CHAPTER 3

第3章

控制台应用程序开发

本章开始，介绍实际应用项目（或简化、模拟）的开发。首先是控制台应用程序的设计与开发，接着是 Applet 应用程序的介绍，然后是 AWT 和 Swing 的使用，IO 流的操作，文件读写操作，多线程开发介绍，网络编程，最后是 Java Web 项目的开发。对于刚接触 Java 编程的读者，在练习本章程序时，不建议使用 Eclipse 集成开发工具，可以先借助一般文本编辑工具、javac、java 命令，以便更好的理解 Java 的编译和执行过程。对于已经有 Java 编程经验的读者，可以略过本章。

软件开发的读者可能对控制台应用程序不是很了解，因为现在使用控制台的应用系统不常见，大多数软件都是界面友好的窗口操作，借助鼠标就可以完成各种操作。控制台应用程序是为了兼容 DOS 程序而设立的，这种应用程序是在一个 DOS 窗口中执行的，没有自己的界面。本章借助一个银行存取款的实际应用，介绍控制台应用程序的开发。学习完本章，读者可以对控制台应用程序有一个比较清晰的认识。

3.1 相 关 知 识

本章应用程序开发比较简单，所涉及的知识点主要有控制台数据的读写，用户数据、账户信息的存储。

3.1.1 控制台数据的读写

Java 读写任何数据信息，都是借助数据流（类似自来水管道，数据从中流动）进行的，关于数据流的详细说明可以阅读第 6 章，在此读者知道就可以了。

向控制台写数据比较简单，借助 System. out. println()即可。从控制台读取数据的方式比较多，根据 JDK 的版本不同，有不同的方式，下面逐一说明。

1. JDK1.4 及以前版本的读取

获取控制台的输入数据，可以借助 System. in，将其“包装”成带缓冲的数据流。使用方式是固定的，如 BufferedReader br = new BufferedReader(new InputStreamReader(System. in))，然后接着 BufferedReader 的 readLine()方法，即可读取控制台上输入的一行数据。详见如下程序：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
public class BTS {
 public static void main(String args[]) throws IOException{
       System.out.println("请输入语句: ");
       BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
       String str = br.readLine();
       System.out.println("Repeat : " + str);
 }
}
```

BufferedReader.readLine()会抛出 IOException 异常,为了处理的简单,程序直接抛出异常,未进行处理。输出结果如下所示:

```
请输入语句:
welcome java
Repeat: welcome java
```

2. JDK5.0 增加的读取(版本的编号方式改变了,5.0 就是 1.5)

使用 java.util 包中的 Scanner 类读取控制台数据,在构建 Scanner 对象时,可以将 System.in 传入,然后使用 next()系列方法,读取输入。Scanner 对象的功能更加强大,可以直接读出任意基本类型数据,而不需要转换。详见如下程序:

```
import java.util.Scanner;
public class BTS {
 public static void main(String args[]){
       Scanner s = new Scanner(System.in);
       System.out.println("请输入语句: ");
       String str = s.nextLine();
       System.out.println("RepeatString : " + str);

       System.out.println("请输入整数: ");
       int num = s.nextInt();
       System.out.println("RepeatInt : " + num);

       System.out.println("请输入实数: ");
       float fot = s.nextFloat();
       System.out.println("RepeatFloat : " + fot);
 }
}
```

程序的执行结果,如下所示。

```
请输入语句:
welcome java
RepeatString : welcome java
请输入整数:
95
RepeatInt : 95
```

```
请输入实数:
12.5
RepeatFloat : 12.5
```

需要注意的是,在读取基本类型时,如果数据类型无法匹配,会抛出 InputMismatchException 异常,所以在使用 next 系列方法时,可以使用 Try Catch 语句。

3. JDK6.0 增加的读取

使用 Console 类实现读取控制台的数据输入,详见如下程序:

```
import java.io.Console;
public class BTS {
 public static void main(String args[]){
       Console c = System.console();
       if(c == null){
              System.out.println("控制台输入无法使用!");
       }
       String str = c.readLine();
       System.out.println("RepeatString : " + str);
 }
}
```

程序运行结果如下:

