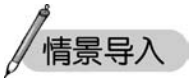
第5章 阶段项目：自动取款机模拟程序



软件工程师小明需要开发一个基于控制台的自动取款机(Automatic Teller Machine，ATM)模拟程序。该系统支持以下功能。

(1) 开户：用户可以使用 ATM 开设账户，开设账户需要提供用户姓名、密码和存入金额等信息。

(2) 登录：用户可以通过 ATM 登录账户，对账户进行管理。用户登录账户时，需要验证账号的合法性。

(3) 管理账户：用户登录到自己的账户，可以进行如下操作。

- 存款：将一定金额的款项存储到当前账户。
- 取款：从当前账户中取出一定数量的款项。
- 转账：将一定数额的款项从当前账户转到另外一个账户。
- 查询余额：查询当前账户的余额。

在学习完本章内容后，读者将能够：

- 了解软件开发的基本流程。
- 使用 Visual Studio 开发和调试 C# 程序。
- 利用 C# 语言编写简单的应用程序。

视频讲解

5.1 项目概述

5.1.1 工作流程

ATM 模拟程序是一个控制台应用程序，其工作流程如图 5-1 所示。

程序运行时首先让用户输入账号和密码。如果账号不存在，则跳转到开户界面。开户成功后，显示服务菜单。如果用户名和密码不匹配，则提示重新输入，连续输入三次不成功，退出应用程序。用户名和密码正确，显示服务菜单，用户输入相应的功能号，执行对应模块。图 5-2 给出了程序服务界面。

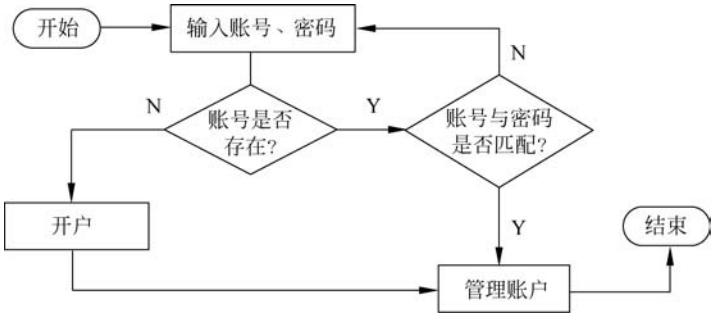


图 5-1　ATM 模拟程序工作流程

图 5-2　程序服务界面

5.1.2　系统类图

在 ATM 模拟程序中，需要建立账户类 Account、账户管理类 AccountManager 和 ATM 类。其中，ATM 类提供用户交互界面，AccountManager 类提供账户管理逻辑，Account 类用来保存账户的信息，图 5-3 给出了这三个类的详细信息。

图 5-3　系统类图

5.2 项目的实现


视频讲解

5.2.1 账户类

在 ATM 模拟程序中，账户类用来保存账户信息，包括账号、姓名、密码和余额 4 个属性。具体代码如下：

```
class Account
{
    public string No { set; get; }                  //账号
    public string Name { set; get; }                //姓名
    public string Password { set; get; }            //密码
    public double Balance { set; get; }             //余额
}
```

