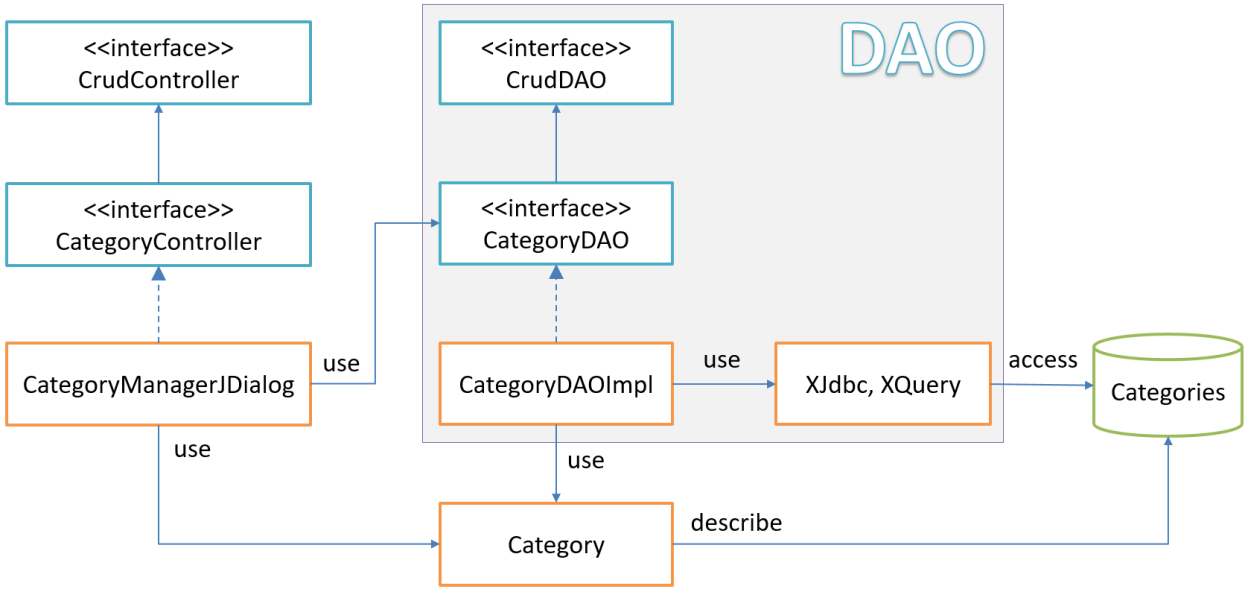
Xây dựng chức năng quản lý loại đồ uống được tổ chức như sơ đồ sau



