订单管理系统

任务要求：

假设工作室近期需要置办一系列搭建，邀请各位编写一个简单的订单管理系统(用来记账)

- 至少需要记录以下信息:
  - 商品: 商品编号、商品名、商品价格
  - 订单: 订单编号、**商品信息**(考虑如何合理存储关联信息)、下单时间、订单价格
- 需要将信息保存至**数据库**
- 编写JDBC工具类，功能包括但不限于：
  - 处理数据库连接
  - 执行增删查改操作
  - 解决**SQL注入问题**
  - 添加**事务管理**
  - 包含异常处理和资源释放
- **使用编写的JDBC工具类实现商品和订单信息的增删改查、更新，商品和订单排序（价格、下单时间）等功能**
  - 在创建订单时，实施数据验证，确保订单信息的完整性和准确性。例如，检查商品是否存在，价格是否合法等等。
  - 如果想要删除已经存在在订单中的商品，你要怎么处理？
  - 避免使用SELECT *
- **完整的测试**
- 可以自己设计新功能，有需要也可以自行增加字段。
- 在代码中加入必要的注释，说明每个方法的作用和功能，以及关键代码的解释。
- 使用合适的代码风格和命名规范，确保代码的可读性和可维护性

1.数据库设计

1. 商品表（goods）
   - 商品编号（goods_id）：INT，主键，自增长，唯一标识商品。
   - 商品名（goods_name）：VARCHAR (255)，存储商品名称。
   - 商品价格（goods_price）：DECIMAL (10, 2)，精确到小数点后两位，存储商品价格。
2. 订单表（orders）
   - 订单编号（order_id）：INT，主键，自增长，唯一标识订单。
   - 下单时间（order_time）：TIMESTAMP，记录订单创建时间。
   - 订单价格（order_price）：DECIMAL (10, 2)，存储订单总价格。
3. 订单细节表（order_details）
   - 订单编号 （order_id）INT 标识订单
   - 商品编号 （goods_id）INT 标识商品

设计思路：从任务要求可以看出，在商品中，商品名和商品价格是取决于商品编号并且是一一对应的，在订单中，下单时间和订单价格和订单编号是一一对应的，商品信息和订单不是一一对应的，会存在一个订单有着多个商品的情况。如果把商品信息放在订单表中实现一一对应，那么在同一行中会存在多个商品编号字段，但是一个订单的商品个数其实是未知的，字段多开了浪费空间，字段少开使得功能受限，所以我将商品信息拉出来新建一个表来实现一个订单对应多个商品编号，减少了下单时间和订单价格的冗余。并且也可以实现对订单中的单个商品做删除处理。

2.建立表类

根据表可以分为 商品表（goods），订单表（orders），订单细节表（order_details）

- 商品表（goods）

  有id,name,price成员变量，并实现基本的get,set方法重写了toString 方法.

- 订单表（orders）

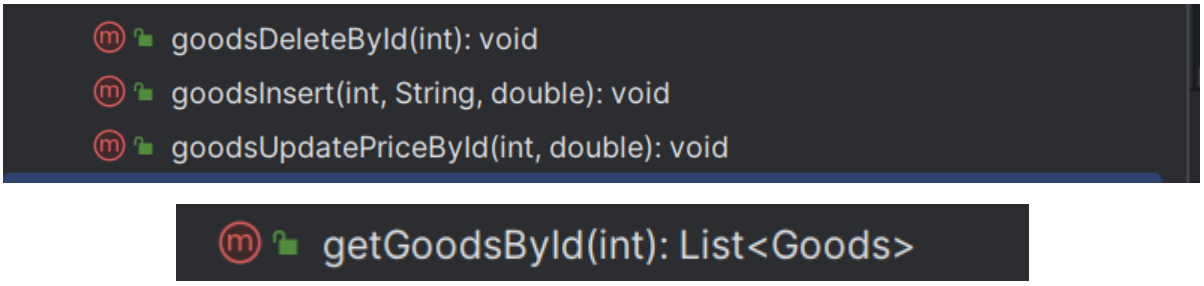
  有id,time,price成员变量，并实现基本的get,set方法重写了toString 方法.

- 订单细节表（order_details）

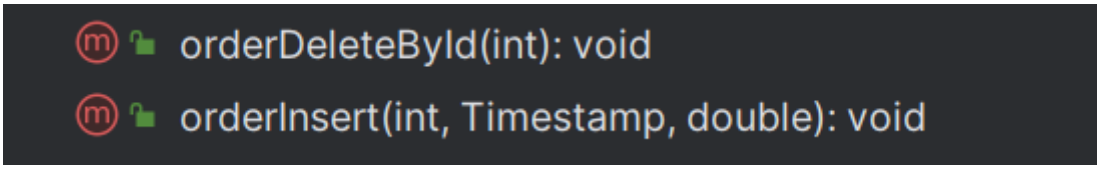
  有goods_id,order_id成员变量，并实现基本的get,set方法重写了toString 方法.

