CHAPTER

第3章 ATM存取款管理系统设计与实现

本章介绍一个简单的ATM存取款管理系统的设计与实现，应用Java编程语言和面向对象开发方法完成，目的是让大家熟悉Java面向对象基本特点——继承、封装和多态以及异常处理、输入输出——在编程中的实现，为以后进行较复杂的项目开发打下良好基础。该系统只关注如何用面向对象的思维和方法将问题描述清楚，不要求实现GUI，只考虑系统逻辑实现。

3.1 项目需求分析

为了使程序实现更加简单，更好地理解面向对象编程思想，本系统模拟一般的ATM机功能并做了简化，只针对其主要业务加以定义和实现。

ATM系统的基本功能包括用户身份验证、存款、取款、查询余额、修改密码、查看用户信息。具体需求如下：

(1) 用户身份验证。进入ATM管理系统之前，输入用户账号和密码，模拟刷卡输入密码的过程，以验证用户身份是否合法。

(2) 存款。系统主要功能之一，需要提示用户存款额和账户余额数。

(3) 取款。系统主要功能之一，要验证取款额不能大于账户余额，并提示用户取款额和账户余额数。

(4) 查询余额。查询当前账户的账面余额。

(5) 修改密码。提示输入用户输入原密码，验证通过后两次输入新密码，如果两次输入密码不一致，提示密码修改失败；原密码输错三次则提示本次不能再修改密码了。

(6) 查看用户信息。显示当前账户的账号、用户名、年龄、账户余额等信息。

3.2 面向对象的分析与设计

3.2.1 实体类分析与设计

面向对象设计把握一个重要的原则：谁拥有数据，谁就对外提供操作这些数据的方法。

ATM 管理系统中，有几种实体数据是必须要表达的，如账户数据、用户数据、银行卡数据。因此，可以设计以下三个类：

（1）银行卡类 Card。封装银行卡号 cardNumbr 和对银行卡号进行操作的 get、set 方法。

（2）用户类 User。封装用户姓名、性别、年龄和所持有的银行卡等信息，并通过 get、set 方法对这些信息进行操作。

（3）账户类 Account。封装账户密码、账户余额和账户所关联的用户等信息，并在这个类中提供存钱、取钱、查询余额、修改密码等业务方法。

各个类的图示如图 3.1 所示。

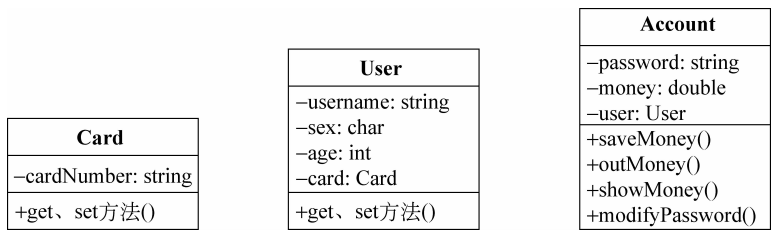

图 3.1　ATM 管理系统实体类图

3.2.2　工具类分析与设计

而系统的处理过程中，有很多具有共性的输入输出功能，可以将其定义成静态方法，全部封装在一个工具类 Tools 里，方便以后调用。类图如图 3.2 所示。

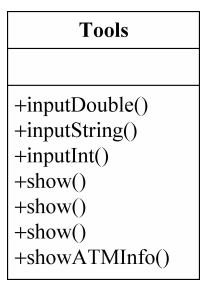

图 3.2　ATM 管理系统工具类图

其中三个 input 方法分别通过键盘读取用户输入的 double 型、String 型和 int 型数据，在 ATM 系统中多处需要从键盘输入数据，如输入账号、输入密码、输入存取款钱数……此时，调用这些静态方法将大大提高代码的复用性，而且使程序更简洁，结构性更好。

而三个重载的 show 方法可以根据参数的不同分别输出银行卡信息、用户信息和银行账户信息，即 show(Card)输出银行卡信息，show(User)输出用户信息，而 show(Account) 输出账户信息。这是 Java 面向对象多态性的一个很好应用。

