## 分析一个水杯的属性和功能，使用类描述并创建对象

高度，容积，颜色，材质

能存放液体

|  |
| --- |
| *class* Cup():  \_\_high = ""  \_\_volume = ""  \_\_colour = ""  \_\_texture = ""  *def* setHigh(*self*,high):  *self*.\_\_high = high  *def* getHigh(*self*):  *return self*.\_\_high  *def* setVolume(*self*,volume):  *self*.\_\_volume = volume  *def* getVolume(*self*):  *return self*.\_\_volume  *def* setColour(*self*,colour):  *self*.\_\_colour = colour  *def* getColour(*self*):  *return self*.\_\_colour  *def* setTexture(*self*,texture):  *self*.\_\_texture = texture  *def* getTextture(*self*):  *return self*.\_\_texture   *def* StoreLiquid(*self*,liquid):  *print*("杯子存放了",*self*.\_\_volume,"毫升的",liquid)  c = Cup() c.setHigh(100) c.setVolume(500) c.setColour("黑色") c.setTexture("玻璃") c.StoreLiquid("水") |

## 有笔记本电脑（屏幕大小，价格，cpu型号，内存大小，待机时长），行为（打字，打游戏，看视频）

|  |
| --- |
| *class* Computer():  \_\_size = ""  \_\_pice = ""  \_\_cpu = ""  \_\_memory = ""  \_\_time = ""  *def* setSize(*self*,size):  *self*.\_\_size = size  *def* getSize(*self*):  *return self*.\_\_size  *def* setPice(*self*,pice):  *self*.\_\_pice = pice  *def* getPice(*self*):  *return self*.\_\_pice  *def* setCpu(*self*,cpu):  *self*.\_\_cpu = cpu  *def* getCpu(*self*):  *return self*.\_\_cpu  *def* setMemory(*self*,memory):  *self*.\_\_memory = memory  *def* getMemory(*self*):  *return self*.\_\_memory  *def* setTime(*self*,time):  *self*.\_\_time = time  *def* getTime(*self*):  *return self*.\_\_time   *def* typing(*self*,words):  *print*("正在输入",words)  *def* playing(*self*,game):  *print*("正在玩",game)  *def* watching(*self*,video):  *print*("正在看",video)  c = Computer() c.setCpu("i7") c.setPice("8888") c.setSize("15.5英寸") c.setMemory("16GB") c.setTime("1小时") c.playing("炉石传说") c.watching("123") c.watching("绝命毒师") |

### 先构思面向对象版的中国工商银行系统

main.py

*import* random  
*from* day09\_1.DBUtils *import* select  
*from* day09\_1.address *import* Address  
*from* day09\_1.user *import* User  
*from* day09\_1.bank *import* Bank  
*from* day09\_1.Interface *import* Interface  
  
*# 确定银行的开户名称*bank\_name = "中国工商银行昌平区回龙观支行"  
  
*# 开户逻辑  
def* adduser():  
 *# 生成账号： 8位随机  
 # u = User()  
 # a = Address()* string = "" *# 随机数缓冲  
 for* i *in range*(8): *# 循环8次取字符* string = string + "1234567890"[random.randint(0,9)] *# 拼接* account = string  
 *print*("账号为：",account)  
 username = *input*("请输入姓名：")  
 password = *input*("请输入密码：")  
 *print*("接下来输入地址信息：")  
 country = *input*("\t输入国家：")  
 province = *input*("\t输入省份：")  
 street = *input*("\t输入街道：")  
 door = *input*("\t输入门牌号：")  
 money = *input*("请初始化您的余额：")  
  
 b = Bank()  
 *# 调用银行的开户方法* s = b.bank\_adduser(account,username,password,country,province,street,door,money)  
  
 *if* s == 1:  
 *print*("开户成功！")  
 *print*("以下是您的开户个人信息：")  
 *print*("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
 *print*("账号：",account)  
 *print*("用户名：", username)  
 *print*("密码：\*\*\*\*\*\*")  
 *print*("国家：", country)  
 *print*("省份：", province)  
 *print*("街道：", street)  
 *print*("门牌号：", door)  
 *print*("账户余额：", money)  
 *print*("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*开户行地址：", bank\_name)  
  
