# 实例 1：银行管理系统

从早期的钱庄到现如今的银行，金融行业在不断地变革；随着科技的发展、计算机的普及，计算机技术在金融行业得到了广泛的应用。银行管理系统是一个集开户、查询、取款、存款、转账、锁定、解锁、退出等一系列的功能的管理系统，该系统中各功能的介绍如下。

- 开户功能：用户在 ATM 机上根据提示“请输入姓名:”、“请输入身份证号:”、“请输入手机号:”依次输入姓名、身份证号、手机号、预存金额、密码等信息，如果开户成功，系统随机生成一个不重复的 6 位数字卡号。
- 查询功能：根据用户输入的卡号、密码查询卡中余额，如果连续 3 次输入错误密码，该卡号会被锁定。
- 取款功能：首先根据用户输入的卡号、密码显示卡中余额，如果连续 3 次输入错误密码，该卡号会被锁定；然后接收用户输入的取款金额，如果取款金额大于卡中余额或取款金额小于 0，系统进行提示并返回功能页面。
- 存款功能：首先根据用户输入的卡号、密码显示卡中余额，如果连续 3 次输入错误密码，该卡号会被锁定，然后接收用户输入的取款金额，如果存款金额小于 0，系统进行提示并返回功能页面。
- 转账功能：用户需要分别输入转出卡号与转入卡号，如果连续 3 次输入错误密码，卡号会被锁定。当输入转账金额后，需要用户再次确认是否执行转账功能；如果确定执行转账功能，转出卡与转入卡做相应金额计算；如果取消转账功能，则回退之前操作。
- 锁定功能：根据输入的卡号密码执行锁定功能，锁定之后该卡不能执行查询、取款、存款、转账等操作。
- 解锁功能：根据输入的卡号密码执行解锁功能，解锁后能对该卡执行查询、取款、存款、转账等操作。
- 存盘功能：执行存盘功能后，程序执行的数据会写入本地文件中。
- 退出功能：执行退出功能时，需要输入管理员的账户密码，如果输入的账号密码错误，则返回功能页面，如果输入的账号密码正确，则执行存盘并退出系统。

本实例要求编写程序，实现一个具有上述功能的银行管理系统。

## 实例目标

- 理解面向对象的思想
- 熟练地定义类
- 熟练地创建对象、访问类的成员
- 熟练使用构造方法

# 实例分析

实际生活中，银行管理系统在由银行工作人员打开时先显示欢迎界面，之后工作人员输入管理员账号与密码，银行管理系统被启动，启动后进入系统功能页面，可观察到该页面中展示了使用 ATM 机可办理的所有业务，包括开户（1）、查询（2）、取款（3）、存款（4）、转账（5）、锁定（6）、解锁（7）、退出（Q）等。用户可根据自己需求选择相应业务的编号，并按照提示完成相应的操作。

从以上模拟的过程中可知，要实现银行管理系统需要用到 5 种对象，分别是管理员、ATM 机、银行卡、用户、银行管理系统。因此，我们需要设计 5 个类承担不同的职责，关于这些类的说明如下：