最后，可以将显示 ATM 主界面信息的功能定义在 showATMInfo()方法中从而简化后面主类的设计，使主类的功能更单一。

3.2.3　主类分析与设计

最后，设计主类 MainClass 实现系统业务流程的控制，只在主类中定义主方法实现系统的流程控制。

ATM 管理系统流程图如图 3.3 所示。

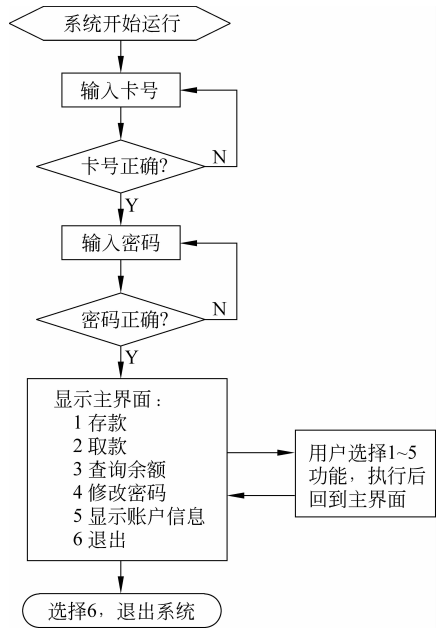

图 3.3　ATM 管理系统流程图

3.3　系统实现与测试

3.3.1　项目环境准备

我们在 MyEclipse 8.6 开发环境下完成这个项目(其他任何环境下也都可以顺利完成),下面以 MyEclipse 8.6 开发环境为例逐步完成项目的构建。

