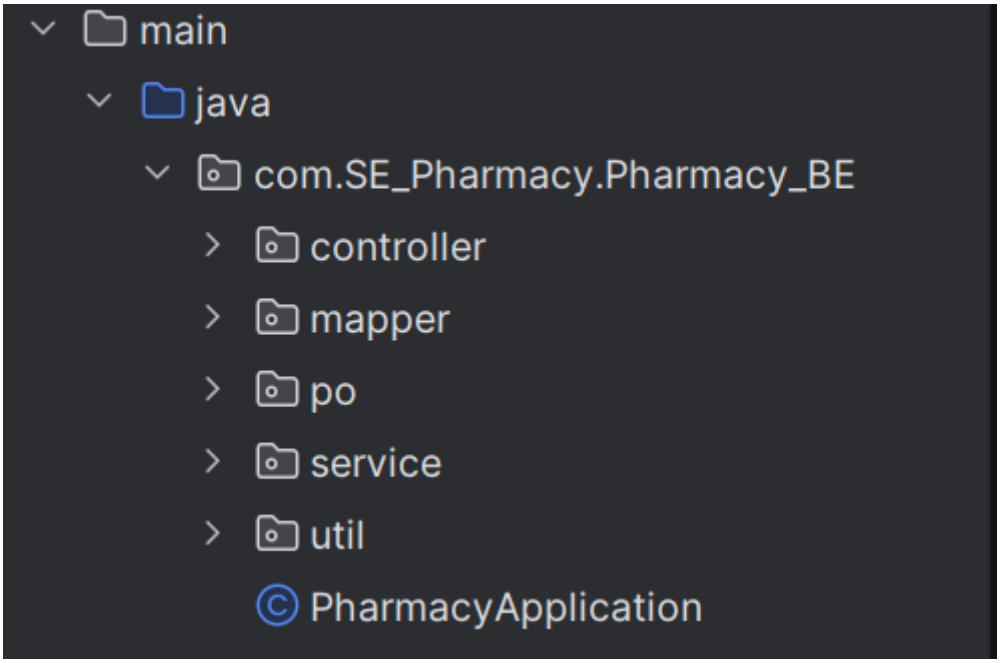
后端

表	实现功能
Administrator	• 添加、删除、更新
Bill	• 增加、删除、根据库存id获取bill、根据id获取bill、更新bill
Drug	• 增加、删除、更新、获取所有药品信息、根据id获取某个药品的信息
Medicine	• 增加删除更新、根据仓库id获取那个仓库的药品储存情况、根据id获取某个药品的储存情况 • 预想还写一个：得到超出有效日期的所有药品（没写）
Queue	• 增加、删除、更新、获取某个仓库的排队情况、获取某个billId的排队
Shopping	• 增加删除更新、获取某个用户的所有购物车、获取某个仓库的所有购物车、后去某个billId的所有购物车
Window	• 增加删除更新、获取某个仓库的所有窗口、获取某个billId的所有窗口 • 肯定是有问题的 后续再更改

> 项目层次

1. Controller：控制器层
    - 控制器层负责处理 HTTP 请求和响应，它接收来自客户端的请求，并根据请求执行相应的操作。它将请求传递给服务层来处理业务逻辑，并将结果返回给客户端。
    - 主要功能包括定义和管理 REST API 端点，接受和验证请求参数，调用适当的服务方法来执行操作，处理异常和错误，返回 HTTP 响应等。
2. Mapper：映射器层
    - 映射器层通常用于将数据从数据库或其他数据源映射到领域对象（POJO）或将领域对象映射到数据存储格式。它在数据存储和应用程序内部领域对象之间执行数据的转换和映射。
    - 主要功能包括定义数据映射规则、执行数据的读取和写入、将数据库记录转换为领域对象以及将领域对象转换为适当的数据格式。
3. PO (Persistence Object)：持久化对象
    - 持久化对象是应用程序中表示数据的对象，通常与数据库表或其他数据存储结构相对应。它们是领域对象的映射，包含数据字段以及与数据存储相关的元数据。
    - 主要功能包括定义数据模型，表示实体之间的关系，以及提供数据的访问和操作方法。

4. Service：服务层
    - 服务层包含应用程序的核心业务逻辑，它提供高层次的抽象，将数据操作、数据处理和业务规则分离。服务层通常调用数据访问层（Mapper）执行数据库操作。
    - 主要功能包括实现应用程序的业务逻辑，协调多个数据操作，提供服务方法用于控制器层调用，执行事务管理，处理业务规则等。
5. Util：工具类
    - 工具类层包含用于执行通用操作的工具方法。这些方法通常与特定领域对象或业务逻辑无关，用于提供重复使用的功能或处理通用任务。
    - 主要功能包括提供各种辅助函数、通用功能（如日期格式化、加密、验证等）以及任何可重复使用的工具。

