Java 实验报告

徐云凯 1713667

已编译的 JavaDoc

目录

问题分析描述

项目基于嗖嗖移动业务大厅 1.0 改进而来。要求在所有的数据访问上使用数据库实现，并在数据库的调用中使用 DAO 模式分离数据库操作与业务逻辑。

实验环境

Java 版本：java version "11.0.2" 2019-01-15 LTS, Java(TM) SE Runtime Environment 18.9 (build 11.0.2+9-LTS)。数据库：腾讯云服务器，mysql Ver 14.14 Distrib 5.7.27, for Linux (x86_64).
IDE: IntelliJ IDEA

实现思路

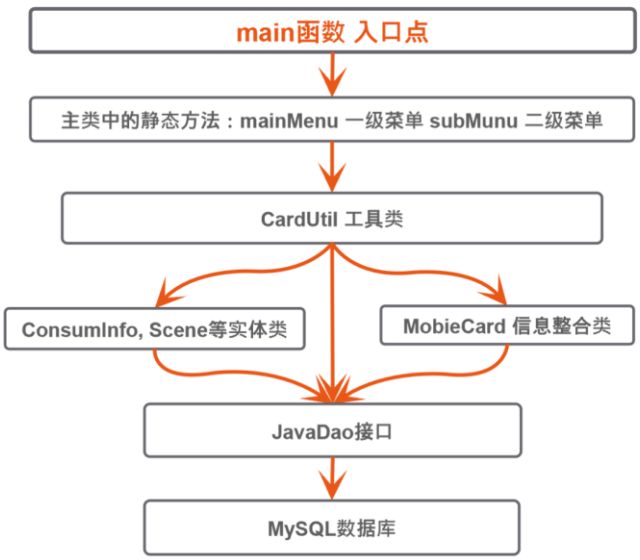
项目中数据流主要涉及电话卡信息、通话记录与套餐信息，故数据库中设置三个表，本别存储。

DAO 除了提供基本的增删改查等功能外再对诸如用户登录验证等功能提供额外端口，无需从数据库还原该用户全部信息，只验证卡号密码。

将前一次的系统结构进行重构，把三个接口中的功能合并到同一个功能类 ServicePackage 中，防止冗余。同时彻底分离主菜单与工具类的代码调用，实现完全的模块化。

具体实现

一． 整体结构



故依据以上要求，项目各个类与接口结构如下：

程序包com.sosoMobie的分层结构

类分层结构

- java.lang.Object
  - com.sosoMobie.**CardUtil**
  - com.sosoMobie.**ConsumInfo**
  - com.sosoMobie.**Main**
  - com.sosoMobie.**MobileCard**
  - com.sosoMobie.**Scene**
  - com.sosoMobie.**ServicePackage**
  - com.sosoMobie.**SosoDaoUtil**
    - com.sosoMobie.**SosoDaoMySQL** (implements com.sosoMobie.SosoDao)
  - java.lang.Throwable (implements java.io.Serializable)
    - java.lang.Exception
      - com.sosoMobie.**InsufficientBalanceException**

接口分层结构

- com.sosoMobie.**SosoDao**

其中，除了在项目中要求的几个类之外，自定义了 InsufficientBalanceException 异常类，用于在处理消费场景时可能遇到的余额不足异常。

二. 各功能实现

（按项目中菜单项次序排列）

1. 一级二级菜单

1.0. 使用 do...while 循环反复执行对菜单项目的选择

内部使用 switch-case 语句选择选项执行，对于非法输入使用 try-catch(InputMismatchException e)处理，并新建 scanner 对象以刷新输入缓冲区

```
private static void mainMenu(CardUtil cardUtil){
    Scanner scanner = new Scanner(System.in);
    int select = -1;

    do{
        System.out.print("\n*******************欢迎使用嗖嗖移动业务大厅*******************\n" +
                "1.用户登录 2.用户注册 3.使用嗖嗖 4.话费充值 5.资费说明 6.退出系统\n" +
                "请选择：");
        try {
            select = scanner.nextInt();
        }
        catch(InputMismatchException e){
            scanner = new Scanner(System.in);
            System.out.println("请输入正确的数字");
        }

        switch (select)
        {...}
    } while(true);  //循环执行主要方法体
}
```