启动 MyEclipse,主体界面如图 3.4 所示。

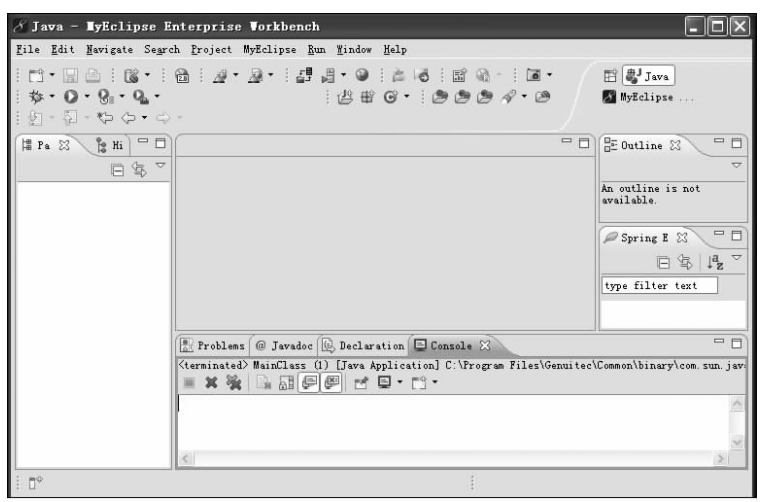

图 3.4　MyEclipse 主界面

在菜单中选择 File→New→Java Project 选项，如图 3.5 所示。

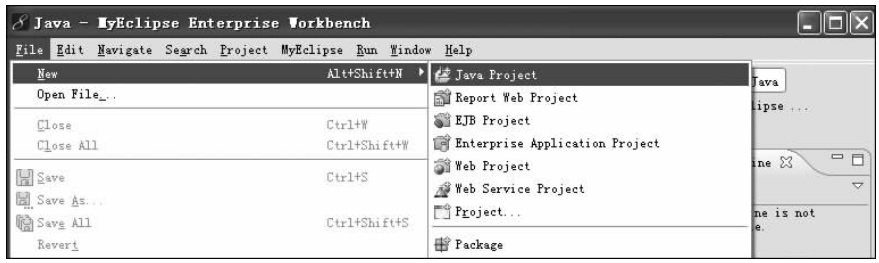

图 3.5　新建 Java 项目

输入项目名称，例如，ATMManagement，单击 Finish 按钮，如图 3.6 所示。

图 3.6　输入项目信息

在菜单中选择 File→New→Package 选项或右击项目名称下 src→New→Package（如图 3.7 所示）新建包。

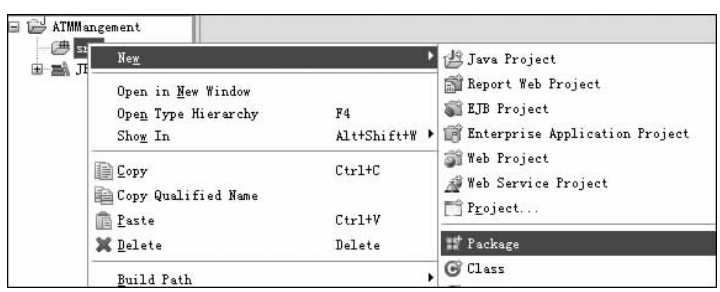

图 3.7　新建包

输入包名 myproject. atm 并单击 Finish 按钮，如图 3. 8 所示。

图 3. 8　输入包名

在 Java 中，不推荐把类放在无名包中，所以项目开发时，一般都会按照功能相似或性质相同的规律建立几个包，把类放到相应包下。

因为我们的项目比较简单，只定义一个包 myproject. atm，将所有类定义在这个包下。

在菜单中选择 File→New→Class 新建类，如图 3. 9 所示。

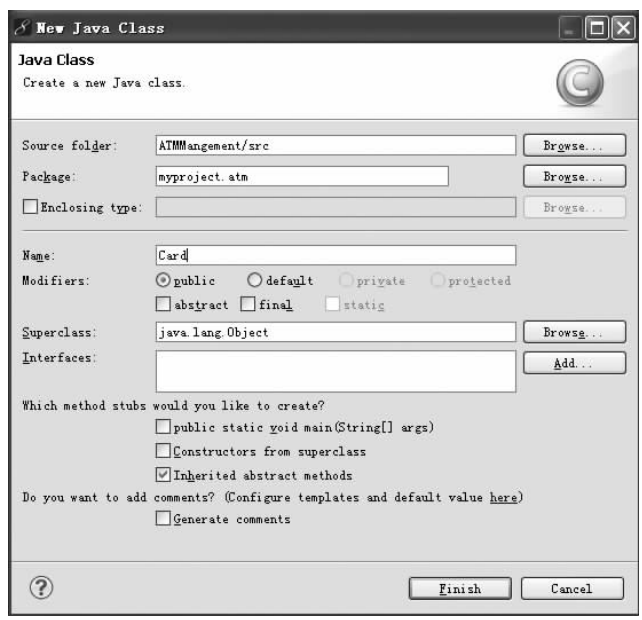

图 3. 9　新建类

单击 Finish 按钮，打开 Card. java 文件编辑页面(如图 3. 10 所示)，输入类文件内容。

注意：在 MyEclipse 中，可以借助软件自带的功能自动生成构造方法和 get、set 方法，大大简化代码的编写，提高编码速度，如图 3. 11～图 3. 13 所示。

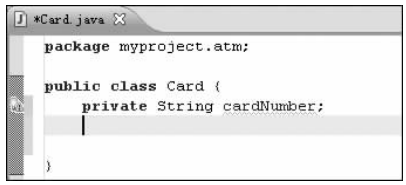

图 3.10　编写类文件内容

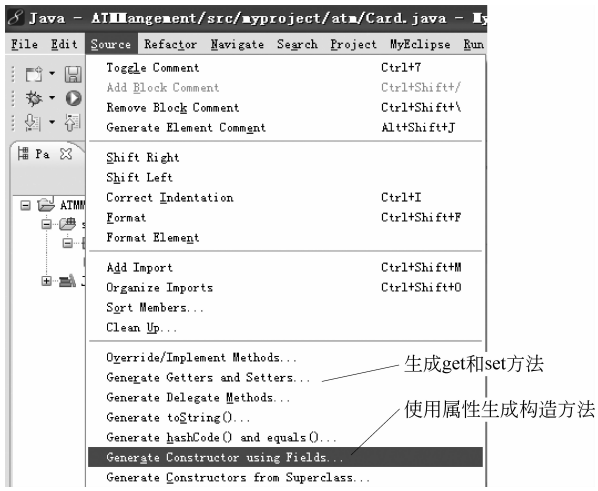

图 3.11　自动生成代码

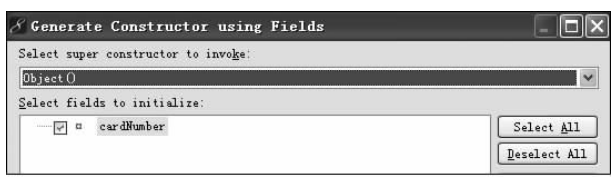

图 3.12　自动生成构造函数

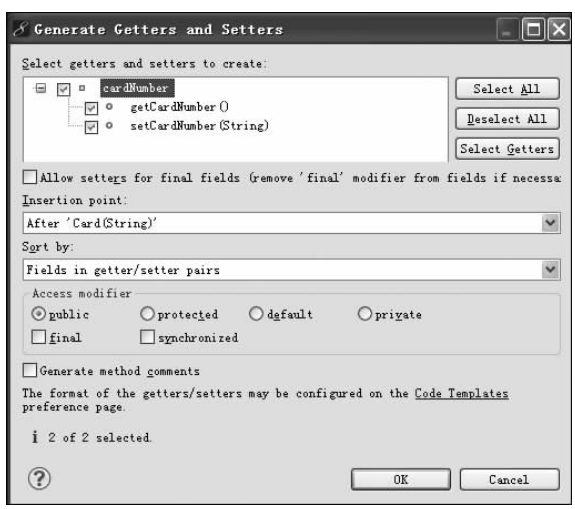

图 3.13　自动生成 get、set 方法

在 Card 类中编写主方法，测试一下，如图 3.14 所示。

*Card.java

```
package myproject.atm;
public class Card {
    private String cardNumber;
    public Card() {
        super();
    }
    public Card(String cardNumber) {
        super();
        this.cardNumber = cardNumber;
    }
    public String getCardNumber() {
        return cardNumber;
    }
    public void setCardNumber(String cardNumber) {
        this.cardNumber = cardNumber;
    }
    //编写主方法测试一下
    public static void main(String args[]){
        Card card=new Card("62251000");
        System.out.println("卡号是: "+card.getCardNumber());
        card.setCardNumber("62252000");
        System.out.println("改变后卡号是: "+card.getCardNumber());
    }
```

