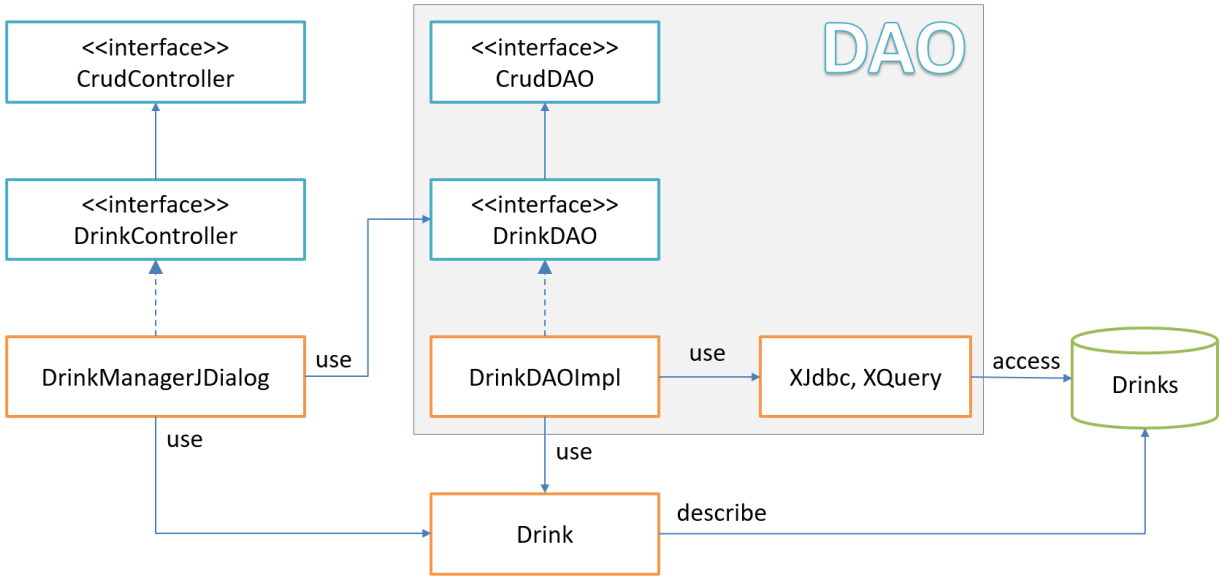
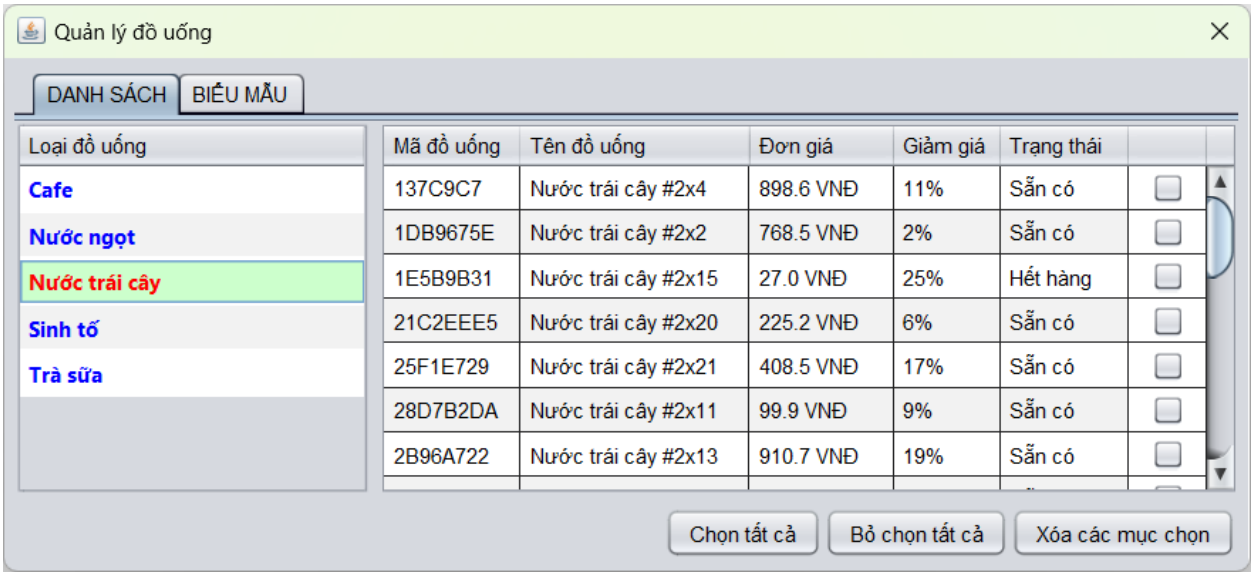
# BÀI 1: QUẢN LÝ ĐỒ UỐNG