1.1. 一级菜单

分类别调用工具类函数

```
switch (select)
{
    case 1:
        String card = cardUtil.singnedIn();
        if (card != null) {
            subMenu1(cardUtil, card);
        }
        break;

    case 2:
        cardUtil.addCard();
        break;

    case 3:
        cardUtil.useSoso();
        break;

    case 4:
        cardUtil.chargeMoney();
        break;

    case 5:
        cardUtil.showDescription();
        break;

    case 6:
        System.out.println("感谢使用嗖嗖移动业务大厅"); //退出
        return;

    default:
        System.out.println("请在1-6中做出选择");    //对于任意其他输入，显示提示信息后重新输入
}
while(true);  //循环执行主要方法体
```

1.2. 二级菜单

```
    switch (select)
    {
        case 1:
            cardUtil.showAmountDetail(card);
            break;

        case 2:
            cardUtil.showRemainDetail(card);
            break;

        case 3:
            cardUtil.printConsumInfo(card);
            break;

        case 4:
            cardUtil.changingPack(card);
            break;

        case 5:
            cardUtil.delCard(card);
            return;

        default:
            return;
    }
} while(true);
```

2. 工具类

2.0. 构造函数与初始化

构造函数通过接口新建 DAO 接口的 MySQL 实现。构造函数之后调用 initScenes() 函数初始化 List<Scene> scenes 完成对场景的初始化

```
public CardUtil() {
    sosoDAO = new SosoDaoMySQL();
    initScenes();
}
```

2.1. 登录验证

调用 CardUtil 类的 isexitCard 方法验证

```
public String singnedIn() {
    Scanner scanner = new Scanner(System.in);
    String card, password;
    boolean flag;

    System.out.print("请输入手机卡号：");
    card = scanner.next();
    flag = isExistCard(card);
    if (flag) {   //先检查卡号是否存在，验证存在后再输入密码
        System.out.print("请输入密码：");
        password = scanner.next();
        flag = isExistCard(card, password);
        if (flag) {   //再检查密码是否正确
            System.out.println("登录成功");
            return card;
        }
        else {
            System.out.println("密码输入错误，请重试");
        }
    }
    else {
        System.out.println("手机卡号不存在，请重试");
    }
    return null;
}
```

2.2. 判断卡号是否存在及卡号密码是否正确

两个函数功能接近，使用函数重载。前者直接使用 DAO 接口中预留的 sosoDAO.isCardExit(number)方法查询卡号是否存在，后者使用重载的 sosoDAO.isCardExit(number, password)检查正确密码是否符合传入的密码参数。

2.3. 获取新卡号

使用 random.nextInt(int)获取不重复的新号码，检查不重复之后加入输出数组中。

```
public String[] getNewNumbers(int count) {
    Random random = new Random();
    int num;
    String[] numbers = new String[count];
    for (int i = 0; i < count; i++){
        do {
            num = random.nextInt( bound: 100000000);
        } while (isExistCard(String.valueOf(13900000000L + num)));
        numbers[i] = String.valueOf(13900000000L + num);
    }
    return numbers;
}
```

2.4. 添加新卡

整合了全部的外部函数功能，分层验证卡号信息，最后调用 DAO 接口的 save 完成记录的更新。

选择卡号，检查输入合法后生成 mobieCard 实体类

```
System.out.println("*****可选的卡号*****");
String[] numbers = getNewNumbers(numberCount);
int cntLine = 0;    //该变量用于计数卡号数量，决定是否换行
for (int i = 0; i < numbers.length; i++, cntLine++) {
    if (cntLine != 0) {
        System.out.print("\t");
    }
    System.out.print(i + 1 + "." + numbers[i]);
    if (cntLine == 2) {
        cntLine = -1;
        System.out.println();
    }
}
do {
    System.out.print("请选择卡号(输入1~9的序号)：");
    try {
        numIndex = scanner.nextInt();
    }
    catch (InputMismatchException e) {
        scanner = new Scanner(System.in);
        numIndex = -1;
    }
} while (numIndex < 1 || numIndex > numberCount);    //循环检查输入是否合法
newCard.cardNumber = numbers[numIndex - 1];
```