（1）银行管理系统类（HomePage）：负责提供整个系统流程的相关操作，包括打印欢迎登录界面和功能界面、接收用户输入、保存用户数据等。

（2）ATM 机类（ATM）：负责处理系统中各个功能的相关操作，包括开户、查询、取款、存款、转账、锁定、解锁、退出功能。

（3）管理员类（Admin）：负责提供检测管理员账号与密码、显示欢迎登录界面和功能界面的相关操作。

（4）用户类（User）：负责提供用户对象的相关操作。

（5）银行卡（Card）：负责提供银行卡对象的相关操作。

设计后的类结构如图 1 所示。

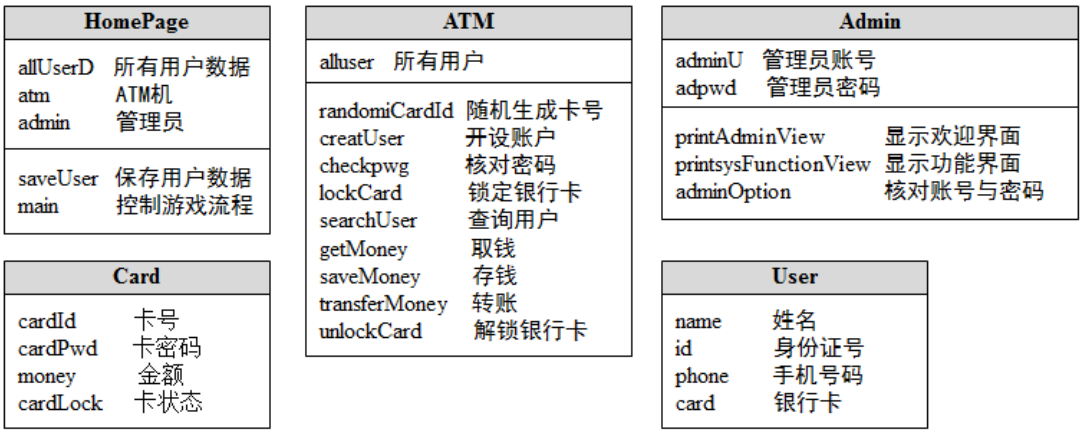


图 1 类设计图

本实例中涉及到多个类，为保证程序具有清晰的结构，可以将每个类的相关代码分别放置到与其同名的.py 文件中。

# 代码实现

本实例的具体实现过程如下所示。

（1）打开 PyCharm 工具，创建一个名为“银行管理系统”的文件夹。在该文件夹下创建 5 个.py 文件，分别为 admin.py、atm.py、card.py、user.py 与“银行系统.py”，此时程序的目录结构如图 2 所示。

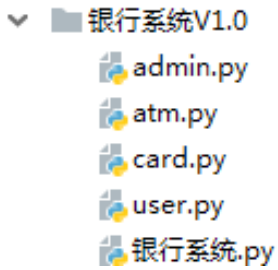


图 2 目录结构

（2）按照图 1 的类设计图，在 card.py 文件中编写 Car 类的代码，具体如下所示。

```
class Card:
    def __init__(self,cardId,cardPwd,money):
        self.cardId = cardId
        self.cardPwd = cardPwd
        self.money = money
        self.cardLock = False
```

（3）按照图 1 的类设计图，在 user.py 文件中编写 User 类的代码，具体如下所示。

```
class User:
    def __init__(self, name, id, phone, card):
        self.name = name
        self.id = id
        self.phone = phone
        self.card = card
```

（4）按照图 1 的类设计图，在 admin.py 文件中编写 Admin 类的代码，具体如下所示。

```
class Admin:
    adminU = '1'  # 管理员的账号
    adpwd = '1'  # 管理员的密码
    def printAdminView(self):
        print("*****************************************")
        print("***                                   ***")
        print("***                                   ***")
        print("***         欢迎登录银行系统           ***")
        print("***                                   ***")
        print("***                                   ***")
        print("*****************************************")
    def printsysFunctionView(self):
        print("***********************************************")
        print("***                                        ***")
        print("***    1.开户(1)              2.查询(2)    ***")
        print("***    3.取款(3)              4.存款(4)    ***")
        print("***    5.转账(5)              6.锁定(6)    ***")
```

```
        print("***    7.解锁(7)                          ***")
        print("***                                       ***")
        print("***    退出(Q)                            ***")
        print("***                                       ***")
        print("*********************************************")
    def adminOption(self):
        adminInput = input("请输入管理员账户：")
        if self.adminU != adminInput:
            print("管理员账户输入错误......")
            return -1
        passwordInput = input("请输入密码：")
        if self.adpwd != passwordInput:
            print("输入密码有误......")
            return -1
        else:
            print("操作成功，请稍后......")
            return 0
```

（5）按照图 1 的类设计图，在 atm.py 文件中编写 ATM 类的代码。ATM 是本实例的核心类，该类中包含所有与系统功能相关的方法。由于 ATM 类包含开户的功能，在实现这些功能时需要访问 Card 与 User 类的属性，而且这些类分别处于不同的 py 文件中，因此这里需提前使用 import 语句（下章会有相关介绍）导入 Card 和 User 类，此时便可以在 ATM 类中访问 Card 类与 User 类的属性，具体代码如下

```
from user import User
from card import Card
```