5.2.2 账户管理类

账户管理类用来管理账户，包括账户列表字段以及对账户进行管理的方法，如开户、登录、存款、取款、转账和查询余额等。具体代码如下：

```
class AccountManager
{
    //账户列表
    private List<Account> lstAccount = new List<Account>();
    //构造函数
    public AccountManager()
    {
        lstAccount.Add(new Account() { No = "123456", Name = "李伟", Password =
"123456" });
        lstAccount.Add(new Account() { No = "654321", Name = "李明", Password =
"123456" });
    }
    //账户是否存在
    public bool IsAccount(string id)
    {
        foreach(var ac in lstAccount)
        {
            if (ac.No.Equals(id))
                return true;
        }
        return false;
    }
    //账户登录验证
    public bool LoginAccount(string id,string pass)
    {
        bool isSucced = false;
        if (IsAccount(id))
        {
```

```
            foreach(var ac in lstAccount)
            {
                if (ac.Password.Equals(pass))
                {
                    isSucced = true;
                    break;
                }
            }
        }
        return isSucced;
    }
    //开户
    public bool CreateAccount(Account ac)
    {
        bool succed = true;
        try
        {
            lstAccount.Add(ac);
        }
        catch(Exception cx)
        {
            Console.WriteLine(cx.ToString());
            succed = false;
        }

        return succed;
    }
    //存款
    public void Despoit(string id, double money)
    {
        foreach(var ac in lstAccount)
        {
            if(ac.No.Equals(id) && money > 0)
            {
                ac.Balance = ac.Balance + money;
                break;
            }
        }
    }
    //取款
    public void WithDraw(string id,double money)
    {
        foreach (var ac in lstAccount)
        {
            if (ac.No.Equals(id) && money > 0 && ac.Balance > money)
            {
                ac.Balance = ac.Balance - money;
                break;
            }
        }
    }
```

```
    //查询
    public Account GetAccount(string id)
    {
        Account obj = null;

        foreach (var ac in lstAccount)
        {
            if (ac.No.Equals(id))
            {
                obj = ac;
                break;
            }
        }

        return obj;
    }
    //转账
    public bool TransferAccount(string sid,string did,double money)
    {
        bool isSucced = false;
        bool flag1 = false;
        bool flag2 = false;
        if(IsAccount(did))
        {
            foreach (var ac in lstAccount)
            {
                if(ac.No.Equals(sid) && money > 0 &&ac.Balance > money)
                {
                    ac.Balance = ac.Balance - money;
                    flag1 = true;
                }

                if (ac.No.Equals(did) && money > 0)
                {
                    ac.Balance = ac.Balance + money;
                    flag2 = true;
                }

                if(flag1 && flag2)
                {
                    isSucced = true;
                    break;
                }
            }
        }

        return isSucced;
    }
}
```

5.2.3 ATM 类

ATM类提供用户与ATM交互的界面,实现用户与系统的交互,包括账户管理对象属性和服务菜单显示等方法,具体代码如下:

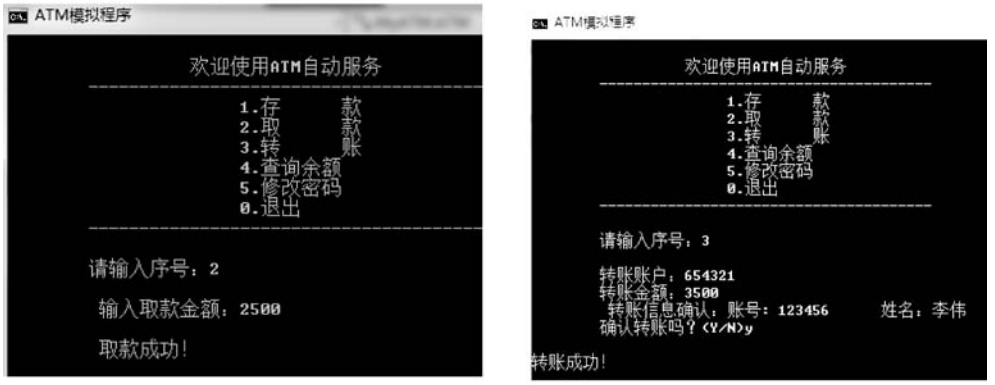
```
class ATM
{
    AccountManager manager;                          //账户管理对象
    Account currentAccount;                          //当前账户
    int count;                                       //计数器,统计输入密码次数
    bool flag;                                       //控制服务菜单显示

    //构造函数
    public ATM()
    {
        manager = new AccountManager();
        currentAccount = new Account();
        count = 0;
        flag = true;
    }
    //信息输入
    public string InputString(string msg)
    {
        Console.Write(msg);
        string str = Console.ReadLine();
        return str;
    }
    //信息输入(数字)
    public double InputNumber(string msg)
    {
        Console.Write(msg);
        string number = Console.ReadLine();
        return Convert.ToDouble(number);
    }
    //功能菜单
    public void ServiceMenu(string ID)
    {
        while(flag)
        {
            Console.Clear();
            Console.WriteLine("\n\t\t欢迎使用ATM自动服务");
            Console.WriteLine("\t----------------------------------------");
            Console.WriteLine("\t\t       1.存       款");
            Console.WriteLine("\t\t       2.取       款");
            Console.WriteLine("\t\t       3.转       账");
            Console.WriteLine("\t\t       4.查询余额");
            Console.WriteLine("\t\t       0.退出");
            Console.WriteLine("\t----------------------------------------");
            Console.Write("\n\t请输入序号:");
            string choice = Console.ReadLine();
```

```
        switch(choice)
        {
            case "0":
                flag = false;
                break;
            case "1":
                double number = InputNumber("输入存款金额: ");
                manager.Despoit(ID, number);
                break;
            case "2":
                double num = InputNumber("输入取款金额: ");
                manager.WithDraw(ID, num);
                break;
            case "3":
                string did = InputString("\t 转账账户: ");
                double money = InputNumber("\t 转账金额: ");
                Account tmpAccount  =  manager.GetAccount(ID);
                Console.Write( $ "\t 账号: {tmpAccount.No} \t\n 姓名: {tmpAccount.Name} \n\t
                                        确认转账吗(Y/N)?");
                string ok = Console.ReadLine();
                if(ok.ToUpper().Equals("Y"))
                {
                   if(manager.TransferAccount(ID, did, money))
                    {
                        Console.WriteLine("\n 转账成功!");
                    }
                }
                break;
            case "4":
                Account ac = manager.GetAccount(ID);
                Console.Clear();
                Console.WriteLine( $ "\t 账号: {ac.No}  姓名: {ac.Name}  余额:
                                        {ac.Balance:C2}");
                Console.ReadKey();
                break;
        }
    }
}
//用户登录
public void LoginAccount()
{
    Console.Clear();
    Console.WriteLine("\n---------- 欢迎使用 ATM 自助服务 ----------");
    string no = InputString("\t 账号: ");
    if (!manager.IsAccount(no))
    {
        Console.Write("用户不存在,需要开户吗(Y/N)?");
        string choice = Console.ReadLine();
        if (choice.ToUpper().Equals("Y"))
        {   //开户
```

```
                    Console.Clear();
                    //建立账户对象
                    Account ac = new Account();
                    Console.WriteLine("\t 欢迎使用 ATM 柜员机\n");
                    ac.No = InputString("输入账号: ");
                    ac.Name = InputString("输入密码: ");
                    ac.Balance = InputNumber("存款金额: ");
                    if (manager.CreateAccount(ac))
                    {
                        Console.WriteLine("开户成功!");
                        currentAccount = ac;
                        //进入服务菜单
                        ServiceMenu(ac.No);
                    }
                }
                else
                {
                    return;
                }
            }
            else
            {
                string pwd = InputString("\t 密码: ");
                if (!manager.LoginAccount(no, pwd))
                {
                    count++;
                    while (count < 3)
                    {
                        Console.Clear();
                        no = InputString("账号: ");
                        pwd = InputString("密码: ");
                        if (manager.LoginAccount(no, pwd))
                        {
                            break;
                        }
                        count++;
                    }
                    if (count >= 3)       return;
                }
                //记录当前账号
                currentAccount.No = no;
                //显示服务菜单
                ServiceMenu(currentAccount.No);
            }
        }
    }
```