录入套餐类型，使用 findPackage()列出数据库中所有的可用套餐并按顺序打印

```
ServicePackage[] servicePackages = sosoDAO.findPackage();

int ind;
do {
    System.out.println("请选择套餐(输入序号)：");
    ind = 1;
    for (ServicePackage i : servicePackages) {
        if (i != null) {
            System.out.println("\t" + ind++ + "\t" + i.name);
        }
    }
    numIndex = scanner.nextInt();
} while (numIndex < 0 || numIndex > ind);

newCard.serPackage = servicePackages[numIndex - 1];
```

录入姓名密码与金额，对比金额可用后录入

```
double preMoney = -1;
boolean tempFlag = true;
do {
    System.out.print("请输入预存话费金额：");
    if (!tempFlag) {
        System.out.print("您预存的话费金额不足以支付本月固定套餐资费，请重新充值：");
    }
    try {
        preMoney = scanner.nextDouble();
    }
    catch (InputMismatchException e) {
        scanner = new Scanner(System.in);
        System.out.println("请输入正确的数字");
    }
    tempFlag = !(preMoney < newCard.serPackage.price) || !(preMoney > 0);
} while (preMoney <= 0 || preMoney < newCard.serPackage.price);    //循环检查输入是否合法
```

2.5. 删除账号

```
public void delCard(String card) {
    sosoDAO.del(card);
    System.out.print("*****办理退网*****\n" + "卡号" + card + "办理退网成功！\n" + "谢谢使用！\n");
}
```

2.6. 显示套餐余量

使用 findPackage 的重载方法判断套餐类型之后调用对应数据

```
public void showRemainDetail(String number) {
    DecimalFormat formatData = new DecimalFormat( pattern: "#.0");

    System.out.print("******套餐余量查询******\n" + "您的卡号是" + number + "套餐内剩余：\n");
    MobileCard c = sosoDAO.findCardByNumber(number);
    ServicePackage tempServerPackage = sosoDAO.findPackage(c.serPackage.type);
    System.out.print("通话时长：" + Math.max(tempServerPackage.getTalkTime() - c.realTalkTime, 0) + "分钟\n"
            + "短信条数：" + Math.max(tempServerPackage.getSmsCount(), 0) + "条\n"
            + "上网流量：" + formatData.format(Math.max(tempServerPackage.getFlow() - c.realFlow, 0)) + "GB\n");
}
```

格式化输出使用 Java 的 DecimalFormat 类

2.7. 本月账单查询

```
public void showAmountDetail(String number) {
    DecimalFormat formatData = new DecimalFormat( pattern: "#.0");

    MobileCard c = sosoDAO.findCardByNumber(number);

    System.out.print(
        "******本月账单查询******\n" +
        "您的卡号：" + number + "，当月账单：\n" +
        "套餐资费：" + formatData.format(c.serPackage.price) + "元\n" +
        "合计：" + formatData.format(c.consumAmount) + "元\n" +
        "账户余额：" + formatData.format(c.money) + "元\n");
}
```

格式化输出使用 Java 的 DecimalFormat 类

2.8. 打印消费记录

为避免频繁调用 print 函数，将所有数据使用 append 添加到缓冲字符串，最后一次性输出。

```
public void printConsumInfo(String number) {
    StringBuilder content = new StringBuilder();
    content.append("**********").append(number).append("消费记录**********\n").append("序号\t类型\t数据[通话(分钟)/上网(MB)/短信(条)]\n");
    ConsumInfo[] list = sosoDAO.findRecByNumber(number);
    int ind = 1;
    for (ConsumInfo i : list) {
        if (i != null) {
            content.append(ind++).append(".\t").append(i.type).append("\t").append(i.consumData).append("\n");
        }
    }
    System.out.print(content);

    try {
        File file = new File( pathname: number + " 消费记录.txt");
        if (file.createNewFile()) {
            FileWriter writer = new FileWriter(file);
            writer.write(content.toString());
            writer.flush();
            writer.close();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

向屏幕输出完成后，调用 File 类将缓冲字符串写入文件

2.9. 显示资费说明

资费说明.txt 被预先制作好放置于项目根目录下，程序会尝试读取并输出