图 3.14　编写主方法测试

执行 Run→Run 命令或者按下 Ctrl＋F11 键，运行文件，在控制台 Console 窗口看到的运行结果如图 3.15 所示。

Problems | Javadoc | Declaration | Console

```
<terminated> Card [Java Application] C:\Program Files\Genuitec\Common\b
卡号是: 62251000
改变后卡号是: 62252000
```

图 3.15　运行结果

3.3.2　项目类定义与实现

在项目环境下依次实现三个实体类 Card、User 和 Account，工具类 Tods 以及主类 MainClass 的定义，参考代码如下。

1. 银行卡类源程序 Card.java

略。参见图 3.14。

2. 用户信息类源程序 User.java

```
package myproject.atm;

public class User {
    //属性
    private String username=null;
    private char sex='男';
    private int age=0;
    private Card card=new Card();
```

```
    //构造方法
    public User(String username, char sex, int age, Card card){
        //略,请自动生成
    }
    public User(){
        super();
    }
    //get、set 成员方法略,请自动生成
}
```

3. 账户类源程序 Account.java

```
package myproject.atm;
public class Account {
    //封装私有属性
    private String password=null;
    private double money=0.0;
    private User user=new User();

    //构造方法略,请自动生成
    //get、set 成员方法略,请自动生成

    //存钱方法
    public void saveMoney(){
        System.out.print("请输入你的存款金额:");
        double money=0.0;
        money=Tools.inputDouble();              //从键盘上读取一个实数
        this.money+=money;
        System.out.println("您本次存入:"+money+"元,账户余额是:"+this.money+"元");
    }
    //取钱方法
    public void outMoney(){
        double money=0.0;
        System.out.print("请输入取款金额:");
        money=Tools.inputDouble();              //从键盘上读取一个实数
        if(money>this.money){
            System.out.print("余额不足!");
        }else{
            this.money-=money;
            System.out.println("您本次取款"+money+"元,账户余额:"+this.money);
        }
    }
    //显示账户余额
    public void showMoney(){
        System.out.println("您的账户余额是:"+"  "+money+"  元");
```

```
    }
    //修改密码
    public void modifyPassword(){
        String s1=null, s2=null;
        int num=3;
        System.out.print("请输入原密码:");
        while(num-->0){
            if(Tools.inputString().equals(this.password)){
                System.out.print("请输入新的密码:");
                s1=Tools.inputString();
                System.out.print("请再输入一次:");
                s2=Tools.inputString();
                if(s1.equals(s2)){
                    password=s1;
                    System.out.println("修改密码成功 ~");
                }else{
                    System.out.println("两次输入的密码不一致,密码修改失败!");
                }
                return;
            }else{
                if(num==0){
                    System.out.println("您已输错 3 次密码,本次不再允许修改~");
                    break;
                }
                System.out.print("密码不对,请重新输入:");
            }
        }
    }
}
```

4. 工具类源程序 Tools.java

```
package myproject.atm;
import java.util.Scanner;
public class Tools {
    //显示银行卡信息
    public static void show(Card card){
        System.out.println("卡号: "+card.getCardNumber());
    }
    //显示用户信息
    public static void show(User user){
        show(user.getCard());
        System.out.println("姓名: "+user.getUsername());
        System.out.println("性别: "+user.getSex());
        System.out.println("年龄: "+user.getAge());
```

```
}
//显示账户信息
public static void show(Account acount){
    show(acount.getUser());
    System.out.println("密码:"+acount.getPassword());
    System.out.println("余额: "+acount.getMoney());
}
//显示提示用户操作的主界面信息
public static void showATMInfo(){
    System.out.println("ATM存取款管理系统,请选择你的操作:");
    System.out.println("--------------------------------");
    System.out.println("  1  存款");
    System.out.println("  2  取款");
    System.out.println("  3  查询余额");
    System.out.println("  4  修改密码 ");
    System.out.println("  5  显示账户信息");
    System.out.println("  6  退出      ");
    System.out.println("--------------------------------");
}
//从键盘上读取一个整数
public static int inputInt(){
    int i=0;
    Scanner in=null;
    try {
        in=new Scanner(System.in);
        i=in.nextInt();
    } catch(Exception e){
        System.out.println("Exception when read String from the keyboard.");
        inputString();
    }
    return i;
}
//从键盘上读取一个实数
public static double inputDouble(){
    double d=0;
    Scanner in=null;
    try {
        in=new Scanner(System.in);
        d=in.nextDouble();
    } catch(Exception e){
        System.out.println("Exception when read String from the keyboard.");
        inputString();
    }
    return d;
```

```
    }
    //从键盘上读取一个字符串
    public static String inputString(){
        String s=null;
        Scanner in=null;
        try {
            in=new Scanner(System.in);
            s=in.nextLine();
        } catch(Exception e){
            System.out.println("Exception when read String from the keyboard.");
            inputString();
        }
        return s.trim();
    }
}
```