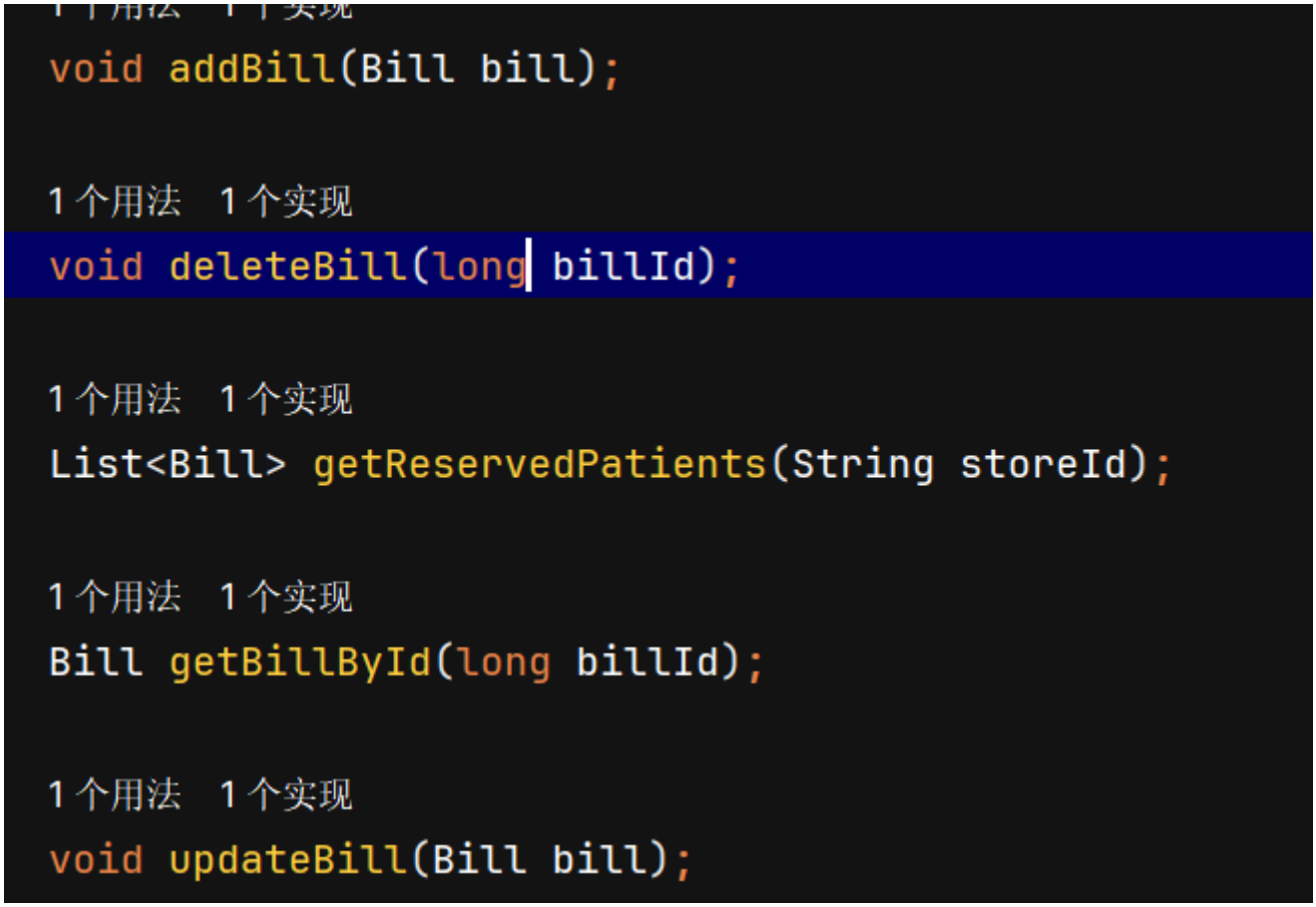


> 实现了哪些功能（针对与每个表）

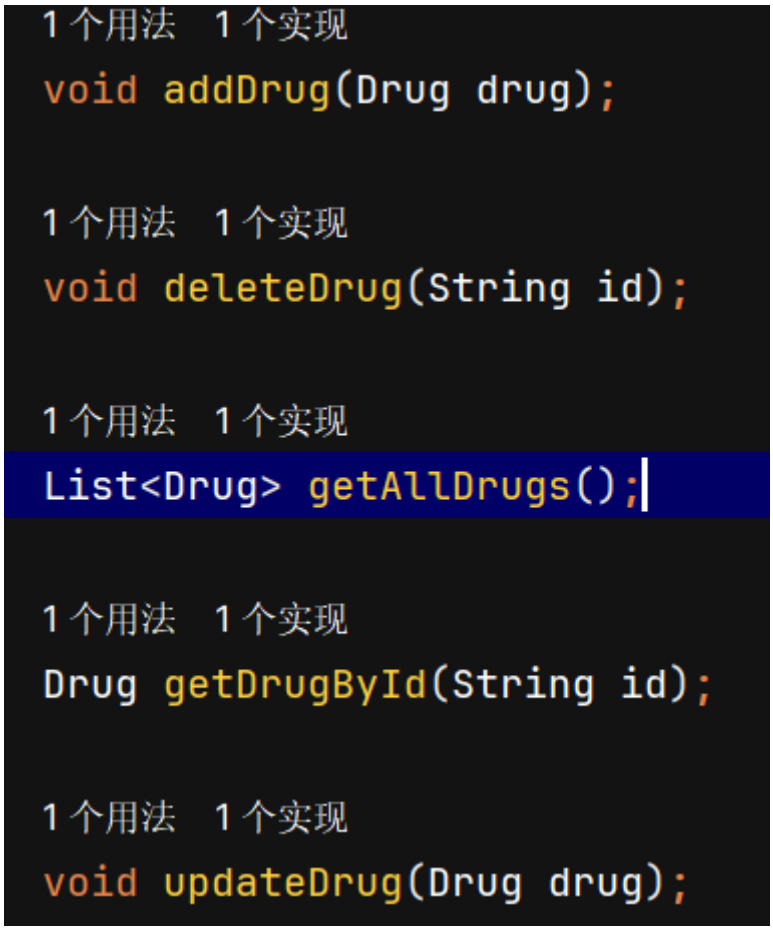
1. Administrator
    - 添加、删除、更新

```
1 个用法
void insertAdministrator(Administrator administrator);

1 个用法
void updateAdministrator(Administrator administrator);

1 个用法
void deleteAdministrator(@Param("ano") String ano);
```

2. Bill
   - 增加、删除、根据库存id获取bill、根据id获取bill、更新bill



3. Drug
   - 增加、删除、更新、获取所有药品信息、根据id获取某个药品的信息



4. Medicine
   - 增加删除更新、根据仓库id获取那个仓库的药品储存情况、根据id获取某个药品的储存情况
   - 预想还写一个：得到超出有效日期的所有药品（没写）

```
1 个用法  1 个实现
void addMedicine(Medicine medicine);

1 个用法  1 个实现
void deleteMedicine(String id);

1 个用法  1 个实现
List<Medicine> getMedicinesByStorehouse(String storehouseId);

1 个用法  1 个实现
Medicine getMedicineById(String id);

1 个用法  1 个实现
void updateMedicine(Medicine medicine);
```

5. Queue

- 增加、删除、更新、获取某个仓库的排队情况、获取某个billId的排队

```
1个用法  1个实现
void addQueue(Queue queue);

1个用法  1个实现
void deleteQueue(Long qid);

1个用法  1个实现
