武汉纺织大学

Web应用开发课程设计

**基于ssm的银行管理系统**

**学 院： 数学与计算机学院**

**班 级： 物联网11801**

**姓 名： 江程**

**学 号： 1804240428**

**指导老师： 聂刚**

**成 绩：**

**完成日期： 2020年12月17日**

# 1 需求分析

设计一个银行ATM系统，实现“取款”、“转账”、“存款”、“查询”、“密码修改”、“显示余额”等功能，具体要求如下：

## 1.1用户登陆

当程序运行时，\*\*\*\*\*\*\*，您好”，提示用户依次输入用户名及其密码，当用户名及密码不正确时，显示提示信息“用户名或密码不正确，请重新输入”，**用户最多可以尝试3次，若输入3次均不正确，则给出提示信息“最多只能尝试3次”**，退出系统；

## 1.2主菜单

当用户名及密码正确时，进入系统主界面，如下：

\*\*\*\*\*，您好！

账号：\*\*\*\*\*

存款 取款 查询 转账 改密 销户

分别点击对应的按键进入对应的功能。

## 1.3各项功能

### 1.3.1 存款

当选择按键1时，进入存款选项：

存款：\_\_\_\_\_\_\_\_\_\_\_\_\_\_

提交

### 1.3.2 取款

当选择2时，进入取款选项

取款：\_\_\_\_\_\_\_\_\_\_\_\_\_\_

提交

### 1.3.3 查询

当选择3时进入查询选项

1 \*\* \*\*\*\*\*\* \*\* \*\*\*\*\*\*

2 \*\* \*\*\*\*\*\* \*\* \*\*\*\*\*\*

### 1.3.4 转账

选择4时进入转账选项：

输入对方账号：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

转账金额：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

提交

### 1.3.5 修改密码

当选择5时，显示如下子菜单：

输入您的身份证号：\_\_\_\_\_\_\_\_\_\_\_\_\_\_

输入您的手机号：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

提交

当验证成功后显示如下菜单，更改新的密码：

输入新的密码：\_\_\_\_\_\_\_\_\_\_\_\_\_\_

提交

### 1.3.6 注册

当用户没有账号时将跳转到用户注册界面，当用户注册成功，将跳转到登陆界面：

### 账号：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### 密码：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### 姓名：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### 性别：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### 提交