 *elif* s == 2:  
 *print*("该用户已存在！")  
 *elif* s == 3:  
 *print*("对不起，该银行已满！请携带证件到其他银行办理！")  
  
  
  
  
  
*# 存款逻辑  
def* deposit():  
 b = Bank()  
 account = *input*("请输入账号：")  
 s = b.bank\_deposit(account)  
  
 *if* s *is True*:  
 sql = "select *\** from bank where 账号="+account  
 data = select(sql,[])  
 *for* d *in* data:  
 *if* d[0] == account:  
 *print*("存款成功，现有金额：", d[8])  
 *if* s *is False*:  
 *print*("账号不存在")  
  
  
  
*# 取钱逻辑  
def* withdraw():  
 b = Bank()  
 account = *input*("请输入账号：")  
 s = b.bank\_withdraw(account)  
 *if* s == 0:  
 sql = "select *\** from bank where 账号 = %s"  
 param = [account]  
 data = select(sql, param)  
 *for* d *in* data:  
 *if* d[0] == account:  
 *print*("取款成功，剩余：", d[8])  
 *elif* s == 1:  
 *print*("账号不存在")  
 *elif* s == 2:  
 *print*("密码不正确")  
 *elif* s == 3:  
 *print*("金额不足")  
  
  
  
*# 转账逻辑  
def* transfer():  
 b = Bank()  
 out\_account = *input*("请输入转出的账号：")  
 in\_account = *input*("请输入转入的账号：")  
  
 s = b.bank\_transfer(out\_account, in\_account)  
 *if* s == 0:  
 sql = "select *\** from bank where 账号=%s"  
 param = [out\_account]  
 param2 = [in\_account]  
 data = select(sql, param)  
 data2 = select(sql, param2)  
 *# 账号存在  
 for* c *in* data2:  
 *if* c[0] == in\_account:  
 *for* d *in* data:  
 *if* d[0] == out\_account:  
 *print*("转账成功")  
 *print*("转出账号剩余：", d[8])  
 *print*("转入账号剩余：", c[8])  
 *elif* s == 1:  
 *print*("转出或转入账号不存在")  
 *elif* s == 2:  
 *print*("密码不正确")  
 *elif* s == 3:  
 *print*("金额不足")  
  
  
  
*# 查询逻辑  
def* inquire():  
 b = Bank()  
 account = *input*("请输入账号")  
 b.bank\_inquire(account)  
  
  
*while True*: *# 一直循环的进入选项* I = Interface()  
 *print*(I.getInterface())  
 chose = *input*("请输入您的选项：")  
 *if* chose == "1": *# 判断是否是1* adduser() *# 开户  
 elif* chose == "2": *# 判断是否是2* deposit()  
 *elif* chose == "3": *# 判断是否是3* withdraw()  
 *elif* chose == "4": *# 判断输入的是否是4* transfer()  
 *elif* chose == "5": *# 判断输入的是否是5* inquire()  
 *elif* chose == "6": *# 判断输入的是否是6，若是6则需要退出 break  
 print*("拜拜了您嘞！")  
 *break  
 else*:  
 *print*("输入非法！重新输入！")

bank.py

*from* day09\_1.DBUtils *import* select  
*from* day09\_1.DBUtils *import* update  
*from* day09\_1.address *import* Address  
*from* day09\_1.user *import* User  
  
*class* Bank():  
 \_\_user = []  
 \_\_bankName = "中国工商银行昌平区回龙观支行"  
  
 *def* setUser(*self*,user):  
 *self*.\_\_user == user  
 *def* getUser(*self*):  
 *return self*.\_\_user  
 *def* getBankName(*self*):  
 *return self*.\_\_bankName  
  