```
public void showDescription() {

    try {
        FileReader fileReader = new FileReader( fileName: "资费说明.txt");
        BufferedReader bufferedReader = new BufferedReader(fileReader);
        String line = bufferedReader.readLine();

        while (line != null) {  //循环输出每一行数据
            System.out.println(line);
            line = bufferedReader.readLine();
        }

        bufferedReader.close();
        fileReader.close();
    } catch (FileNotFoundException e) {
        System.out.println("资费说明文件未找到");
    } catch (IOException e) {
        System.out.println("文件读取错误");
    }
}
```

2.10. 更换套餐

先使用 find Package 检查套餐是否即将更换

```
System.out.println("******套餐变更******");
String numIndex;
int nIndex;
int ind;
do {
    ind = 1;
    System.out.println("请选择(序号)：");
    for (ServicePackage i : servicePackages)
        if (i != null) {
            System.out.println("\t" + ind++ + "\t" + i.name);
        }
    numIndex = scanner.next();
    nIndex = Integer.parseInt(numIndex);
} while (nIndex < 0 || nIndex > ind);

MobileCard c = sosoDAO.findCardByNumber(number);

//检查更换套餐是否发生了变更
if (servicePackages[nIndex - 1].type.equals(c.serPackage.type)) {
    System.out.println("对不起，您已经是该套餐用户，无需更换套餐！");
    return;
}
```

再检查余额是否满足新套餐月资费，最后更换套餐，扣除余额与各项使用时长并显示新套餐与账户信息。

```
    //检查余额
    ServicePackage toSp = servicePackages[nIndex - 1];

    if (c.money < toSp.price) {
        System.out.println("对不起，您的余额不足以支付新套餐本月资费，请充值后再办理更换套餐业务！");
        return;
    }

    //扣除套餐费用，清空使用时间，更换套餐
    c.money -= toSp.price;
    c.realTalkTime = c.realSMSCount = c.realFlow = 0;
    c.serPackage = toSp;
    sosoDAO.update(c);
    System.out.println("套餐更换成功");

    //输出相关信息
    c.showMeg();
    c.serPackage.showInfo();
}
```

2.11. 充值

最低充值限额检查被分离到菜单方法中进行，这里不做重复检查。

```
Scanner scanner = new Scanner(System.in);

System.out.print("请输入充值卡号：");
String number = scanner.next();

if (isExistCard(number)) {
    double money = -1;
    do {
        System.out.print("请输入充值金额：");
        try {
            money = scanner.nextDouble();
        }
        catch (InputMismatchException e) {
            scanner = new Scanner(System.in);
            System.out.println("请输入正确的数字");
        }
    } while (money <= 0);    //循环检查输入是否合法

    if (money < 50) {
        System.out.println("充值金额至少为50元");
    }
    else {
        DecimalFormat formatData = new DecimalFormat( pattern: "#.0");    //用于控制精度
        MobileCard mobileCard = sosoDAO.findCardByNumber(number);
        mobileCard.money += money;
        sosoDAO.update(mobileCard);
        System.out.println(String.format("充值成功，当前话费余额为%s元。", formatData.format(mobileCard.money)));
    }
}
else {
    System.out.println("充值卡号不存在，请重试");
}
```

充值完成后按要求的精度输出。

2.12. 使用嗖嗖

首先使用 random 类生成随机数，调用对应的场景