### 提交1.3.7 销户

用户可以选择销除账号，输入销户账号，销户成功后会提示删除成功

# **2 系统设计**

## 2.1用户用例图

系统用户用例如下图2-1所示：

存款

取款

用户

查询

转账

改密

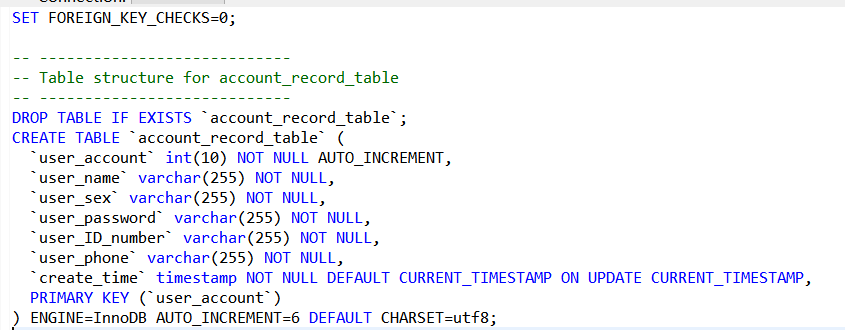
注册

图2-1 用户用例图

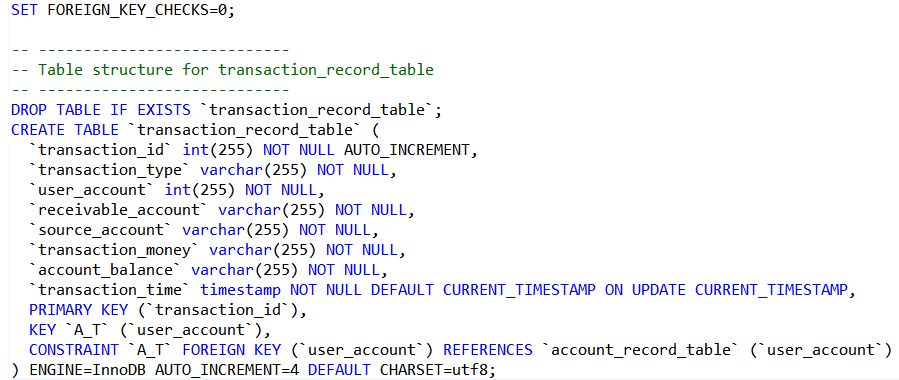
## 2.2 数据库

数据库设计2张表，用户表，转账表

Account\_record\_table.sql



Transaction\_record\_table.sql



## 2.3 UML类图（Class Diagram）

对于用户功能模块，共设计如下6个类。

* 数据库连接类DatabaseConnection：连接数据库并获取连接对象，关闭连接对象。
* VO实体类User：与数据库结构进行映射的类。主要由属性，setter, getter方法组成，VO类中的属性与表中的字段相对应，每一个VO类的对象都表示表中的每一条记录
* DAO接口IUserDAO：主要定义操作的接口，定义一系列数据库的原子性操作，例如增删改查（通常称为CRUD）等。
* DAO实现类UserDAOImpl：DAO接口的真实实现类，主要完成具体数据库操作，但不负责数据库的打开和关闭。
* DAO工厂类DAOFactory：通过工厂类取得一个DAO的实例化对象
* 业务逻辑实现类UserService：对于数据层的原子操作进行整合。还要负责数据库的打开与关闭（不管是否出异常，数据库都要关闭）
* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*：

各类的结构及类之间的关系如图2-5所示：

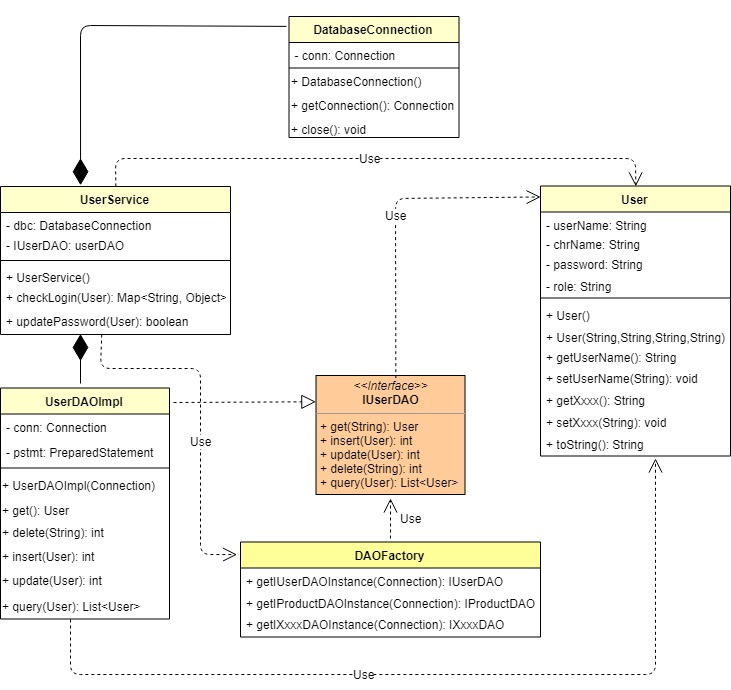
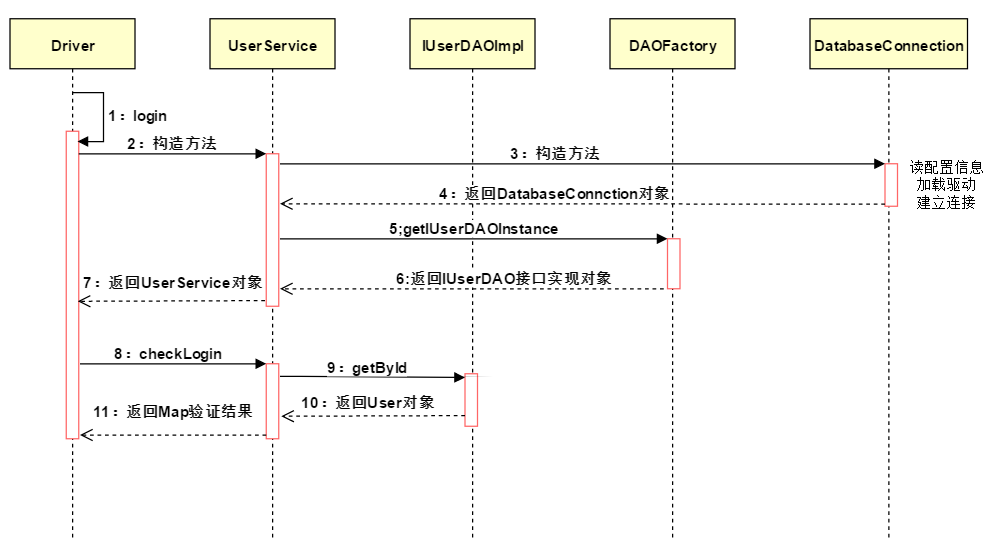