5. 主类 MainClass.java 及项目结构

定义主类文件 MainClass.java，完成整个项目，如图 3.16 所示。

图 3.16　项目结构图

参考代码主要部分如下(主方法部分)：

```
    //为卡号为 0101 的用户开户
    Card card=new Card("0101");
    User user=new User("张三",'男',20,card);
    Account acount=new Account("1234",1000,user);

    //用户登录
    System.out.print("请输入卡号:");
    String cardNumber=Tools.inputString();
    //验证卡号是否正确
while(!cardNumber.equals(acount.getUser().getCard().getCardNumber()))
    {
        System.out.print("卡号错误,请重新输入:");
```

```
    cardNumber=Tools.inputString();
}
System.out.print("请输入密码:");
String password=Tools.inputString();

//验证密码是否正确
while(!password.equals(acount.getPassword())){
    System.out.print("密码错误,请重新输入:");
    password=Tools.inputString();
}
while(true){
        Tools.showATMInfo();
        int i=Tools.inputInt();
        switch(i){
          case 1:
            acount.saveMoney();    break;
          case 2:
            acount.outMoney();    break;
          case 3:
            acount.showMoney();    break;
          case 4:
             acount.modifyPassword(); break;
          case 5:
            Tools.show(acount);break;
          case 6:
            System.out.println("系统已经退出,再见!");return;
          default:
              Tools.showATMInfo();
        }
}
```

3.3.3 项目测试与改进

1. 项目运行与测试

(1) 完成项目文件的编写后,运行项目(Ctrl+F11 键),运行结果如图 3.17 所示。

(2) 分别输入正确的卡号和不正确的卡号测试程序,应该分别进入输入密码流程(见图 3.18)和提示“卡号错误,请重新输入:”流程(见图 3.19)。

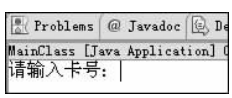

图 3.17 输入卡号

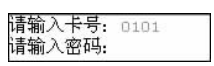

图 3.18 卡号输入正确

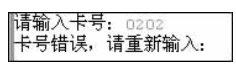

图 3.19 卡号输入错误

(3) 在输入密码阶段,分别输入正确的密码和不正确的密码,应该分别进入 ATM 系统主界面(见图 3.20)和提示“密码错误,请重新输入:”流程(见图 3.21)。

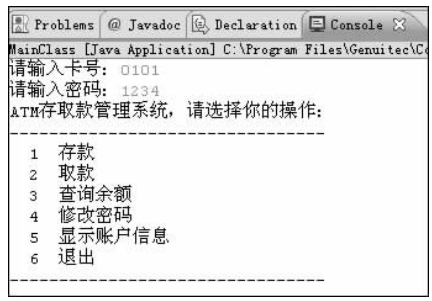

图 3.20　ATM 系统主界面

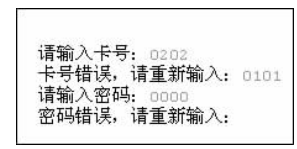

图 3.21　密码输入错误

依次选择 1～5 功能并进行测试，总结项目运行结果和设计目标是否一致，并思考项目中存在哪些问题。

2. 项目改进

上述的 ATM 管理系统程序基本实现了项目需求，在此基础上还可以考虑从以下几个方面进行改进。

(1) 为程序增加金额数据输入合法性检查功能：在用户输入存取款金额时，如果输入的数据不是数值型数据，应提示用户数据输入错误，并提示应该输入数值型数据。

(2) 改进程序界面：可以将程序界面改成 GUI 方式的，用户体验会更友好。

3.4　课后训练项目：银行业务调度系统

1. 需求描述

应用 Java 编程语言和面向对象开发方法，编程模拟实现银行业务调度系统逻辑，具体需求如下。

- 银行内有 6 个业务窗口，1～4 号窗口为普通窗口，5 号窗口为快速窗口，6 号窗口为 VIP 窗口。
- 有三种对应类型的客户：VIP 客户、普通客户、快速客户(办理如交水电费、电话费之类业务的客户)。
- 异步随机生成各种类型的客户，生成各类型用户的概率比例为：

  VIP 客户：普通客户：快速客户＝1：6：3

- 客户办理业务所需时间有最大值和最小值，在该范围内随机设定每个 VIP 客户以及普通客户办理业务所需的时间，快速客户办理业务所需时间为最小值(提示：办理业务的过程可通过线程 Sleep 的方式模拟)。
- 各类型客户在其对应窗口按顺序依次办理业务，当 VIP(6 号)窗口和快速业务(5 号)窗口没有客户等待办理业务的时候，这两个窗口可以处理普通客户的业务，而一旦有对应的客户等待办理业务的时候，则优先处理对应客户的业务。
- 随机生成客户时间间隔以及业务办理时间最大值和最小值自定，可以设置。
- 不要求实现 GUI，只考虑系统逻辑实现，可通过 Log 方式展现程序运行结果。