Xây dựng chức năng Quản lý đồ uống được tổ chức theo mô hình sau đây

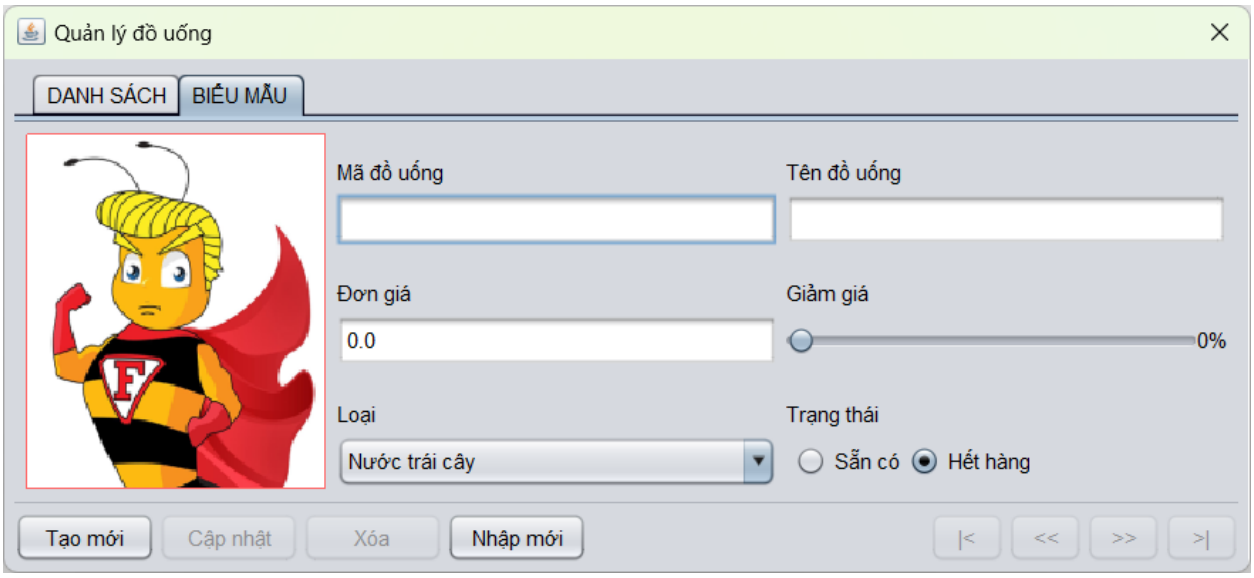
## SƠ ĐỒ TỔ CHỨC




## GIAO DIỆN




Hình 1: Danh sách đồ uống theo loại




Hình 2: Biểu mẫu

## DRINKCONTROLLER

- Khai báo fillCategories() để tải loại đồ uống lên cboCategories và tblCategories
- chooseFile() xử lý tương tác chọn file

```
public interface DrinkController extends CrudController<Drink>{
   void fillCategories();
   void chooseFile();
}
```

## CÀI ĐẶT MÃ NGUỒN CHO CONTROLLER

Chú ý: Một số điểm khác biệt so với CrudController cơ bản như sau:

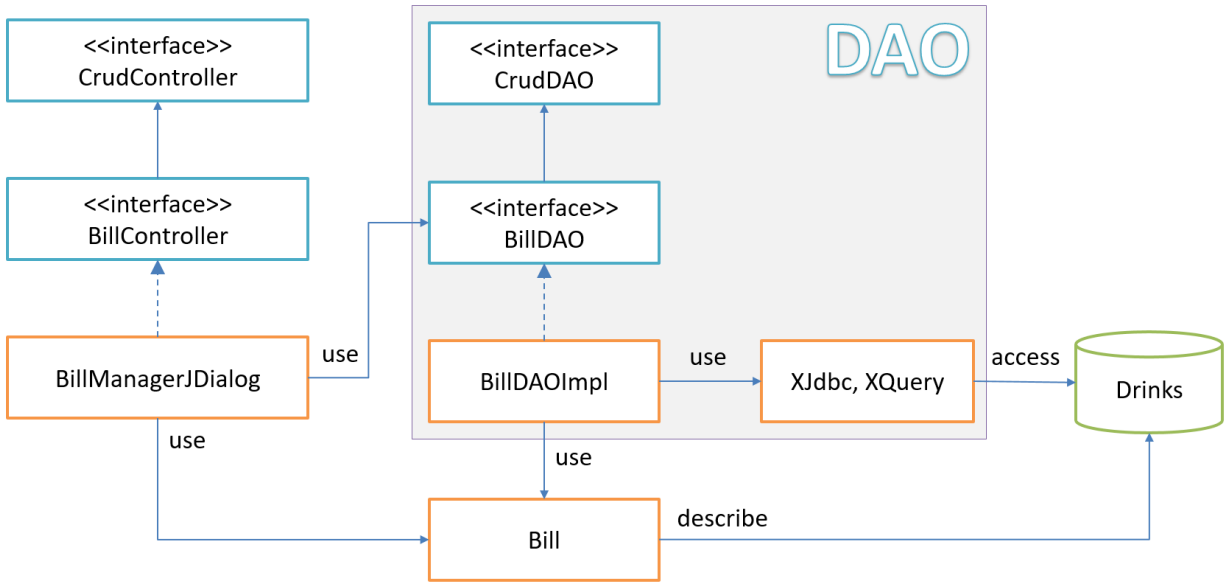
- Tải và hiển thị loại lên cboCategories và tblCategories
- Trong fillToTable sử dụng findByCategoryId() để truy vấn đồ uống theo loại thay vì findAll() truy vấn tất cả.
- Chọn và xử lý hình (lưu, hiển thị)

```
....
@Override
public void fillCategories() {
   DefaultComboBoxModel cboModel = (DefaultComboBoxModel) cboCategories.getModel();
   cboModel.removeAllElements();

   DefaultTableModel tblModel = (DefaultTableModel) tblCategories.getModel();
   tblModel.setRowCount(0);

   CategoryDAO cdao = new CategoryDAOImpl();
   categories = cdao.findAll();
```