 *# 银行的开户逻辑  
 def* bank\_adduser(*self*,account, username, password, country, province, street, door, money):  
 *# 判断数据库是否已满* b = Bank()  
 sql1 = "select *count*(*\**) from bank";  
 data = select(sql1, [])  
 *if* data[0][0] >= 100: *# 如果返回的统计数据超出100，则已满  
 return* 3  
 *# 判断数据是否存在改用户  
 # 获取所有键，然后在判断是否有* sql2 = "select *\** from bank where 账号 = %s"  
 param2 = [account]  
 data2 = select(sql2, param2)  
 *if len*(data2) != 0: *# 如果通过sql语句能查到数据并且不为空，则说明改用户已存在  
 return* 2  
 *# 正常开户：insert into 表 ，否则则执行存储数据操作* sql3 = "insert into bank values(%s,%s,%s,%s,%s,%s,%s,%s,%s)"  
 param3 = [account, username, password, country, province, street, door, b.getBankName(), money]  
 update(sql3, param3)  
 *return* 1  
  
 *# 银行的存款逻辑  
 def* bank\_deposit(*self*,account):  
 sql = "select *\** from bank where 账号 = %s"  
 param = [account]  
 data = select(sql, param)  
 *# 账号存在  
 for* d *in* data:  
 *if* d[0] == account:  
 money = *int*(*input*("请输入金额："))  
 money = money + *int*(d[8])  
 sql2 = "update bank set 余额='" + *str*(money) + "' where 账号=" + account  
 param2 = []  
 update(sql2, param2)  
 *return True  
  
 # 账号不存在  
 return False  
  
 # 银行的取钱逻辑  
 def* bank\_withdraw(*self*,account):  
 sql = "select *\** from bank where 账号 = %s"  
 param = [account]  
 data = select(sql, param)  
 *# 账号存在  
 for* d *in* data:  
 *if* d[0] == account:  
 password = *input*("请输入密码：")  
 *if* d[2] == password:  
 money = *int*(*input*("请输入金额："))  
 *if* money <= *int*(d[8]):  
 money = *int*(d[8]) - money  
 sql2 = "update bank set 余额='" + *str*(money) + "' where 账号=" + account  
 update(sql2, [])  
 *return* 0  
 *else*:  
 *return* 3  
 *else*:  
 *return* 2  
 *return* 1  
  
 *# 银行的转账逻辑  
 def* bank\_transfer(*self*,out\_account, in\_account):  
 sql = "select *\** from bank where 账号=%s"  
 param = [out\_account]  
 param2 = [in\_account]  
 data = select(sql, param)  
 data2 = select(sql, param2)  
 *# 账号存在  
 for* c *in* data2:  
 *if* c[0] == in\_account:  
 *for* d *in* data:  
 *if* d[0] == out\_account:  
 password = *input*("请输入密码：")  
 *if* d[2] == password:  
 money = *int*(*input*("请输入金额："))  
 *if* money <= *int*(d[8]):  
 money1 = *int*(d[8]) - money  
 money2 = *int*(c[8]) + money  
 sql2 = "update bank set 余额='" + *str*(money1) + "' where 账号=" + out\_account  
 sql3 = "update bank set 余额='" + *str*(money2) + "' where 账号=" + in\_account  
 update(sql2, [])  
 update(sql3, [])  
 *return* 0  
 *else*:  
 *return* 3  
 *else*:  
 *return* 2  
  
 *return* 1  
  
 *# 银行的查询逻辑  
 def* bank\_inquire(*self*,account):  
 sql = "select *\** from bank where 账号 = %s"  
 param = [account]  
 data = select(sql, param)  
 *# 账号存在* i = 0  
 *for* c *in* data:  
 *if* c[0] == account:  
 i = i + 1  
 password = *input*("请输入密码：")  
 *if* password == c[2]:  
 *print*("前账号：", account)  
 *print*("密码：\*\*\*\*\*\*")  
 *print*("余额：", c[8])  
 *print*("用户居住地址：", c[3], c[4], c[5], c[6])  
 *print*("当前账户的开户行：", c[7])  
 *else*:  
 *print*("密码不正确")  
 *if* i == 0:  
 *print*("账户不存在")

user.py

*class* User:  
 \_\_account = ""  
 \_\_username = ""  
 \_\_password = ""  
 \_\_money = ""  
 \_\_address = []  
 \_\_time = ""  
 \_\_bankName = ""  
  