```
welcome java
Repeat : welcome java
```

需要注意的是,Console 控制台就只有一个,无法产生多个对象,所以上面的这段程序无法在 Eclipse 中运行成功,原因就在于 Eclipse 会一直占有控制台,应用程序是无法获取到的。

上面介绍的三种从控制台获取数据的方法,比较常用的是第一种,兼容性也比较好,不会因为 JDK 的版本低而无法执行,具体使用哪种方法,读者可以自己尝试它们的区别。

3.1.2 数据的存放

一般的应用程序,都可以简化为三部分,即输入、处理和输出,每个部分都有多种方式来实现,不同的实现方式,决定了不同的逻辑处理过程,也就形成了不同类型的应用程序,如图 3.1 所示。比如日记本程序,可以接受来自键盘和鼠标的数据,经过相应的处理,将用户的数据保存在文件中或数据库中;售后服务管理平台可以将数据保存在数据库中等。

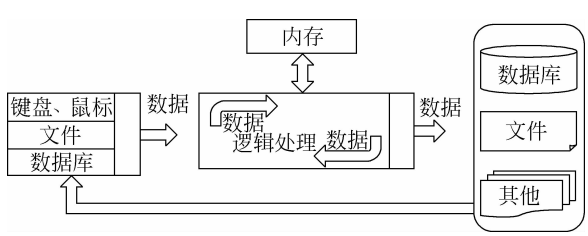

图 3.1 常见应用程序的组成

存在于数据库、文件或其他外围设备中的数据可以持久保存，即使程序结束，保存的数据也不会消失；而在处理过程中，暂存在内存中的数据，等程序关闭后，数据就被清除了。

本章模拟银行存取款应用程序，其数据的来源是用户在控制台上的输入，上一节已经详细说明。数据的存放是在内存中，如果读者想实现数据库或文件的存放，可以在学习完第7章和第10章后，扩展这个应用程序。在此只需要将用户数据、账户信息放在数组或集合中，置于内存中即可，无须考虑持久保存数据。

3.2　需求分析

现实中的银行终端存取款应用程序是极为复杂的，本章旨在模拟该操作，实现简单的存取款，查询与退出功能。为了能够实现多个账号的查询操作，模拟终端系统还提供“开户”功能，即允许用户申请账户，账号由系统自动生成，用户登录时输入账号和密码即可。

简易银行终端模拟器的功能要求如下：

- 用户选择申请账户时，输入用户姓名和相应密码等信息，系统就生成一个用户，一个与其对应的账号，并将账号显示给用户。
- 用户选择登录时，要求输入账号和密码，验证无误，允许用户进入存取款操作。
- 在存取款操作中，用户可以实现存款，取款，查询余额等操作。

3.3　系统设计

由于该应用程序是在控制台上运行，不涉及操作界面，在用户输入相应数据时，输出提示语句即可。下面详细说明该应用程序的功能部分。

3.3.1　类的设计

在该应用程序中，划分用户和账户，以及终端三个类就足够了，其中账户类持有用户类的引用，用户类也持有账户类的引用(双向引用)。具体各类的明细如表3.1所示。

表3.1　程序类明细

类名	方　法　名	说　　明
BTS	regest()	注册方法
	login()	登录方法，返回验证状态，返回卡号：OK 1:查无此卡 2：密码错误
	opration(String cardNum)	完成存取款操作
Account	xxSet()	设置 Account 类的属性
	xxGet()	获取 Account 类的属性
User	xxSet()	设置 User 类的属性
	xxGet()	获取 User 类的属性

为了简化系统复杂度，在实现过程中规定一个用户只对应一个账户，如果需要实现一个用户拥有多个账户的功能，可以在用户类中用一个集合存放账户对象即可。用户类的基本属性包括用户姓名和账户，以及其他相应信息(根据需要添加)。账户类的基本属性包括账

号，密码，余额，户主(用户类)，以及其他信息(根据需要添加)。

3.3.2 流程处理

程序运行后，根据提示用户选择注册账户、登录或退出操作。当用户选择注册账户后，进入账户注册流程，完成必要信息的输入，生成账号。通过账号和密码就可以登录系统，经过账号和密码核实后，进入存取款操作流程，在此用户可以选择存款、取款、查询和退出操作。程序流程详见图 3.2。

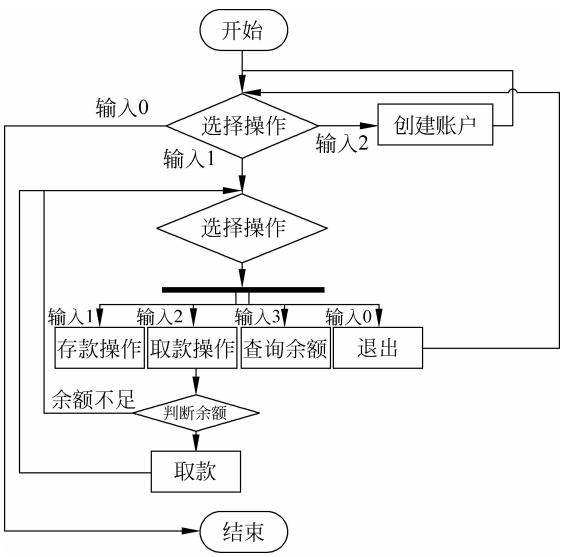

图 3.2 简易银行终端流程图

3.4 编 码 实 现

简易银行终端应用程序源代码文件有三个，分别是 BTS.java 包含 BTS 类，完成简易银行终端的业务操作(逻辑处理)；User.java 包含 User 类；Account.java 包含 Account 类。其中后两个不涉及业务操作，称为实体类(虽然都是 Java 类，根据其操作特点又划分为很多类型，随着学习的深入，会逐渐接触)。完整代码和解释如下：