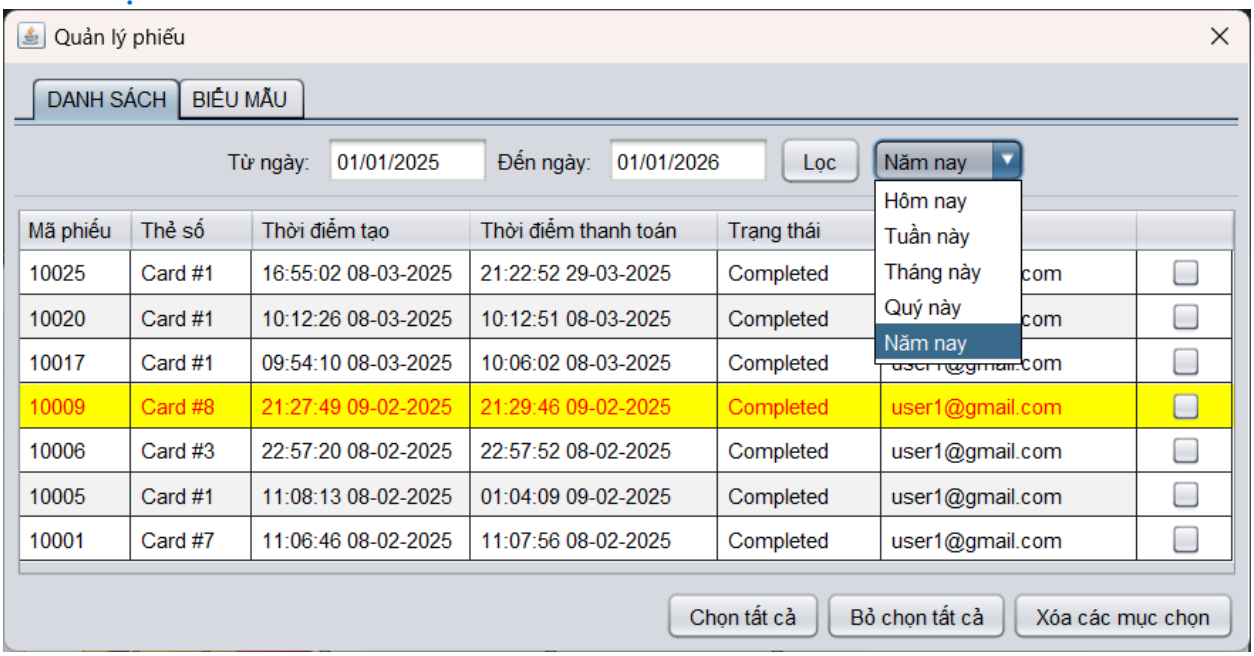
```
    categories.forEach(category -> {
        cboModel.addElement(category);
        tblModel.addRow(new Object[]{category.getName()});
    });

    tblCategories.setRowSelectionInterval(0, 0);
}
@Override
public void chooseFile() {
    if(fileChooser.showOpenDialog(this) == JFileChooser.APPROVE_OPTION){
        File selectedFile = fileChooser.getSelectedFile();
        File file = XIcon.copyTo(selectedFile, "images");
        lblImage.setToolTipText(file.getName());
        XIcon.setIcon(lblImage, file);
    }
}
@Override
public void fillToTable() {
    DefaultTableModel model = (DefaultTableModel) tblDrinks.getModel();
    model.setRowCount(0);

    Category category = categories.get(tblCategories.getSelectedRow());
    items = dao.findByCategoryId(category.getId()); // thay vì findAll()
    items.forEach(item -> {….});
    this.clear();
}
```