 *def \_\_init\_\_*(*self*,account, username, password, money, time, bank, address):  
 *self*.\_\_account = account  
 *self*.\_\_password = password  
 *self*.\_\_username = username  
 *self*.\_\_money = money  
 *self*.\_\_time = time  
 *self*.\_\_bank\_name = bank  
 *self*.\_\_address = address  
  
 *def* setAccount(*self*,account):  
 *self*.\_\_account = account  
 *def* getAccount(*self*):  
 *return self*.\_\_account  
 *def* setUsername(*self*,username):  
 *self*.\_\_username = username  
 *def* getUsername(*self*):  
 *return self*.\_\_username  
 *def* setPassword(*self*,password):  
 *self*.\_\_password = password  
 *def* getPassword(*self*):  
 *return self*.\_\_password  
 *def* setMoney(*self*,money):  
 *self*.\_\_money = money  
 *def* getMoney(*self*):  
 *return self*.\_\_money  
 *def* setTime(*self*,time):  
 *self*.\_\_time = time  
 *def* getTime(*self*):  
 *return self*.\_\_time  
 *def* setBankname(*self*,bank):  
 *self*.\_\_bank\_name = bank  
 *def* getBankname(*self*):  
 *return self*.\_\_bank\_name  
 *def* setAddress(*self*,address):  
 *self*.\_\_address = address  
 *def* getAddress(*self*):  
 *return self*.\_\_address

address.py

*class* Address:  
 \_\_country = *None* \_\_province = *None* \_\_street = *None* \_\_door = *None  
  
 def \_\_init\_\_*(*self*,country, province, street, door):  
 *self*.\_\_country = country  
 *self*.\_\_province = province  
 *self*.\_\_street = street  
 *self*.\_\_door = door  
  
 *def* setCountry(*self*,country):  
 *self*.\_\_country = country  
 *def* getCountry(*self*):  
 *return self*.\_\_country  
 *def* setProvince(*self*,province):  
 *self*.\_\_province = province  
 *def* getProvince(*self*):  
 *return self*.\_\_province  
 *def* setStreet(*self*,street):  
 *self*.\_\_street = street  
 *def* getStreet(*self*):  
 *return self*.\_\_street  
 *def* setDoor(*self*,door):  
 *self*.\_\_door = door  
 *def* getDoor(*self*):  
 *return self*.\_\_door

interface.py

*class* Interface:  
 \_\_interface = '''  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* 中国工商银行账户管理系统 \*  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* 1.开户 \*  
 \* 2.存钱 \*  
 \* 3.取钱 \*  
 \* 4.转账 \*  
 \* 5.查询 \*  
 \* 6.退出 \*  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 '''  
  
 *def* getInterface(*self*):  
 *return self*.\_\_interface

DBUtils.py

*import* pymysql  
host = "localhost"  
database="gsyh"  
user="root"  
password="root"  
  
*# 可以处理增，删，改的所有操作  
def* update(sql,param):  
 con = pymysql.connect(host=host,database=database,user=user,password=password)  
 cursor = con.cursor()  
 cursor.execute(sql,param)  
 con.commit() *# 提交数据* cursor.close()  
 con.close()  
  
*# 可以处理查询所有炒作  
def* select(sql,param,mode="many",size=0):  
 con = pymysql.connect(host=host, database=database, user=user, password=password)  
 cursor = con.cursor()  
 cursor.execute(sql, param)  
 con.commit() *# 提交数据  
 if* mode == "all":  
 *return* cursor.fetchall()  
 *elif* mode == "one":  
 *return* cursor.fetchone()  
 *elif* mode == "many":  
 *return* cursor.fetchmany(size)  
 cursor.close()  
 con.close()