```
Random random = new Random();
int i;
MobileCard c = sosoDAO.findCardByNumber(number);
Scene scene;
ConsumInfo info;
boolean flag = false;

scene = scenes.get(random.nextInt( bound: 6));
```

```
try {
    if (scene.type.equals("通话")) {
        (c.serPackage).call(scene.data, c);
    }
    else if (scene.type.equals("短信")) {
        (c.serPackage).send(scene.data, c);
    }
    else {
        (c.serPackage).netPlay(scene.data, c);
    }
} catch (InsufficientBalanceException e) {
    System.out.println("本次已" + e.done() + "，您的余额不足，请充值后再使用！");
    e.printStackTrace();
} finally {
    System.out.print(scene.description + scene.type + scene.data);
    if (scene.type.equals("通话")) System.out.println("分钟");
    else if (scene.type.equals("上网")) System.out.println("MB");
    else System.out.println("条");
}
```

随后分场景类型调用对应套餐的功能性方法，并通过异常捕获，检查是否出现了余额不足。并进行相应的异常处理。具体扣款与检查逻辑分离至套餐内部实现。一切完成后添加消费记录。

```
if (sosoDAO.findRecByNumber(number) == null) {
    System.out.print("不存在此卡的消费记录，");
}
info = new ConsumInfo(number, scene.type, scene.data);
addConsumInfo(number, info);
System.out.println("已添加一条消费记录。");
```

3. 电话卡实体类

3.0. 构造函数

为方便工具类中的方法，这里保留无参构造函数，同时也编写了有参构造函数。

3.1. 打印电话卡信息

```
public void showMeg(){
    DecimalFormat formatData = new DecimalFormat( pattern: "#.0");
    System.out.println("卡号：" + cardNumber + "用户名：" + userName + "当前余额" + formatData.format(money));
}
```

使用 DecimalFormat 控制精度

4. 用户信息类

4.0. 构造函数

初始化用户通话记录中各项变量

5. 场景类

5.0. 构造函数

初始化场景各项成员变量，包括消费类型，数额与详细信息

6. 异常类 InsufficientBalanceException

6.0. 异常类包括一个成员变量完成度 done，用于记录在发生余额不足异常时，已经完成的消费数量，构造函数为其 setter，成员方法 print 为其 getter。

```
class InsufficientBalanceException extends Exception {
    private String done;

    public InsufficientBalanceException(String done) { this.done = done; }

    public String done() { return done; }
}
```

7. 套餐类 ServicePackage

7.0. 用于记录任意套餐信息，由单独的 type 变量记录类型，name 变量记录中文名称。showInfo()输出信息。

7.1. 功能函数 call(), send(), netplay()。分别判断用户套餐余量及类型进行消费的扣费。

```
public int send(int count, MobileCard card) throws InsufficientBalanceException {

    if (card.realSMSCount + count >= smsCount) {
        int temp = 0;
        if (card.realSMSCount < smsCount) {
            temp += smsCount - card.realSMSCount;
            count -= smsCount - card.realSMSCount;
            card.realSMSCount = smsCount;
        }
        if (card.money >= count * 0.1) {
            card.realSMSCount += count;
            card.money -= count * 0.1;
            temp += count;
            return temp;
        }
        else {
            temp += (int) Math.round(card.money / 0.1);
            card.realSMSCount += temp;
            card.money = 0;
            throw new InsufficientBalanceException("短信" + temp + "条");
        }
    }
    else {
        card.realSMSCount += count;
        return count;
    }
}
```

8. DAO 接口

8.0. 为实际使用方便， 为 DAO 准备了以下方法：

```
void save(MobileCard mobileCard);
void del(String number);
void update(MobileCard mobileCard);
MobileCard findCardByNumber(String Number);
```

```
boolean isCardExit(String number);
boolean isCardExit(String number, String password);
```

```
void save(ConsumInfo consumInfo);
void del(ConsumInfo consumInfo);
void update(ConsumInfo consumInfo);
ConsumInfo[] findRecByNumber(String Number);
```

```
ServicePackage findPackage(String name);
ServicePackage[] findPackage();
```

分别用于对三张表（电话卡（用户信息），消费记录，套餐）的增删改查

9. DAO 工具类

9.0. Init()初始化工具类，该方法在 DAO 类被构造时调用，用于读取配置信息，设置好数据库的几个静态参数变量。该参数变量一旦加载会在程序的生命周期中长期有效。