2. 系统分析提示

有三种对应类型的客户：VIP 客户、普通客户、快速客户，异步随机生成各种类型的客户，各类型客户在其对应窗口按顺序依次办理业务。

每一个客户其实就是由银行的一个取号机器产生号码的方式来表示的。所以，可以设计一个号码管理器对象，让这个对象不断地产生号码，就等于随机生成了客户。

由于有三类客户，每类客户的号码编排都是完全独立的，所以，一共要产生三个号码管理器对象，各自管理一类用户的排队号码。这三个号码管理器对象统一由一个号码机器进行管理，这个号码机器在整个系统中始终只能有一个，所以，它要被设计成单例。

各类型客户在其对应窗口按顺序依次办理业务，准确地说，应该是窗口依次叫号，服务窗口每次找号码管理器获取当前要被服务的号码。

根据上面的分析，可以设计以下类实现系统业务逻辑：

1）NumberManager 类

定义一个用于存储上一个客户号码的成员变量和用于存储所有等待服务的客户号码的队列集合。

定义一个产生新号码的方法和获取马上要为之服务的号码的方法，这两个方法被不同的线程操作了相同的数据，所以，要进行同步。

2）NumberMachine 类

定义三个成员变量分别指向三个 NumberManager 对象，分别表示普通、快速和 VIP 客户的号码管理器，定义三个对应的方法来返回这三个 NumberManager 对象。

将 NumberMachine 类设计成单例。

3）CustomerType 枚举类

系统中有三种类型的客户，所以用定义一个枚举类，其中定义三个成员分别表示三种类型的客户。

重写 toString 方法，返回类型的中文名称。这是在后面编码时重构出来的，刚开始不用考虑。

4）ServiceWindow 类

定义一个 start 方法，内部启动一个线程，根据服务窗口的类别分别循环调用三个不同的方法。

定义三个方法分别对三种客户进行服务，为了观察运行效果，应详细打印出其中的细节信息。

5）MainClass 类

用 for 循环创建出四个普通窗口，再创建出一个快速窗口和一个 VIP 窗口。

接着再创建三个定时器，分别定时去创建新的普通客户号码、新的快速客户号码、新的 VIP 客户号码。

6）Constants 类

定义三个常量：MAX_SERVICE_TIME、MIN_SERVICE_TIME、COMMON_CUSTOMER _INTERVAL_TIME。

CHAPTER

第 4 章 Java 在线考试系统设计与实现

本章将应用 J2SE 和 MySQL 数据库完成一个 Java 在线考试系统项目的开发。之所以选择考试系统，是因为大家比较熟悉该系统功能，能很容易地理解业务流程，更多地关注设计方法和技术实现，熟悉 Java 数据访问技术和“分层”设计思想，从而掌握一般软件项目的设计思路。

4.1 系统分析

一个软件项目的开发是从分析用户需求开始的。本节将对考试系统软件项目进行简单的需求分析、业务流程分析和数据分析，从而明确项目需求和业务流程，并为数据库设计做好准备。

4.1.1 需求分析

随着计算机和网络的普及应用，传统的卷面考试方式已经不能适应现代考试的需要，也不利于学生随时把握知识掌握情况。建立一个便利的在线考试平台，提供学生在线考试和评阅、教师在线试题维护和管理，以及教师、学生个人信息的管理，实现考试、评分、管理一体化与信息化，可以提高考生学习效率，降低老师评卷工作量，减少管理成本。

具体需求如下：

1. 按用户类别存储个人信息

要求对考生、教师和管理员用户分别存储其个人信息：记录考生用户的考生号、姓名、性别、身份证号、密码、系别和电话号码；记录教师用户的教工号、姓名、性别、身份证号、密码、职称和电话号码；记录管理员用户的管理员号、姓名、性别、身份证号、密码和电话号码。

学生和教师用户可以有多个，其信息由管理员用户创建、删除和修改；而管理员用户在系统运行后不能增加、删除和修改。

2. 按试题类型存储试题信息

要求分别存储填空题、选择题及其答案信息：题号顺序从小到大，不允许空号，从 1

开始；填空题记录题干和答案；选择题记录题干、四个选项和答案序号(A、B、C、D)。

试题可以在系统运行过程中由教师用户维护，如增加、删除和修改；学生用户和管理员用户不能修改试题。

3. 按用户类别登录进入系统

从同一界面登录到系统，进行用户身份的验证并根据用户身份进入不同的操作界面：学生用户进入考试界面，可以进行答题和修改个人密码；教师用户进入教师管理界面，可以进行试题维护和修改个人密码；管理员用户进入系统管理界面，可以进行学生和教师用户信息的增删改和修改个人密码。