图2-5 用户功能类图

## 2.4 UML时序图（Sequence Diagram）

### 2.4.1 系统登录



## 2.5 UML活动图（Activity Diagram）

### 2.5.1 登陆

登录操作需要验证用户名，密码，只有当用户名和密码同时存在即用户存在的情况下才能成功登录系统，否则会提示错误信息，具体流程如图2-6所示。

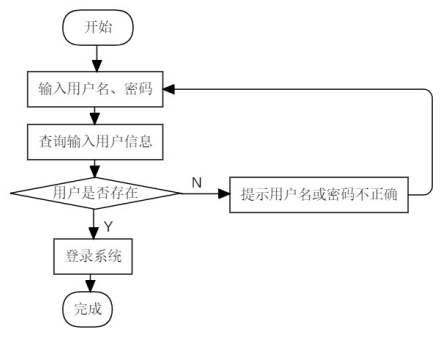


图2-2 登录流程图

### 2.5.2 存款

存款功能的活动图如图2-5所示。

存款时的账号

存款时输入的数值

查询存款余额

输入数值加入余额内

图2-5 功能活动图

### 2.5.3 取款

存款功能的活动图如图2-6所示。

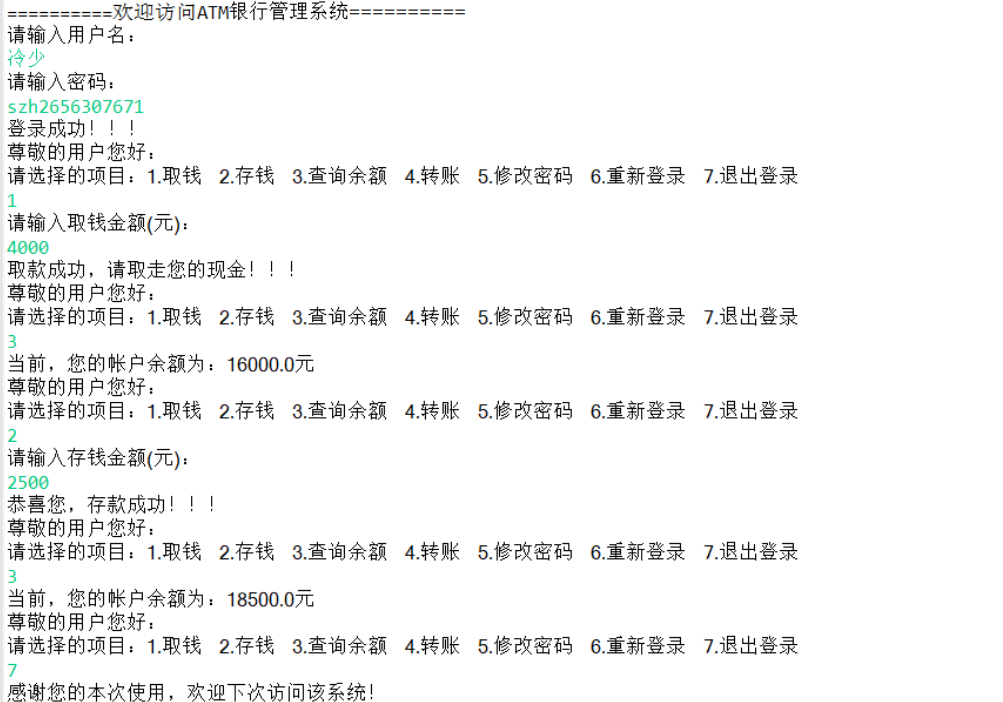
取款时的账号

取款时输入的数值

### 

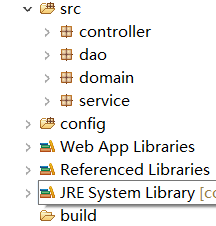
查询存款余额

存款减去输入的数值



# **3 系统实现**

## 3.1 项目结构



## 3.2 配置文件

### 3.2.1 jdbc.properties文件

该配置文件主要是以文件形式保存数据库的驱动类名称，连接数据库的URL地址，访问数据库的用户名及对应的密码，程序运行时会读取该文件相关信息，避免硬编码，当相关信息发生变化时，只需修改配置文件而不用修改源代码，增加程序的可扩展型。