```
/*
代码文件: BTS.java
功能描述: 包含 BTS 类,主要实现简易银行终端的逻辑处理
*/
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.HashMap;
import java.util.Map;

public class BTS {
 public static long cardID = 20101001;            //用于生成卡号的卡号生成器
```

```
//用 Map 集合存放账户对象和用户对象
public static Map<String ,Account> accounts = new HashMap<String,Account>();
public static Map<String,User> users = new HashMap<String,User>();
//使用 JDK1.4 控制台读取
public static BufferedReader br = new BufferedReader(
                                                    new InputStreamReader(System.in));
//main 方法
public static void main(String []args){
      String line;                                  //接收控制台输入
      int flag = -1;                                //控制变量
      //主要逻辑判断部分
      try{
            while (flag != 0){
                  System.out.println("*****Welcome to BTS*****");
                  System.out.println("1->登录     2->注册     0->退出");
                  System.out.println("**********************");
                  line = br.readLine();
                  flag = Integer.parseInt(line);
                  if(flag == 1){
                        String r = login();
                        if(r! = null && r.length()> 2){
                              opration(r);
                        }
                 }else if(flag == 2){
                 regest();
           }
         }
    }catch(IOException e){
        e.printStackTrace();
    }
}
//注册方法
public static void regest(){
     try{
        //生成用户对象和账户对象
        User user = new User();
        Account acot = new Account();
        //输入对象的属性
        System.out.println("请输入用户名:");
        user.setName(br.readLine());
        System.out.println("请输入联系方式:");
        user.setPhone(br.readLine());
        System.out.println("请设定密码:");
        acot.setPwd(br.readLine());
        //生成卡号
        String cardNum = cardID + "";
        acot.setCardNum(cardNum);
        System.out.println("你的卡号是:" + cardNum);
        //双向引用
        acot.setBts(user);
        user.setAccount(acot);
```

```
            //存放到集合中
            users.put(cardNum, user);
            accounts.put(cardNum, acot);
            cardID++; //卡号生成器自增
        }catch(IOException e){
            e.printStackTrace();
        }
    }
    //登录方法,返回验证状态    返回卡号: OK  1:查无此卡    2: 密码错误
    private static String login() {
          String cardNum = null;
          System.out.println("请输入卡号:");
          try {
                cardNum = br.readLine();
                Account a  =  accounts.get(cardNum);
                if(a == null){
                      System.out.println("查无此卡,请确认后再输入");
                      return "1";
                }else{
                      System.out.println("请输入密码: ");
                      String pwd  =  br.readLine();
                      if(!pwd.equalsIgnoreCase(a.getPwd())){
                            System.out.println("密码错误!");
                                  return "2";
                      }
               }
        } catch (IOException e) {
              e.printStackTrace();
        }
        return cardNum;
    }
    //存取款操作方法
    private static void opration(String cardNum) {
        String line;                                  //接收控制台输入
        int flag = -1;                                //控制变量
        float money;
        //主要逻辑判断部分
        try{
              while (flag ! = 0){
                    System.out.println("*****Welcome to BTS*****");
                    System.out.println("1->存款                2->取款");
                    System.out.println("3->查询                0->退出");
                    System.out.println("************************");
                    line = br.readLine();
                    flag = Integer.parseInt(line);
                    Account a  = accounts.get(cardNum);
                    if(flag == 1){
                          System.out.println("请输入存入金额: ");
                          line = br.readLine();
                          money = Float.parseFloat(line);
                          a.setMoney(a.getMoney() + money);
```