4. 试题随机生成并自动判分

由系统在试题库中随机抽取题目组成由填空题和选择题构成的试卷，考试结束由系统自动判分并反馈给考生。

5. 实现考试计时

考生进入考试系统则开始计时并显示给考生，允许考生在允许的时间范围内主动交卷或在达到考试时间后由系统强行收卷；考试时间剩余 5 分钟的时候给学生提示。

6. 统一而有效的数据存储

实现系统数据的永久存储，试题数据、用户信息等全部统一存储在安全的数据库服务器系统中，所有用户访问的是相同的数据；存取数据迅速有效、代价低。

4.1.2 业务流程分析

为实现系统要求的功能，经分析应具备的相关业务主要包括学生在线考试及评分、教师在线维护试题、管理员管理用户。业务流程图如图 4.1 所示。

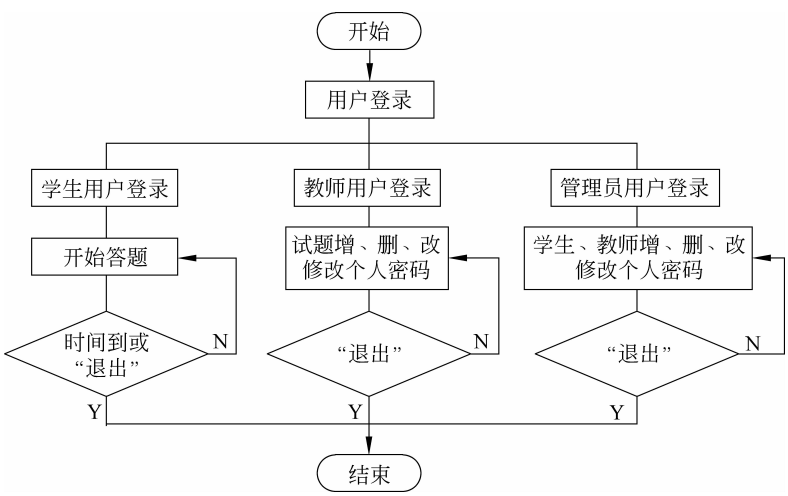

图 4.1 考试系统业务流程图

4.1.3 数据分析

由于考试系统项目比较简单，经过前面的分析，涉及的数据主要包括两大类别：用户信息和试题信息。这两类数据在系统之间没有复杂的关联，只需按照需求设计相应的数据实体即可。

4.2 系统设计

4.2.1 系统设计思路

本系统采用了分层化模块设计，以系统公用数据库为操作对象，将整个系统划分为考生管理模块、教师管理模块、系统管理模块。

由于本项目规模小，数据量也不大，业务逻辑较简单，设计用 MySQL 数据库实现数据的永久存储；将不同的用户、试题等数据以及对这些数据的操作封装到不同的类中（实体类）；将对数据库和数据表的操作也封装起来（数据访问类）；再定义用户操作界面（视图类）调用这些方法实现相应模块的功能。

将系统设计为分层实现，可以使系统实现起来更加容易；另外，也更容易改进系统，例如，可以很容易将这个系统改造为 B/S 架构的 Web 应用程序，只需要将用户界面部分改用某种网页形式（如 JSP）就可以了，而且分层结构也是复杂软件项目基本的设计思路。