```
protected static void init() {
    Properties params = new Properties();
    InputStream is = SosoDaoUtil.class.getClassLoader().getResourceAsStream(CONFIG_FILE);
    try {
        params.load(is);
    }
    catch (IOException e) {
        e.printStackTrace();
    }
    catch (NullPointerException ne) {
        System.out.println("数据库配置文件无法访问！");
        return;
    }
    JDBC_DRIVER = params.getProperty("driver");
    DB_URL = params.getProperty("url");
    USER = params.getProperty("username");
    PASS = params.getProperty("password");
}
```

9.1. getConnection() 建立数据库连接并返回获取到的连接类

```java
public Connection getConnection() {
    Connection connection = null;
    try {
        Class.forName(JDBC_DRIVER);
    }
    catch (ClassNotFoundException e) {
        e.printStackTrace();
    }

    try {
        connection = DriverManager.getConnection(DB_URL, USER, PASS);
    }
    catch (SQLException e) {
        e.printStackTrace();
    }
    return connection;
}
```

9.2. closeConnection() 关闭 connection 连接及相关工具类 PreparedStatement 和 ResultSet。

```java
if (connection != null) {
    try {
        connection.close();
    }
    catch (SQLException e) {
        e.printStackTrace();
    }
}
```

```java
if (stmt != null) {
    try {
        stmt.close();
    }
    catch (SQLException e) {
        e.printStackTrace();
    }
}
```

```java
if (rs != null) {
    try {
        rs.close();
    }
    catch (SQLException e) {
        e.printStackTrace();
    }
}
```

10. Dao 接口的 MySQL 实现

10.0. 实现中全部使用 prepareStatement 提交 sql，不存在 sql 注入漏洞

10.1. 实现类继承自工具类，实现 DAO 接口，构造函数调用工具类的 init()方法初始化连接参数

```java
public SosoDaoMySQL() { init(); }
```

10.2. mobieCard 类使用的 save

```java
public void save(MobileCard mobileCard) {

    String sql = "INSERT INTO user_info " +
            "( card_number, user_name, password, service_package, money, real_talk_time, real_SMS_count, real_flow) " +
            "VALUES (?, ?, ?, ?, ?, 0, 0, 0)";

    Connection connection = getConnection();
    PreparedStatement preparedStatement = null;

    try {
        preparedStatement = connection.prepareStatement(sql);
        preparedStatement.setString( parameterIndex: 1, mobileCard.cardNumber);
        preparedStatement.setString( parameterIndex: 2, mobileCard.userName);
        preparedStatement.setString( parameterIndex: 3, mobileCard.password);
        preparedStatement.setString( parameterIndex: 4, mobileCard.serPackage.type);
        preparedStatement.setDouble( parameterIndex: 5, mobileCard.money);

        preparedStatement.executeUpdate();
    }
    catch (SQLException e) {
        e.printStackTrace();
    }

    closeConnection(connection, preparedStatement, rs: null);
}
```

10.3. mobieCard 类使用的 del

```java
public void del(String number) {
    String sql = "DELETE FROM user_info WHERE card_number = ?";

    Connection connection = getConnection();
    PreparedStatement preparedStatement = null;

    try {
        preparedStatement = connection.prepareStatement(sql);
        preparedStatement.setString( parameterIndex: 1, number);

        preparedStatement.executeUpdate();
    }
    catch (SQLException e) {
        e.printStackTrace();
    }

    closeConnection(connection, preparedStatement, rs: null);
}
```

10.4. mobieCard 使用的 updata

```java
String sql = "UPDATE user_info SET " +
        "card_number = ?, " +
        "user_name = ?, " +
        "password = ?, " +
        "service_package = ?, " +
        "money = ?, " +
        "real_talk_time = ?, " +
        "real_SMS_count = ?, " +
        "real_flow = ? " +
        "WHERE card_number = ?";
```

10.5. mobieCard 使用的查找方法，包括由手机号码查找完整类和由手机号或与密码一起验证存在

```java
public MobileCard findCardByNumber(String Number) {
    MobileCard mobileCard = null;