driver=com.mysql.jdbc.Driver

url=jdbc:mysql://localhost:3306/atm?useUnicode=true&characterEncoding=utf-8

username=root

password=000000

#定义初始连接数

initialSize=0

#定义最大连接数

maxActive=20

#定义最小空闲

minIdle=1

#定义最长等待时间

maxWait=60000

#验证连接是否可用，使用的SQL语句

validationQuery =SELECT 1

### 3.2.2 log4j.properties文件

描述该文件作用，并对里面的配置进行解释

</Console>

<RollingFile name=*"RollingFile"* filename=*"log/ZycTest.log"*

filepattern=*"${logPath}/%d{YYYYMMddHHmmss}-fargo.log"*>

<PatternLayout pattern=*"%d{YYYY-MM-dd HH:mm:ss} %-5p [%c,%L] - %msg%n"* />

<Policies>

<SizeBasedTriggeringPolicy size=*"100 MB"* />

</Policies>

<DefaultRolloverStrategy max=*"20"* />

</RollingFile>

</Appenders>

<!--定义logger，只有定义了logger并引入的appender，appender才会生效 -->

<Loggers>

<!-- Logger节点用来单独指定日志的形式，可以为通过name属性设置指定包下的class指定不同的日志级别等 ,

可以设置Logger的additivity="false"只在自定义的Appender中进行输出 -->

<logger name=*"dao"* level=*"debug"* additivity=*"false"*>

<appender-ref ref=*"Console"* />

</logger>

<!-- Root节点用来指定项目的根日志，如果没有单独指定Logger，那么就会默认使用该Root日志输出 -->

<Root level=*"info"*>

<AppenderRef ref=*"Console"* />

<AppenderRef ref=*"RollingFile"* />

</Root>

</Loggers>

</Configuration>

### 3.2.3 mybatis-config.xml文件

描述该文件作用，并对里面的配置进行解释<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE configuration

PUBLIC "-//mybatis.org//DTD Config 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-config.dtd">

<!-- 根标签 -->

<configuration>

<!-- 指定 资源文件 -->

<properties resource="jdbc.properties"></properties>

<!-- 指定 MyBatis 所用日志的具体实现，未指定时将自动查找-->

<settings>

<setting name="logImpl" value="LOG4J" />

</settings>

<!-- 环境，可以配置多个，default：指定采用哪个环境 -->

<environments default="test">

<!-- id：唯一标识 -->

<environment id="test">

<!-- 事务管理器，JDBC类型的事务管理器 -->

<transactionManager type="JDBC" />

<!-- 数据源，池类型的数据源 -->

<dataSource type="POOLED">

<property name="driver" value="${DBDRIVER}" />

<property name="url" value="${DBURL}" />

<property name="username" value="${DBUSER}" />

<property name="password" value="${PASSWORD}" />

</dataSource>

</environment>

</environments>

<!-- 配置mapper映射文件 -->

<mappers>

<mapper resource="dao/mapper/UserMapper.xml" />

</mappers>

</configuration>

## 3.3 Controller controller.java

AccountController.java

**package** vo;

**public** **class** User {

**private** String id;

**private** String userName;

**private** String password;

**private** **double** money;

**private** **boolean** locker;

**public** User(String id, String userName, String password, **double** money, **boolean** locker) {

**super**();

**this**.id = id;

**this**.userName = userName;

**this**.password = password;

**this**.money = money;

**this**.locker = locker;

}

**public** User() {

**super**();

}

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** String getUserName() {

**return** userName;

}

**public** **void** setUserName(String userName) {

**this**.userName = userName;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

**public** **double** getMoney() {

**return** money;

}

**public** **void** setMoney(**double** money) {

**this**.money = money;

}

**public** **boolean** getLocker() {

**return** locker;

}

**public** **void** setLocker(**boolean** locker) {

**this**.locker = locker;

}

@Override

**public** String toString() {

**return** "User [id=" + id + ",userName=" + userName + ", password=" + password + ", ye=" + money + ", status=" + locker + "]";