# BÀI 2: QUẢN LÝ PHIẾN BÁN HÀNG

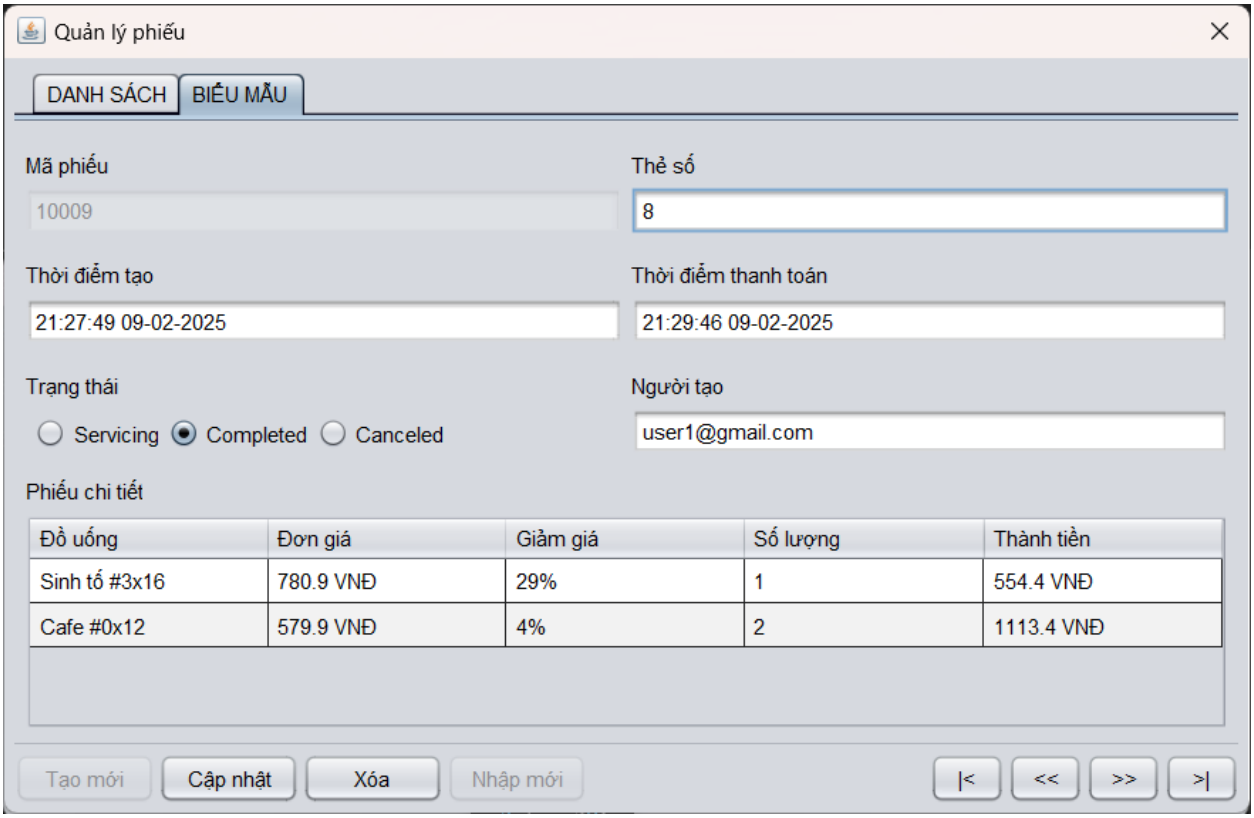
## SƠ ĐỒ TỔ CHỨC




## GIAO DIỆN




Hình 1: Danh sách




Hình 2: Biểu mẫu

## MỞ RỘNG ENTITY VÀ HIỆU CHỈNH DAO

Do cấu trúc chi tiết phiếu cần hiển thị tên đồ uống nên cần chỉnh BillDetail và BillDetailDAOInpl

- BillDetail: bổ sung thuộc tính drinkName
- BillDetailDAOImpl: tất cả các câu lệnh SELECT cần JOIN với Drinks để lấy tên đồ uống
- Bổ sung findByTimeRange() vào BillDAO để truy vấn phiếu theo khoảng thời gian lọc thay vì findAll() để truy vấn tất cả.

```
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Data
public class BillDetail {
   private Long id;
   private Long billId;
   private String drinkId;
   private double unitPrice;
   private double discount;
   private int quantity;

   private String drinkName;
```

```
}
```

```
public class BillDetailDAOImpl implements BillDetailDAO {
   String createSql = "…";
   String updateSql = "…";
   String deleteSql = "…";
String findAllSql = "SELECT bd.*, d.name AS drinkName
                    FROM BillDetails bd JOIN Drinks d ON d.Id=bd.DrinkId";
String findByIdSql = "SELECT bd.*, d.name AS drinkName
                    FROM BillDetails bd JOIN Drinks d ON d.Id=bd.DrinkId WHERE bd.Id=?";
String findByBillIdSql = "SELECT bd.*, d.name AS drinkName
                    FROM BillDetails bd JOIN Drinks d ON d.Id=bd.DrinkId WHERE bd.BillId=?";
String findByDrinkIdSql = "SELECT bd.*, d.name AS drinkName
                    FROM BillDetails bd JOIN Drinks d ON d.Id=bd.DrinkId WHERE bd.DrinkId=?";

   @Override
   public BillDetail create(BillDetail entity) {…}
   @Override
   public void update(BillDetail entity) {…}
   @Override
   public void deleteById(Long id) {…}
   @Override
   public List<BillDetail> findAll() {…}
   @Override
   public BillDetail findById(Long id) {…}

   @Override
   public List<BillDetail> findByBillId(Long billId) {…}
   @Override
   public List<BillDetail> findByDrinkId(String drinkId) {…}
}
```