下面分别介绍 ATM 类的属性与方法。

- alluser 属性

在 ATM 类的构造方法中添加属性 alluser，具体代码如下。

```
class ATM:
    def __init__(self, alluser):
        self.alluser = alluser
```

- randomiCardId()方法

randomiCardId()方法的作用是在用户开户时生成随机银行卡号，该方法中需要借助 random 模块的函数生成随机数，也需要排除生成重复卡号的情况，具体代码如下。

```
import random
def randomiCardId(self):     # 随机生成开户卡号
    while True:
        str_data = ''         # 存储卡号
        for i in range(6):    # 随机生成 6 位卡号
            ch = chr(random.randrange(ord('0'), ord('9') + 1))
```

```
            str_data += ch
        if not self.alluser.get(str):  # 判断卡号是否重复
            return str_data
```

- creatUser()方法

creatUser()方法用于为用户开设账户，在该方法中需要用户先根据提示信息依次输入姓名、身份证号、手机号、预存金额（必须大于 0，否则提示“开户失败”），再连续输入两次银行卡的密码（必须一致，否则因开户失败重新回到功能界面），最后随机生成开户卡号，根据该卡号创建卡信息和用户信息，并将用户的信息存储到 alluser 中，具体代码如下。

```
def creatUser(self):
    # 目标向用户字典中添加一个键值对（卡号、用户对象）
    name = input("请输入姓名:")
    Uid = input("请输入身份证号:")
    phone = input("请输入手机号:")
    prestMoney = float(input("请输入预存金额:"))
    if prestMoney <= 0:
        print("预存款输入有误，开户失败")
        return -1
    oncePwd = input("请输入密码：")
    passWord = input("请再次输入密码：")
    if passWord != oncePwd:
        print("两次密码输入不同......")
        return -1
    print("密码设置成功，请牢记密码： %s " % passWord)
    cardId = self.randomiCardId()
    card = Card(cardId, oncePwd, prestMoney)    # 创建卡
    user = User(name, Uid, phone, card)         # 创建用户
    self.alluser[cardId] = user                  # 存入用户字典
    print("您的开户已完成，请牢记开户账号：%s" % cardId)
```

- checkpwg ()方法

creatUser()方法用于核对用户输入的密码，且限定至多输入 3 次，超过三次则返回 False，输入正确则返回 True，具体代码如下。

```
def checkpwg(self, realPwd):
    for i in range(3):
        psd2 = input("请输入密码：")
        if realPwd == psd2:
            return True
    print("密码输错三次，系统自动退出......")
    return False
```

- lockCard ()方法

lockCard()方法用于在用户主要要求锁定银行卡，该方法中需要用户输入银行卡号，并将银行卡设为锁定状态，锁定失败的情况包括以下任一情况：若用户输入错误的银行卡号，则提示“此卡号不存在...锁定失败!”；若用户持有的银行卡已经处于锁定状态，则提示“该卡已经被锁定，无需再次锁定!”；若用户输入错误的密码，则提示“密码错误...锁定失败!!”。

```
def lockCard(self):
    inptcardId = input("请输入您的卡号：")
    user = self.alluser.get(inptcardId)
    if not self.alluser.get(inptcardId):
        print("此卡号不存在...锁定失败！")
        return -1
    if user.card.cardLock:
        print("该卡已经被锁定，无需再次锁定！")
        return -1
    if not self.checkpwg(user.card.cardPwd):  # 验证密码
        print("密码错误...锁定失败！！")
        return -1
    user.card.cardLock = True
    print("该卡被锁定成功！")
```