}

## 3.4 DAO接口类AccountDao.java

public interface AccountDao {

// 用于用户登录

public Account selectUserAccount(@Param("userAccount") String userAccount,

@Param("userPassword") String userPassword);

// @param表示上面sql中的参数

// 查询账户余额

public AccountRecord selectUserAccountBalanceDao(@Param("userAccount") String userAccount);

// 用于用户添加

int insertUserAccount(Account account);

// 用于用户存款

int userAccountSave\_Dao(@Param("userAccount") String userAccount, @Param("transactionMoney") Float transactionMoney,

@Param("accountBalance") Float accountBalance);

// 用于用户取款

int userAccountDraw\_Dao(@Param("userAccount") String userAccount, @Param("transactionMoney") Float transactionMoney,

@Param("accountBalance") Float accountBalance);

// 用于用户转账

int userAccountTransfer\_Dao(@Param("userAccount") String userAccount,

@Param("receivableAccount") String receivableAccount, @Param("transactionMoney") Float transactionMoney,

@Param("accountBalance") Float accountBalance);

// 查询收款账号存不存在

int selectReceivablesAccount\_Dao(@Param("receivableAccount") String receivableAccount);

// 查询收款账户余额

public AccountRecord selectTransferAccountBalanceDao(@Param("receivableAccount") String receivableAccount);

// 用于用户收款

int userAccountReceivables\_Dao(@Param("receivableAccount") String receivableAccount, @Param("userAccount") String sourceAccount,

@Param("transactionMoney") Float transactionMoney, @Param("accountBalance") Float accountBalance);

// 查询身份证号和手机号码

public Account select\_UserIdNumber\_UserPhone\_Dao(@Param("userAccount") String userAccount);

// 用于用户修改密码

int updateAccountPassword\_Dao(@Param("userAccount") String userAccount, @Param("userPassword") String userPassword);

// 删除用户

void deleteUserAccount\_Dao(@Param("userAccount") String userAccount);

// 删除用户 记录

void deleteUserAccountRecord\_Dao(@Param("userAccount") String userAccount);

}

## 3.5 接口映射文件bankCloudServices-servlet.xml

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE mapper

PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace=*"dao.IUserDAO"*>

<select id=*"get"* resultType=*"vo.User"*>

select \* from user where id = #{id}

</select>

<update id=*"updatePassword"*>

update user set password=#{password} where id=#{id}

</update>

<update id=*"updateMoney"*>

update user set money=#{money} where id=#{id}

</update>

<update id=*"lock"*>

update user set locker=#{lock} where id=#{id}

</update>

<update id=*"inMoney"*>

update user set money=money+#{money} where id=#{id}

</update>

<update id=*"deMoney"*>

update user set money=money-#{money} where id=#{id}

</update>

</mapper>

### 3.6.Do

public class Account {

private String userAccount;// 用户账号

private String userPassword;// 用户密码

private String accountName; // 用户姓名

private String accountSex; // 用户性别

private String userIdNumber;// 用户身份证

private String userPhone;// 用户电话

private Date creationTime;// 创建时间

private List<AccountRecord> accountRecordList;

public Account() {

}

public Account(String userAccount, String userPassword, String accountName, String accountSex, String userIdNumber,

String userPhone, Date creationTime, List<AccountRecord> accountRecordList) {

super();

this.userAccount = userAccount;

this.userPassword = userPassword;

this.accountName = accountName;

this.accountSex = accountSex;

this.userIdNumber = userIdNumber;

this.userPhone = userPhone;

this.creationTime = creationTime;

this.accountRecordList = accountRecordList;

}

public List<AccountRecord> getAccountRecordList() {

return accountRecordList;

}

public void setAccountRecordList(List<AccountRecord> accountRecordList) {

this.accountRecordList = accountRecordList;

}

public String getUserAccount() {

return userAccount;

}

public void setUserAccount(String userAccount) {

this.userAccount = userAccount;

}

public String getUserPassword() {

return userPassword;

}

public void setUserPassword(String userPassword) {

this.userPassword = userPassword;

}

public String getAccountName() {

return accountName;

}

public void setAccountName(String accountName) {

this.accountName = accountName;

}

public String getAccountSex() {

return accountSex;

}

public void setAccountSex(String accountSex) {

this.accountSex = accountSex;

}

public String getUserIdNumber() {

return userIdNumber;