BillDAO

```
public interface BillDAO extends CrudDAO<Bill, Long>{
   ….
   List<Bill> findByTimeRange(Date begin, Date end);
}
```

```
public class BillDAOImpl implements BillDAO {
   ….
   String findByTimeRangeSql = “SELECT * FROM Bills
                    WHERE Checkin BETWEEN ? AND ? ORDER BY Checkin DESC”;
   @Override
   public List<Bill> findByTimeRange (Date begin, Date end) {
      return XQuery.getBeanList(Bill.class, findByTimeRangeSql, begin, end)
   }
```

```
}
```

### BillController

```
public interface BillController extends CrudController<Bill>{
   void fillBillDetails(); // tải và hiển thị chi tiết phiếu
   void selectTimeRange(); // xử lý chọn khoảng thời gian trong cboTimeRanges
}
```

### Cài đặt mã nguồn cho controller

```
....
// Khai báo bổ sung
BillDAO dao = new BillDAOImpl();
List<Bill> items= List.of(); // phiếu bán hàng
BillDetailDAO billDetailDao = new BillDetailDAOImpl();
List<BillDetail> details = List.of(); // chi tiết phiếu bán hàng

@Override
public void open() {
   this.setLocationRelativeTo(null);
   this.selectTimeRange();
   this.clear();
}
@Override
public void setForm(Bill entity) {
   ...
   this.fillBillDetails();
}

@Override
public void fillBillDetails() {
   DefaultTableModel model = (DefaultTableModel) tblBillDetails.getModel();
   model.setRowCount(0);
   details = List.of();
   if (!txtId.getText().isBlank()) {
      Long billId = Long.valueOf(txtId.getText());
      details = billDetailDao.findByBillId(billId);
   }
   details.forEach(d -> {
      var amount = d.getUnitPrice() * d.getQuantity() * (1 - d.getDiscount());
      Object[] rowData = {
         d.getDrinkName(),
         String.format("%.1f VNĐ", d.getUnitPrice()),
         String.format("%.0f%%", d.getDiscount() * 100),
         d.getQuantity(), String.format("%.1f VNĐ", amount)
      };
```

```
        model.addRow(rowData);
    });
}

@Override
public void selectTimeRange() {
    TimeRange range = TimeRange.today();
    switch (cboTimeRanges.getSelectedIndex()) {
        case 0 -> range = TimeRange.today();
        case 1 -> range = TimeRange.thisWeek();
        case 2 -> range = TimeRange.thisMonth();
        case 3 -> range = TimeRange.thisQuarter();
        case 4 -> range = TimeRange.thisYear();
    }
    txtBegin.setText(XDate.format(range.getBegin(), "MM/dd/yyyy"));
    txtEnd.setText(XDate.format(range.getEnd(), "MM/dd/yyyy"));
    this.fillToTable();
}

@Override
public void fillToTable() {
    DefaultTableModel model = (DefaultTableModel) tblBills.getModel();
    model.setRowCount(0);
    Date begin = XDate.parse(txtBegin.getText(), “MM/dd/yyyyy”);
    Date end = XDate.parse(txtEnd.getText(), “MM/dd/yyyyy”);
    items = dao.findByTimeRange(begin, end);
    items.forEach(item -> {…});
}
```