- searchUser ()方法

searchUser ()方法实现查询余额的功能，确保用户在发生存钱、取钱和转账行为之前能输入正确的银行卡号，以及银行卡处于非锁定状态，此时返回拥有该银行卡的用户，否则返回-1，具体代码如下。

```
def searchUser(self, base=1):
    if base == 2:
        inptcardId = input("请输入转出主卡号：")
    elif base == 3:
        inptcardId = input("请输入转入卡号：")
    elif base == 1:
        inptcardId = input("请输入您的卡号：")
    user = self.alluser.get(inptcardId)
    # 如果卡号不存在，下面代码就会执行
    if not self.alluser.get(inptcardId):
        print("此卡号不存在...查询失败！")
        return -1
    if user.card.cardLock:
        print("该用户已经被锁定...请解卡后使用！")
        return -1
    if not self.checkpwg(user.card.cardPwd):  # 验证密码
        print("密码错误过多...卡已经被锁定，请解卡后使用！")
```

```
            user.card.cardLock = True
            return -1
        if not base == 3:  # 查询转入账户   不打印余额
            print("账户: %s   余额: %.2f  " % (user.card.cardId, user.card.money))
        return user
```

- getMoney ()方法

getMoney ()方法实现用户使用银行管理系统取钱的功能，该方法中首先需调用searchUser ()方法根据用户输入的卡号返回拥有该卡的用户，然后处理用户输入不同金额的情况：若输入的金额低于银行卡余额，则将取款后的银行卡余额展示给用户，同时提示取款成功；若输入的金额高于银行卡余额，则提示用户信息并返回系统功能界面，具体代码如下。

```
def getMoney(self):
    userTF = self.searchUser()
    if userTF != -1:
        if userTF.card.cardId != '':
            inptMoney = float(input("请输入取款金额:"))
            if inptMoney > int(userTF.card.money):
                print("输入的金额大于余额，请先查询余额！")
                return -1
            userTF.card.money = float(userTF.card.money) - inptMoney
            print("取款成功!  账户: %s   余额: %.2f  " %
                (userTF.card.cardId, userTF.card.money))
    else:
        return -1
```

- saveMoney ()方法

saveMoney()方法实现用户使用银行管理系统存钱的功能，与 getMoney ()方法的功能类似，需要先查询银行卡的用户，查询结果无误后则需要用户输入要存入的金额：金额小于 0 提示错误信息，并返回到银行系统的功能界面；金额大于等于 0 则累加到银行卡的余额中，并向用户展示账户余额，具体代码如下。

```
def saveMoney(self):
    userTF = self.searchUser()
    if userTF != -1:
        if not userTF.card.cardLock == True:
            if userTF.card.cardId != '':
                inptMoney = float(input("请输入要存入得金额:"))
                if inptMoney < 0:
                    print("请输入正确金额")
                else:
                    userTF.card.money += inptMoney
                    print("存款成功!  账户: %s   余额: %.2f  " %
```

```
                    (userTF.card.cardId, userTF.card.money))
    else:
        return -1
```

- transferMoney ()方法

transferMoney ()方法实现用户向其它银行卡转账的功能，该方法中需先确保主卡用户和转入卡用户同时存在，再向用户提示要转入的金额进而判断：若主卡中的余额充足，将减少主卡中相应的金额，将增加转入卡中相应的金额；若主卡中的余额不足，则将为用户显示主卡余额，具体代码如下。

```
def transferMoney(self):
    MasterTF = self.searchUser(base=2)
    if (MasterTF == -1):
        return -1
    userTF = self.searchUser(base=3)
    if (userTF == -1):
        return -1
    in_tr_Money = float(input("请输入转账金额："))
    if MasterTF.card.money >= in_tr_Money:
        str = input("您确认要继续转账操作吗（y/n）？：")
        if str.upper() == "Y":
            MasterTF.card.money -= in_tr_Money
            userTF.card.money += in_tr_Money
            print("转账成功！  账户：%s  余额：%.2f  " %
                  (MasterTF.card.cardId, MasterTF.card.money))
        else:
            print("转账失败，中止了操作")
    else:
        print("转账失败,余额不足！  账户：%s  余额：%.2f  " %
              (MasterTF.card.cardId, MasterTF.card.money))
```