List<Queue> getQueuesByStorehouse(String storehouseId);

1个用法  1个实现
Queue getQueueByBillId(Long billId);

1个用法  1个实现
void updateQueue(Queue queue);
```

6. Shopping
   - 增加删除更新、获取某个用户的所有购物车、获取某个仓库的所有购物车、后去某个billId的所有购物车

```
void addShoppingCart(ShoppingCart shoppingCart);

1个用法  1个实现
void deleteShoppingCart(Long id);

1个用法  1个实现
List<ShoppingCart> getShoppingCartByUserId(String userId);

1个用法  1个实现
List<ShoppingCart> getShoppingCartByStorehouse(String storehouseId);

1个用法  1个实现
List<ShoppingCart> getShoppingCartByBillId(Long billId);

1个用法  1个实现
void updateShoppingCart(ShoppingCart shoppingCart);
```

7. Window

- 增加删除更新、获取某个仓库的所有窗口、获取某个billId的所有窗口
- 肯定是有问题的 后续再更改

```
1 个用法  1 个实现
void addWindow(Window window);

1 个用法  1 个实现
void deleteWindow(String wid);

1 个用法  1 个实现
List<Window> getWindowsByStorehouse(String storehouseId);

1 个用法  1 个实现
Window getWindowByBillId(Long billId);

1 个用法  1 个实现
void updateWindow(Window window);
```