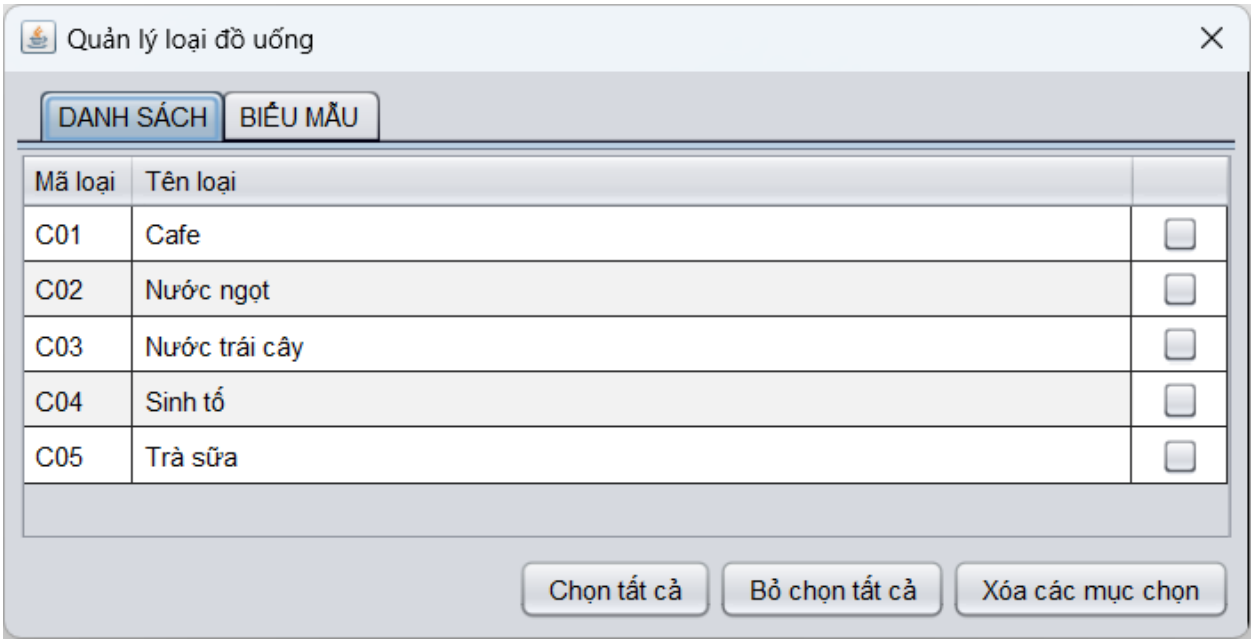
Các thành phần cần phải tạo mới

- CategoryManagerJDialog
- CrudController
- CategoryController

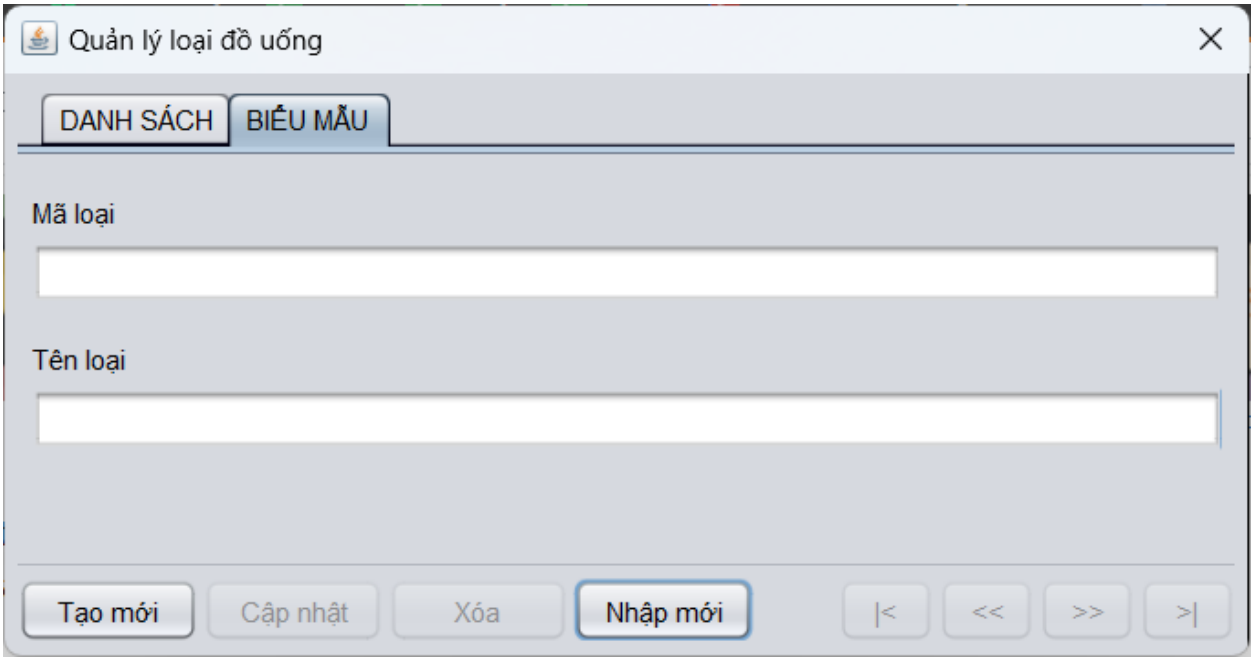
Các thành phần đã xây dựng trước (cần kiểm tra lại)

- Entity
- XJdbc/XQuery
- CrudDAO, CategoryDAO, CategoryDAOImpl

1. THIẾT KẾT GIAO DIỆN CATEGORYMANAGERJDIALOG



Hình 1: DANH SÁCH



Hình 2: BIỂU MẪU

ĐẶT TÊN CHO CÁC THÀNH PHẦN GIAO DIỆN

Text/Title	**Control**	**Name**
Quản lý loại đồ uống	JDialog	CategoryManager**JDialog**
DANH SÁCH/BIỂU MẪU	JTabbedPane	tabs

Mã loại	JTextField	txtId
Tên loại	JTextField	txtName
Bảng hiển thị loại đồ uống	JTable	tblCategories
Tạo mới	JButton	btnCreate
Cập nhật	JButton	btnUpdate
Xóa	JButton	btnDelete
Nhập mới	JButton	btnClear
|<	JButton	btnMoveFirst
<<	JButton	btnMovePrevious
>>	JButton	btnMoveNext
>|	JButton	btnMoveLast
Chọn tất cả	JButton	btnCheckAll
Bỏ chọn tất cả	JButton	btnUncheckAll
Xóa các mục chọn	JButton	btnDeleteCheckedItems