}

public void setUserIdNumber(String userIdNumber) {

this.userIdNumber = userIdNumber;

}

public String getUserPhone() {

return userPhone;

}

public void setUserPhone(String userPhone) {

this.userPhone = userPhone;

}

public Date getCreationTime() {

return creationTime;

}

public void setCreationTime(Date creationTime) {

this.creationTime = creationTime;

}

@Override

public String toString() {

return "Account [userAccount=" + userAccount + ", userPassword=" + userPassword + ", accountName=" + accountName

+ ", accountSex=" + accountSex + ", userIdNumber=" + userIdNumber + ", userPhone=" + userPhone

+ ", creationTime=" + creationTime + ", accountRecordList=" + accountRecordList + "]";

}

}

3.6.2 AccountRecord.java

public class AccountRecord {

private String transactionId;// 交易ID

private String transactionType;// 交易类型

private String userAccount; // 交易账号

private String receivableAccount; // 收款账号

private String sourceAccount;// 来源账号

private float transactionMoney;// 交易钱数

private float accountBalance;// 账户余额

private Date transactionTime;// 交易时间

public AccountRecord() {

}

public String getTransactionId() {

return transactionId;

}

public void setTransactionId(String transactionId) {

this.transactionId = transactionId;

}

public String getTransactionType() {

return transactionType;

}

public void setTransactionType(String transactionType) {

this.transactionType = transactionType;

}

public String getTransactionAccount() {

return userAccount;

}

public void setTransactionAccount(String transactionAccount) {

this.userAccount = transactionAccount;

}

public String getReceivableAccount() {

return receivableAccount;

}

public void setReceivableAccount(String receivableAccount) {

this.receivableAccount = receivableAccount;

}

public String getSourceAccount() {

return sourceAccount;

}

public void setSourceAccount(String sourceAccount) {

this.sourceAccount = sourceAccount;

}

public float getTransactionMoney() {

return transactionMoney;

}

public void setTransactionMoney(float transactionMoney) {

this.transactionMoney = transactionMoney;

}

public float getAccountBalance() {

return accountBalance;

}

public void setAccountBalance(float accountBalance) {

this.accountBalance = accountBalance;

}

public Date getTransactionTime() {

return transactionTime;

}

public void setTransactionTime(Date transactionTime) {

this.transactionTime = transactionTime;

}

@Override

public String toString() {

return "AccountRecord [transactionId=" + transactionId + ", transactionType=" + transactionType

+ ", userAccount=" + userAccount + ", receivableAccount=" + receivableAccount + ", sourceAccount="

+ sourceAccount + ", transactionMoney=" + transactionMoney + ", accountBalance=" + accountBalance

+ ", transactionTime=" + transactionTime + "]";

}

}

## 3.7 服务层Service.java

AccountService.java

public interface AccountService {

// 用户登录

public Account accountLoginService(String userAccount, String userPassword);

// 添加用户

boolean insertUserAccount(Account account);

// 查询账号余额

public float selectUserAccountBalanceService(String userAccount);

// 用户存款

boolean userAccountSave\_Service(String userAccount, Float transactionMoney, Float accountBalance);

// 用户取款

boolean userAccountDraw\_Service(String userAccount, Float transactionMoney, Float accountBalance);

// 用户转账

boolean userAccountTransfer\_Service(String userAccount, String receivableAccount, Float transactionMoney,

Float accountBalance);

// 查询收款账号存不存在

boolean selectReceivablesAccount\_Service(String receivableAccount);

// 查询收款账户余额

public float selectTransferAccountBalanceService(String receivableAccount);

// 用户收款

boolean userAccountReceivables\_Service(String receivableAccount, String sourceAccount, Float transactionMoney,

Float accountBalance);

// 查询身份证号和手机号码

public Account select\_UserIdNumber\_UserPhone\_Service(String userAccount);

// 用户修改密码

int updateAccountPassword\_Service(String userAccount, String userPassword);

// 删除用户

void deleteUserAccount\_Service(String userAccount);

}

AccountServiceImpl.java

@Service("accountService")

public class AccountServiceImpl implements AccountService {

private Logger logger = Logger.getLogger(AccountServiceImpl.class);

float money;

float smoney;

@Autowired

private AccountDao accountDao;

// 用户登录

@Override

public Account accountLoginService(String userAccount, String userPassword) {

Account account = accountDao.selectUserAccount(userAccount, userPassword);