    String sql = "SELECT * FROM user_info WHERE card_number = ?";
    Connection connection = getConnection();
    PreparedStatement preparedStatement;
    ResultSet rs;

    try {
        preparedStatement = connection.prepareStatement(sql);
        preparedStatement.setString( parameterIndex: 1, Number);

        rs = preparedStatement.executeQuery();
    }
    catch (SQLException e) {
        e.printStackTrace();
        return null;
    }

    try {
        rs.last();
        if (rs.getRow() == 0) {
            return null;  //判断是否查询到结果
        }
        mobileCard = new MobileCard(
                rs.getString( columnLabel: "card_number"),
                rs.getString( columnLabel: "user_name"),
                rs.getString( columnLabel: "password"),
                findPackage(rs.getString( columnLabel: "service_package")),
                rs.getDouble( columnLabel: "money")
        );
    }
    catch (SQLException e) {
        e.printStackTrace();
    }
    closeConnection(connection, preparedStatement, rs);
    return mobileCard;
}
```

```java
public boolean isCardExit(String number) {

    String sql = "SELECT user_name FROM user_info WHERE card_number = ?";
    Connection connection = getConnection();
    PreparedStatement preparedStatement = null;
    ResultSet rs = null;

    try {
        preparedStatement = connection.prepareStatement(sql);
        preparedStatement.setString( parameterIndex: 1, number);

        rs = preparedStatement.executeQuery();

        rs.last();
        if (rs.getRow() == 0) {
            return false;  //判断是否查询到结果
        }
        if (rs.getString( columnLabel: "user_name") != null) {
            return true;
        }
    }
    catch (SQLException e) {
        e.printStackTrace();
    }
    closeConnection(connection, preparedStatement, rs);
    return false;
}
```

```java
public boolean isCardExit(String number, String password) {

    String sql = "SELECT user_name, password FROM user_info WHERE card_number = ?";
    Connection connection = getConnection();
    PreparedStatement preparedStatement = null;
    ResultSet rs = null;

    try {
        preparedStatement = connection.prepareStatement(sql);
        preparedStatement.setString( parameterIndex: 1, number);

        rs = preparedStatement.executeQuery();
        rs.last();
        if (rs.getRow() == 0) {
            return false;  //判断是否查询到结果
        }
        if (rs.getString( columnIndex: 1) != null) {
            if (rs.getString( columnIndex: 2).equals(password)) {
                return true;
            }
        }
    }
    catch (SQLException e) {
        e.printStackTrace();
    }
    closeConnection(connection, preparedStatement, rs);
    return false;
}
```

10.6. 通话记录类使用的 save

```java
public void save(ConsumInfo consumInfo) {
    String sql = "INSERT INTO call_record " +
            "( card_number, type, consum_data, time) " +
            "VALUES (?, ?, ?, NOW())";

    Connection connection = getConnection();
    PreparedStatement preparedStatement = null;

    try {
        preparedStatement = connection.prepareStatement(sql);
        preparedStatement.setString( parameterIndex: 1, consumInfo.cardNumber);
        preparedStatement.setString( parameterIndex: 2, consumInfo.type);
        preparedStatement.setInt( parameterIndex: 3, consumInfo.consumData);

        preparedStatement.executeUpdate();
    }
    catch (SQLException e) {
        e.printStackTrace();
    }

    closeConnection(connection, preparedStatement,  rs: null);
}
```

10.7. 通话记录使用的 del

```java
public void del(ConsumInfo consumInfo) {
    String sql = "DELETE FROM call_record WHERE card_number = ?";

    Connection connection = getConnection();
    PreparedStatement preparedStatement = null;

    try {
        preparedStatement = connection.prepareStatement(sql);
        preparedStatement.setString( parameterIndex: 1, consumInfo.cardNumber);

        preparedStatement.executeUpdate();
    }
    catch (SQLException e) {
        e.printStackTrace();
    }

    closeConnection(connection, preparedStatement,  rs: null);
}
```

10.8. 通话记录的 update

```java
public void update(ConsumInfo consumInfo) {

    String sql = "UPDATE call_record SET " +
            "card_number = ?, " +
            "type = ?, " +
            "consum_data = ?, " +
            "time = NOW()" +
            "WHERE card_number = ?";