```
                System.out.println("操作成功!");
            }else if(flag = = 2){
                System.out.println("请输入取款金额: ");
                line = br.readLine();
                money = Float.parseFloat(line);
                if(money > a.getMoney()){
                    System.out.println("余额不足!");
                    continue;
                }
                a.setMoney(a.getMoney() - money);
                System.out.println("操作成功!");
            }else if(flag = = 3){
                System.out.println("账户信息: ");
                System.out.println("户主: " + users.get(cardNum).getName());
                System.out.println("余额: " + a.getMoney());
            }
        }
      }catch(IOException e){
          e.printStackTrace();
      }
    }
}
```

下面两个是实体类,主要用于封住属性,提供 get 和 set 操作。

```
/*
代码文件: Account.java
功能描述: 包含 Account 类,账户类
*/
public class Account {
 private String cardNum;
 private String pwd;
 private float money;
 private User user;                                    //用户的引用
 public Account() {
       super();
 }
 public Account(String cardNum, String pwd, float money, User user) {
       super();
       this.cardNum = cardNum;
       this.pwd = pwd;
       this.money = money;
       this.user = user;
 }
 public String getCardNum() {
      return cardNum;
 }
 public void setCardNum(String cardNum) {
      this.cardNum = cardNum;
 }
```

```
 public String getPwd() {
       return pwd;
 }
 public void setPwd(String pwd) {
       this.pwd = pwd;
 }
 public float getMoney() {
       return money;
 }
 public void setMoney(float money) {
       this.money = money;
 }
public User getBts() {
     return user;
 }
public void setBts(User user) {
     this.user = user;
 }

}

/*
代码文件：User.java
功能描述：包含 User 类,用户类
*/
public class User {
 private String name;
 private String phone;
 private Account account;                          //账户类的引用
 public String getName() {
       return name;
 }
 public void setName(String name) {
       this.name = name;
 }
 public String getPhone() {
       return phone;
 }
 public void setPhone(String phone) {
       this.phone = phone;
 }
 public Account getAccount() {
       return account;
 }
 public void setAccount(Account account) {
       this.account = account;
 }
}
```

3.5　测 试 运 行

经测试运行，简易银行终端模拟程序实现预期功能，运行效果如图 3.3 所示。

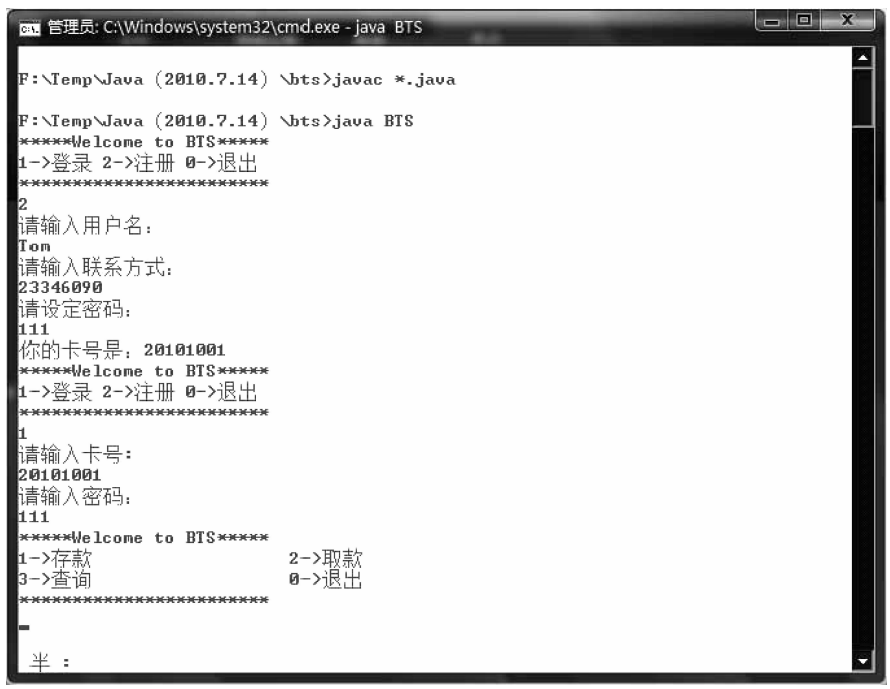

图 3.3　控制台运行简易银行终端

扩展练习 3

1. 增加账户透支功能，当用户取款时，允许用户透支一定的额度。

2. 增加密码错误次数限制功能，当用户输入密码错误达三次时，不允许再做尝试，锁定该账户(冻结)，使其无法进入存取款操作。

3. 增加多账户功能，一个用户可以申请多个账户。在查询功能下面，增加查询该用户其他账户的功能。