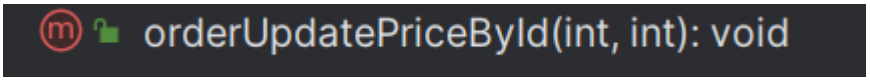
3.JDBC工具类的实现

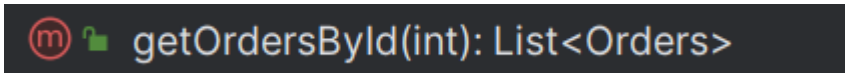
商品表（goods）的增删改查



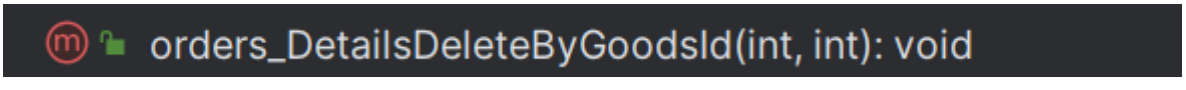
订单表（orders）的增删改查

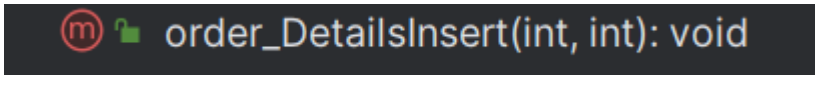


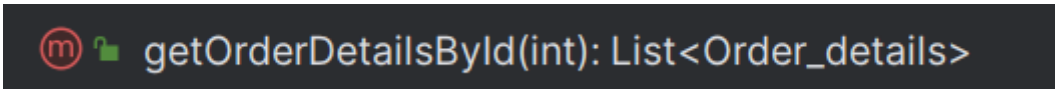




订单细节表（order_details）的增删查







4.处理数据库连接

1. 加载数据库驱动：

```
String driverName="com.mysql.cj.jdbc.Driver";
Class.forName(driverName);
```

- `Class.forName(driverName)`：这行代码的作用是加载 MySQL 的 JDBC 驱动程序。在 Java 中，使用 `Class.forName()` 方法动态加载类，这里加载的是 `com.mysql.cj.jdbc.Driver` 类，这是 MySQL 8.0 及以上版本使用的 JDBC 驱动类。当类被加载时，会自动执行静态代码块，该静态代码块会注册驱动程序，使得 JDBC 能够使用这个驱动程序来连接 MySQL 数据库。

2. 构建数据库连接 URL：

```
String dbURL="jdbc:mysql:///order management system";
```

- `dbURL`：这是数据库的连接字符串，`jdbc:mysql:///order management system` 是一个相对简单的 URL 格式。这里使用的是默认主机（本地主机）和默认端口（3306），并且数据库名称是 `order management system`。完整的形式通常是 `jdbc:mysql://host:port/databaseName`，例如 `jdbc:mysql://localhost:3306/order_management_system`。这里使用 `jdbc:mysql:///order management system` 会让 JDBC 驱动程序使用默认的 `localhost` 和 `3306` 端口。

3. 提供数据库用户名和密码：

```
String userName="root";
String userPwd="123456";
```

- `userName`：数据库的用户名，这里使用 `root`。
- `userPwd`：数据库的密码，这里是 `123456`。

实现：

```
4   public class LinkDatabase {
        15 个用法
5       public static Connection getConnection() {
6           // TODO Auto-generated method stub
7           String driverName="com.mysql.cj.jdbc.Driver";
8
9           String dbURL="jdbc:mysql:///order management system";
10          //账户
11          String userName="root";
12          //密码
13          String userPwd="123456";
14          Connection conn=null;
15          try {
16              Class.forName(driverName);
17              conn= DriverManager.getConnection(dbURL,userName,userPwd);
18              System.out.println("数据库连接成功");
19          }catch(Exception e) {
20              System.out.println("数据库连接失败");
21          }
22          return conn;
23      }
24  }
```

5.sql注入问题

使用 `PreparedStatement` 防止sql注入。

- 当使用 `PreparedStatement` 时，参数部分被视为数据而不是代码。
- 数据库会对传递的参数进行转义或处理，以确保它们不会影响 SQL 语句的结构和逻辑。例如，如果用户输入包含特殊字符（如 `'`、`--`、`;` 等），数据库会将它们作为普通数据处理，而不是 SQL 命令的一部分。
- 因为 SQL 语句已经预编译，攻击者无法修改 SQL 语句的结构，只能提供参数，这些参数会被安全地处理，不会影响 SQL 语句的执行逻辑。

6.添加事务管理

```
connection.setAutoCommit(false);//调用connection方法关闭事务的自动提交。
```

```
connection.commit();//提交
connection.rollback();//回滚
```

7.异常处理和资源释放

使用try..catch实现异常处理。

通过

```
 connection.close();
 preparedStatement.close();
```

释放资源

8.在创建订单时，实施数据验证，确保订单信息的完整性和准确性。例如，检查商品是否存在，价格是否合法等等。

在订单表里的更新和插入，创建了一个isPriceValid方法，去查询订单详情表里的订单应该需要的价格，进行比对，如果不合法，则不执行，直接返回。

在订单详情表里的插入，创建了existGoods方法，通过查询商品表里的id字段是否存在，如果不存在，则不执行，直接返回。

9.如果想要删除已经存在在订单中的商品，你要怎么处理?

我采用了订单详情表来分条记录订单所对应的商品信息，只需要对那个表执行删除语句就可以了。

针对原订单表里对应订单价格是否需要更改，我觉得这个操作可以在主函数内执行，再对订单进行更新价格。或者也可以在那个方法内再执行订单更新方法，放入对应的参数去执行。不过我这里选择前者。

10.待改进

1.减少连接数据库的次数：

可以尝试主函数中开启连接，然后再传参进方法中。

2.尝试使用异常器去捕获异常