    Connection connection = getConnection();
    PreparedStatement preparedStatement = null;

    try {
        preparedStatement = connection.prepareStatement(sql);
        preparedStatement.setString( parameterIndex: 1, consumInfo.cardNumber);
        preparedStatement.setString( parameterIndex: 2, consumInfo.type);
        preparedStatement.setInt( parameterIndex: 3, consumInfo.consumData);
        preparedStatement.setString( parameterIndex: 4, consumInfo.cardNumber);

        preparedStatement.executeUpdate();
    }
    catch (SQLException e) {
        e.printStackTrace();
    }

    closeConnection(connection, preparedStatement,  rs: null);
}
```

10.9. 通话记录查找

```java
public ConsumInfo[] findRecByNumber(String Number) {
    ConsumInfo[] consumInfos;

    String sql = "SELECT type, consum_data FROM call_record WHERE card_number = ? ";
    Connection connection = getConnection();
    PreparedStatement preparedStatement;
    ResultSet rs;

    try {
        preparedStatement = connection.prepareStatement(sql);
        preparedStatement.setString( parameterIndex: 1, Number);
        rs = preparedStatement.executeQuery();
        consumInfos = new ConsumInfo[rs.getMetaData().getColumnCount()];
    }
    catch (SQLException e) {
        e.printStackTrace();
        return null;
    }

    try {
        int i = 0;
        while (rs.next()) {
            ConsumInfo tempConsumInfo = new ConsumInfo(
                    Number,
                    rs.getString( columnIndex: 1),
                    rs.getInt( columnIndex: 2)
            );
            consumInfos[i++] = tempConsumInfo;
        }
    }
    catch (SQLException e) {
        e.printStackTrace();
    }
    closeConnection(connection, preparedStatement, rs);
    return consumInfos;
}
```

10.10. 电话套餐遍历与特定套餐的查找

```java
public ServicePackage findPackage(String name) {
    ServicePackage servicePackage = null;

    String sql = "SELECT price, talktime, smscount, flow FROM servicePackage WHERE name = ?";
    Connection connection = getConnection();
    PreparedStatement preparedStatement;
    ResultSet rs;

    try {
        preparedStatement = connection.prepareStatement(sql);
        preparedStatement.setString( parameterIndex: 1, name);

        rs = preparedStatement.executeQuery();
    }
    catch (SQLException e) {
        e.printStackTrace();
        return null;
    }

    try {
        rs.last();
        if (rs.getRow() == 0) {
            return null;
        }
        servicePackage = new ServicePackage(
                name,
                rs.getDouble( columnIndex: 1),
                rs.getInt( columnIndex: 2),
                rs.getInt( columnIndex: 3),
                rs.getInt( columnIndex: 4)
        );
    }
    catch (SQLException e) {
        e.printStackTrace();
    }
    closeConnection(connection, preparedStatement, rs);
    return servicePackage;
}
```

```java
public ServicePackage[] findPackage() {

    ServicePackage[] servicePackage;

    String sql = "SELECT name, price, talkTime, smsCount, flow, zhName FROM servicePackage";
    Connection connection = getConnection();
    PreparedStatement preparedStatement;
    ResultSet rs;

    try {
        preparedStatement = connection.prepareStatement(sql);
        rs = preparedStatement.executeQuery();
        servicePackage = new ServicePackage[rs.getMetaData().getColumnCount()];
    }
    catch (SQLException e) {
        e.printStackTrace();
        return null;
    }

    try {
        int i = 0;
        while (rs.next()) {
            ServicePackage tempServicePackage = new ServicePackage(
                    rs.getString( columnIndex: 1),
                    rs.getDouble( columnIndex: 2),
                    rs.getInt( columnIndex: 3),
                    rs.getInt( columnIndex: 4),
                    rs.getInt( columnIndex: 5)
            );
            tempServicePackage.name = rs.getString( columnLabel: "zhName");
            servicePackage[i++] = tempServicePackage;
        }
    }
    catch (SQLException e) {
        e.printStackTrace();
    }

    closeConnection(connection, preparedStatement, rs);
    return servicePackage;
}
```