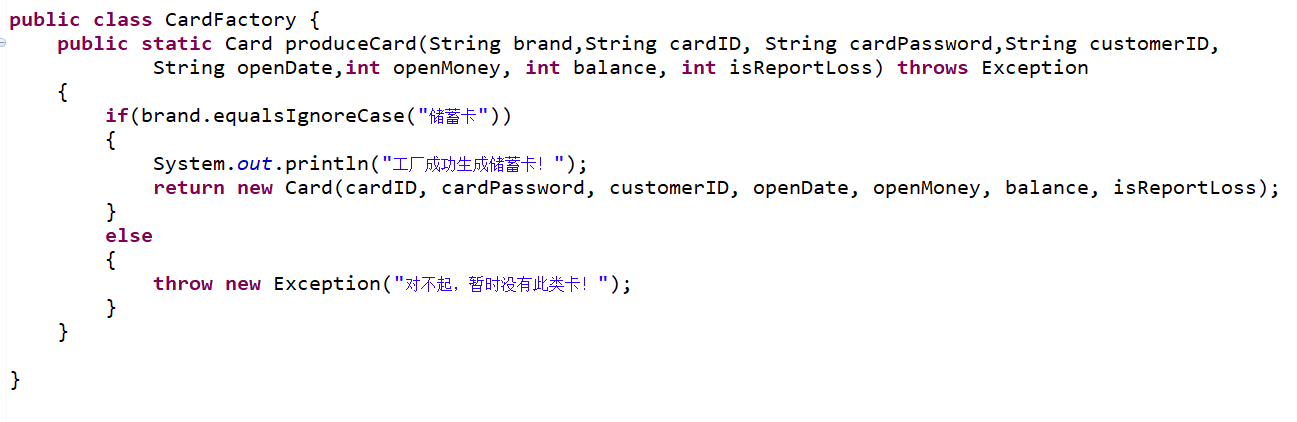


### 4.2.9 简单工厂模式应用

将简单工厂模式运用于生成银行卡类中，目前只有储蓄卡，后面可以加信用卡等各类卡。



部分代码如下，具体代码见附件：

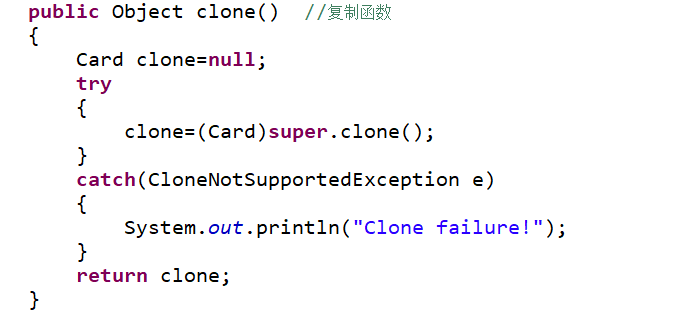


### 4.2.10 原型模式应用

将原型模式应用于银行卡类创建中，一个用户可能拥有很多张银行卡，这样可以方便创建。



部分代码如下，具体代码见附件：



## 4.3 本章小结

本章主要介绍系统的数据库设计和设计模式的应用。数据库设计从需求分析、逻辑关系、结构创建展示整个数据库从想法到实现的过程。设计模式应用主要讲述各个设计模式在系统中是如何实现的，类与类之间的关系是怎么样的，具体代码是如何实现的。

# 5 总结与展望

## 5.1 总结

本课程学习了设计模式的应用，设计模式(Design Pattern)是一套被反复使用、多数人知晓的、经过分类编目的、代码设计经验的总结，使用设计模式是为了可重用代码、让代码更容易被他人理解、保证代码可靠性。在程序编写中考虑使用设计模式可以使得设计方案更加灵活，且易于修改。设计模式有助于我们初学者更深入地理解面向对象思想，一方面可以帮助我们更加方便地阅读和学习现有类库与其他系统中的源代码，另一方面还可以提高软件的设计水平和代码质量。

## 5.2 展望

本项目是我们第一次使用设计模式编写的一个程序，通过对设计模式的应用，对面向对象思想有了更深入的理解。本组的ATM机仿真系统目前只实现了客户模块的功能，虽与实际生活中的ATM机有所不同，但基本的功能也都能实现。本系统暂未实现管理员对ATM机的管理，这将是我们后续仍需在改进的地方。