运行上述程序,输入正确的账号和密码,显示如图 5-2 所示的程序服务界面,选择相应的功能代码,可以进行存款、取款、转账和查询余额等功能,运行结果如图 5-4 所示。

(a) 存款和查询余额界面



(b) 取款和转账界面

图 5-4　ATM 程序运行结果

5.3　知识点提炼

(1) 使用 C# 开发应用程序时，首先根据程序的业务流程确定程序的功能以及程序中的类以及类的成员。

(2) 类是一个封装了数据成员和函数成员的数据单元，是面向对象编程的基本结构。在 C# 程序中，通过 Class 关键字来定义类的类型。

(3) 在面向对象编程中，通过方法调用完成业务逻辑的处理。方法调用时，需要为方法提供相匹配的参数。

(4) 利用 Visual Studio 2017 解决方案视图中的类视图可以直观地查看项目中类的结构以及它们的可见范围。

5.4　思考与练习

在本章银行 ATM 示例的基础上，利用所学的面向对象的思想及语法进行改进。要求如下：

1. 使用面向对象的编程思想模拟真实世界中的银行、账号和 ATM 等对象，其中类中有字段、方法。

2. 在程序中适当的地方使用属性、索引器等面向对象的高级特征。

3. 使用事件。在程序中适当位置使用事件，例如当取款金额超过 10 000 元时，向用户发送提示信息。

4. 设计一个工具类，用来生成随机的银行账号。

5. 使用继承。继承账号(Account 类)得到一个子类(如信用账号)，增加字段(如信用额度)、属性、方法，覆盖(override)一些方法(如 WithdrawMoney)。

6. 根据程序的需要，使用 C# 的其他语法成分，如接口、结构、枚举等。