4.2.2 功能模块设计

依据前面的设计思路，把在线考试系统划分为三大模块，分别为考生管理模块、教师管理模块、系统管理模块。系统功能结构图如图 4.2 所示。

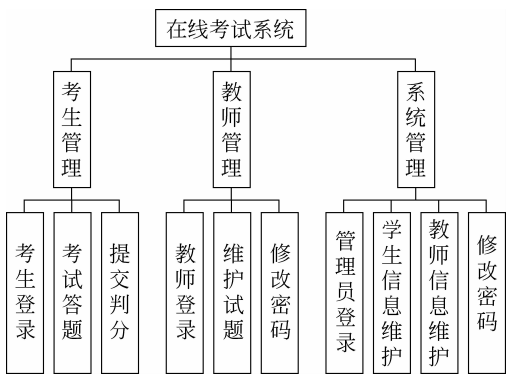

图 4.2 系统功能结构图

4.2.3 数据库设计

1. 概念结构设计

依据前面的数据分析和需求，抽象出考生、教师和管理员用户实体和填空题、选择题试卷信息实体共六个，它们的 E-R 图如图 4.3～图 4.7 所示。

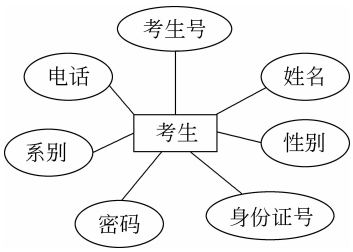

图 4.3 考生 E-R 图

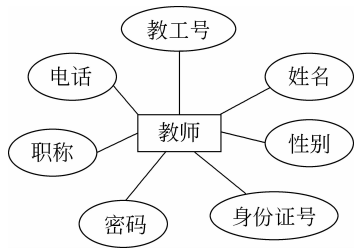

图 4.4 教师 E-R 图

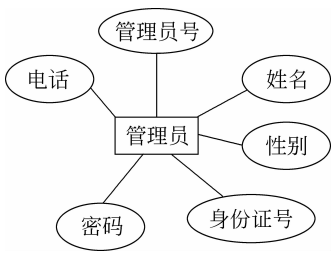

图 4.5 管理员 E-R 图

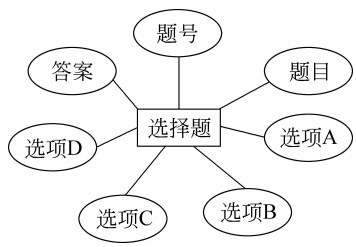

图 4.6 选择题 E-R 图

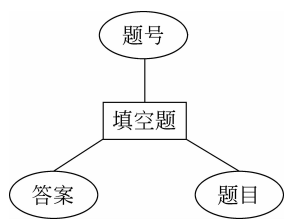

图 4.7 填空题 E-R 图

从图 4.3 中可以看到“考生”实体拥有考生号、姓名、性别、身份证号、密码、系别和电话共七项属性。

从图 4.4 中可以看到“教师”实体拥有教工号、姓名、性别、身份证号、密码、职称和电话七项属性。

从图 4.5 中可以看到“管理员”实体拥有管理员号、姓名、性别、身份证号、密码和电话六项属性。

从图 4.6 中可以看到“选择题”实体拥有题号、题目、选项 A、选项 B、选项 C、选项 D、和答案七项属性。

从图 4.7 中可以看到“填空题”实体拥有题号、题目和答案三项属性。

2. 逻辑结构设计

通过概念结构设计获得了实体和实体拥有的属性，下面采用关系型数据库把概念设计模型中的实体转换为关系模型下的关系，即进行数据库的逻辑结构设计。

逻辑结构设计得出的五种实体关系如表 4.1 所示。

表 4.1　逻辑结构设计实体关系表

关 系 表 名	用　　途
Student	学生实体关系表,用于保存本系统的考生信息
Teacher	教师实体关系表,用于保存本系统的教师信息
Administrator	管理员实体关系表,用于保存本系统的管理员信息
FillQuestion	填空题实体关系表,用于保存考试系统题库中的填空题及答案
ChoiceQuestion	选择题实体关系表,用于保存考试系统题库中的选择题及答案

1) 学生实体关系表(Student)说明(见表 4.2)

表 4.2　学生实体关系表

字 段 名	类　型	长度	可为空	约束条件	注　　释
sid	varchar	6	否	PK	主键、考生登录 ID
sname	varchar	8	否		考生姓名
sex	varchar	2			考生性别
cardnumber	varchar	18			身份证号
pwd	varchar	3	否		考生密码
department	varchar	20			系别
phone	varchar	11			考生电话

2) 教师实体关系表(Teacher)说明(见表 4.3)

表 4.3　教师实体关系表

字 段 名	类　型	长度	可为空	约束条件	注　　释
tid	varchar	6	否	PK	主键、教师登录 ID
tname	varchar	8	否		教师姓名
sex	Varchar	2			教师性别
cardnumber	varchar	18			身份证号
pwd	varchar	6	否		教师用户密码
title	varchar	4			职称
phone	varchar	11			电话

3）管理员实体关系表(Administrator)说明(见表 4.4)

表 4.4　教师管理员实体关系表

字 段 名	类　型	长度	可为空	约束条件	注　　释
adid	varchar	6	否	PK	主键、管理员登录 ID
aname	varchar	8	否		管理员姓名
sex	Varchar	2			管理员性别
cardnumber	varchar	18			身份证号
pwd	varchar	8	否		管理员用户密码
phone	varchar	11			电话

4）填空题实体关系表(FillQuestion)说明(见表 4.5)

表 4.5　填空题实体关系表

字 段 名	类　型	长度	可为空	约束条件	注　　释
f_id	int		否	PK	题目编号、主键、自增
f_question	varchar	200	否		题目、唯一
f_answer	varchar	50			答案

5）选择题实体关系表(ChoiceQuestion)说明(见表 4.6)

表 4.6　选择题实体关系表

字 段 名	类　型	长度	可为空	约束条件	注　　释
c_id	int		否	PK	题目编号、主键、自增
c_question	varchar	200	否		题目、唯一
c_choiceA	varchar	100			选项 A
c_choiceB	varchar	100			选项 B
c_choiceC	varchar	100			选项 C
c_choiceD	varchar	100			选项 D
c_answer	varchar	2			答案

4.2.4　类的分层设计

依据前面的设计思路，在线考试系统采用分层设计模式，可设计实体类(entity)、数据访问类(dao)和视图类(view)三种不同性质的类，并将其分别定义在 entity、dao 和 gui 包中，使逻辑层次更清晰。

1. 实体类

实体类也称 Bean 类，因通常代表数据表中的数据实体信息而得名，实体类一般包含与数据表结构相同的私有成员变量和对这些变量进行操作的公有 get 和 set 方法。