System.out.println("登录时的AccountServiceImpl查询---------->" + account);

return account;

}

// 用户注册

@Override

public boolean insertUserAccount(Account account) {

int resultRow = accountDao.insertUserAccount(account);

logger.info("插入的行：" + resultRow);

return resultRow > 0;

}

// 查询账户余额

@Override

public float selectUserAccountBalanceService(String userAccount) {

AccountRecord accountRecord = accountDao.selectUserAccountBalanceDao(userAccount);

if (accountRecord == null) {

return 0;

}

money = accountRecord.getAccountBalance();

logger.info("查询到的账户余额----->" + money);

return money;

}

// 用户存款

@Override

public boolean userAccountSave\_Service(String userAccount, Float transactionMoney, Float accountBalance) {

logger.info("存款之前查询到的账户余额----->" + money);

accountBalance = money + transactionMoney;

logger.info("相加完以后的钱：" + accountBalance);

int c = accountDao.userAccountSave\_Dao(userAccount, transactionMoney, accountBalance);

if (c == 1) {

return true;

}

return false;

}

// 用户取款

@Override

public boolean userAccountDraw\_Service(String userAccount, Float transactionMoney, Float accountBalance) {

logger.info("取款之前查询到的账户余额----->" + money);

accountBalance = money - transactionMoney;

logger.info("取款完以后剩的钱：" + accountBalance);

int c = accountDao.userAccountDraw\_Dao(userAccount, transactionMoney, accountBalance);

if (c == 1) {

return true;

}

return false;

}

// 用户转账

@Override

public boolean userAccountTransfer\_Service(String userAccount, String receivableAccount, Float transactionMoney,

Float accountBalance) {

logger.info("转账之前查询到的账户余额----->" + money);

accountBalance = money - transactionMoney;

logger.info("转账完以后剩的钱：" + accountBalance);

int c = accountDao.userAccountTransfer\_Dao(userAccount, receivableAccount, transactionMoney, accountBalance);

if (c == 1) {

return true;

}

return false;

}

// 查询收款账号存不存在

@Override

public boolean selectReceivablesAccount\_Service(String receivableAccount) {

int c = accountDao.selectReceivablesAccount\_Dao(receivableAccount);

logger.info("转账时的账号状态：" + c);

if (c > 0) {

return true;

}

return false;

}

// 收款账户的余额查询

@Override

public float selectTransferAccountBalanceService(String receivableAccount) {

AccountRecord accountRecord = accountDao.selectTransferAccountBalanceDao(receivableAccount);

if (accountRecord == null) {

return 0;

}

smoney = accountRecord.getAccountBalance();

logger.info("收款账户查询到的余额----->" + smoney);

return smoney;

}

// 用户收款

@Override

public boolean userAccountReceivables\_Service(String receivableAccount, String sourceAccount,

Float transactionMoney, Float accountBalance) {

logger.info("收款之前查询到的账户余额----->" + smoney);

accountBalance = smoney + transactionMoney;

logger.info("收款完以后的钱：" + accountBalance);

int c = accountDao.userAccountReceivables\_Dao(receivableAccount, sourceAccount, transactionMoney,

accountBalance);

if (c == 1) {

return true;

}

return false;

}

// 查询身份证号和手机号码

@Override

public Account select\_UserIdNumber\_UserPhone\_Service(String userAccount) {

Account account = accountDao.select\_UserIdNumber\_UserPhone\_Dao(userAccount);

logger.info("身份证号是：" + account.getUserIdNumber());

logger.info("手机号码是：" + account.getUserPhone());

logger.info("account------>" + account);

return account;

}

// 修改用户密码

@Override

public int updateAccountPassword\_Service(String userAccount, String userPassword) {

return accountDao.updateAccountPassword\_Dao(userAccount, userPassword);

}

// 删除用户

@Override

public void deleteUserAccount\_Service(String userAccount) {

accountDao.deleteUserAccount\_Dao(userAccount);

accountDao.deleteUserAccountRecord\_Dao(userAccount);

}

}

# **5 系统总结**

这次课程设计我做的是银行管理系统。系统的基本功能都已实现， 但还是存在系统不稳定等多个问题尚待解决。这次实验参考了很多资料，一个系统要细分做好这么多模块很难。通过这次实验，我也意识到了实践与课堂的紧密关联，很多未知的东西需要不断去探索，改进才能熟练掌握。