- unlockCard ()方法

unlockCard ()方法实现解锁银行卡的功能，该方法中需要求用户先输入要解锁的银行卡号，再分别对不同的情况做处理：若用户输入错误的卡号，提示用户解锁失败的信息；若用户输入卡号的银行卡未被锁定，提示用户无需解锁；若用户输入错误的密码，提示用户解锁失败；其它情况则提示解锁成功，具体代码如下。

```
def unlockCard(self):
    inptcardId = input("请输入您的卡号：")
    user = self.alluser.get(inptcardId)
    while 1:
        if not self.alluser.get(inptcardId):
            print("此卡号不存在...解锁失败！")
```

```
            return -1
        elif not user.card.cardLock:
            print("该卡未被锁定，无需解锁！")
            break
        elif not self.checkpwg(user.card.cardPwd):
            print("密码错误...解锁失败！！")
            return -1
            user.card.cardLock = False  # 解锁
        print("该卡 解锁 成功！")
        break
```

（6）按照图 7 的类设计图，在“银行系统.py”文件中编写 HomePage 类的代码，具体如下所示。

```
from admin import Admin
from atm import ATM
import time
class HomePage:
    def __init__(self):
        self.allUserD = {}    # 使用字典存储数据
        self.atm = ATM(self.allUserD)
        self.admin = Admin()  # 管理员开机界面
    def saveUser(self):
        self.allUserD.update(self.atm.alluser)
        print("数据存盘成功")
# 程序的入口
    def main(self):
        self.admin.printAdminView()
        resL = self.admin.adminOption()
        if not resL:
            while True:
                self.admin.printsysFunctionView()
                option = input("请输入您的操作：")
                if option not in ("1", "2", "3", "4", "5",
                                  "6", "7", "S", "Q", "q"):
                    print("输入操作项有误,请仔细确认！")
                    time.sleep(1)
                if option == "1":  # 开户
                    self.atm.creatUser()
                elif option == "2":  # 查询
                    self.atm.searchUser()
```

```
            elif option == "3":  # 取款
                self.atm.getMoney()
            elif option == "4":  # 存储
                self.atm.saveMoney()
            elif option == "5":  # 转账
                self.atm.transferMoney()
            elif option == "6":  # 锁定
                self.atm.lockCard()
            elif option == "7":  # 解锁
                self.atm.unlockCard()
            elif option.upper() == "Q":
                if not (self.admin.adminOption()):
                    self.saveUser()
                    print('退出系统')
                    return -1
```

（7）在“银行系统.py”文件中，创建 HomePage 类对象，调用 main()函数，具体代码如下。

```
if __name__ == "__main__":
    homepage = HomePage()
    homepage.main()
```

至此，程序的全部功能全部实现。

## 代码测试

运行程序，在控制台输入管理员账户“1”和密码“1”的结果如下所示：

```
*****************************************
***                                   ***
***                                   ***
***         欢迎登录银行系统           ***
***                                   ***
***                                   ***
*****************************************
请输入管理员账户：1
请输入密码：1
操作成功，请稍后......
*********************************************
***                                     ***
***    1.开户(1)              2.查询(2)   ***
***    3.取款(3)              4.存款(4)   ***
```

```
***    5.转账(5)              6.锁定(6)   ***
***    7.解锁(7)                          ***
***                                      ***
***    退出(Q)                            ***
***                                      ***
**********************************************
请输入您的操作：
```

开户功能的运行结果如下所示：

```
请输入您的操作：1
请输入姓名:小明
请输入身份证号:123456789
请输入手机号:188888
请输入预存金额:500
请输入密码：000000
请再次输入密码：000000
密码设置成功，请牢记密码： 000000
您的开户已完成，请牢记开户账号：836095
```

存款功能的运行结果如下所示：

```
请输入您的操作：4
请输入您的卡号：836095
请输入密码：000000
账户：836095   余额：500.00
请输入要存入得金额:100000
存款成功！  账户：836095   余额：100500.00
```

取款功能的运行结果如下所示：

```
请输入您的操作：3
请输入您的卡号：836095
请输入密码：000000
账户：836095   余额：100500.00
请输入取款金额:5000
取款成功！  账户：836095   余额：95500.00
```