2. CRUDDAO (GENERIC)

```
public interface CrudController<Entity> {
   void open(); // Xử lý mở cửa sổ

   void setForm(Entity entity); // Hiển thị thực thể lên form
   Entity getForm(); // Tạo thực thể từ dữ liệu form

   void fillToTable(); // Tải dữ liệu và đổ lên bảng
   void edit(); // Hiển thị dữ liệu của hàng được chọn lên form

   void create(); // Tạo thực thể mới
   void update(); // Cập nhật thực thể đang xem
   void delete(); // Xóa thực thể đang xem
   void clear();  // Xóa trắng form
   void setEditable(boolean editable); // Thay đổi trạng thái form

   void checkAll(); // Tích chọn tất cả các hàng trên bảng
   void uncheckAll(); // Bỏ tích chọn tất cả các hàng trên bảng
   void deleteCheckedItems(); // Xóa các thực thể được tích chọn

   void moveFirst(); // Hiển thị thực thể đầu tiên
   void movePrevious(); // Hiển thị thực thể kế trước
   void moveNext(); // Hiển thị thực thể kế sau
```

```
    void moveLast(); // Hiển thị thực thể cuối cùng
    void moveTo(int rowIndex); // Hiển thị thực thể tại vị trí
}
```

3. CategoryController (specific)

```
public interface CategoryController extends CrudController<Category>{
}
```

4. Kết nối Controller vào giao diện

4.1 Cài đặt mã nguồn cho Controller

```
public class CategoryManagerJDialog extends JDialog implements CategoryController {
        …
        // cài đặt mã nguồn cho CategoryController như bên dưới
}
```

```
CategoryDAO dao = new CategoryDAOImpl();
List<Category> items = List.of();

@Override
public void open() {
   this.setLocationRelativeTo(null);
   this.fillToTable();
   this.clear();
}

@Override
public void fillToTable() {
   DefaultTableModel model = (DefaultTableModel) tblCategories.getModel();
   model.setRowCount(0);

   items = dao.findAll();
   items.forEach(item -> {
      Object[] rowData = {
         item.getId(),
         item.getName(),
         false
      };
      model.addRow(rowData);
   });
}

@Override
public void edit() {
   Category entity = items.get(tblCategories.getSelectedRow());
   this.setForm(entity);
   this.setEditable(true);
```

```
    tabs.setSelectedIndex(1);
}

@Override
public void checkAll() {
    this.setCheckedAll(true);
}

@Override
public void uncheckAll() {
    this.setCheckedAll(false);
}
private void setCheckedAll(boolean checked) {
    for (int i = 0; i < tblCategories.getRowCount(); i++) {
        tblCategories.setValueAt(checked, i, 2);
    }
}

@Override
public void deleteCheckedItems() {
    if (XDialog.confirm("Bạn thực sự muốn xóa các mục chọn?")) {
        for (int i = 0; i < tblCategories.getRowCount(); i++) {
            if ((Boolean) tblCategories.getValueAt(i, 2)) {
                dao.deleteById(items.get(i).getId());
            }
        }
        this.fillToTable();
    }
}

@Override
public void setForm(Category entity) {
    txtId.setText(entity.getId());
    txtName.setText(entity.getName());
}

@Override
public Category getForm() {
    Category entity = new Category();
    entity.setId(txtId.getText());
    entity.setName(txtName.getText());
    return entity;
}
```

```
@Override
public void create() {
   Category entity = this.getForm();
   dao.create(entity);
   this.fillToTable();
   this.clear();
}

@Override
public void update() {
   Category entity = this.getForm();
   dao.update(entity);
   this.fillToTable();
}

@Override
public void delete() {
   if (XDialog.confirm("Bạn thực sự muốn xóa?")) {
      String id = txtId.getText();
      dao.deleteById(id);
      this.fillToTable();
      this.clear();
   }
}

@Override
public void clear() {
   this.setForm(new Category());
   this.setEditable(false);
}

@Override
public void setEditable(boolean editable) {
   txtId.setEnabled(!editable);
   btnCreate.setEnabled(!editable);
   btnUpdate.setEnabled(editable);
   btnDelete.setEnabled(editable);

   int rowCount = tblCategories.getRowCount();
   btnMoveFirst.setEnabled(editable && rowCount > 0);
   btnMovePrevious.setEnabled(editable && rowCount > 0);
   btnMoveNext.setEnabled(editable && rowCount > 0);
   btnMoveLast.setEnabled(editable && rowCount > 0);
}
```

```
@Override
public void moveFirst() {
   this.moveTo(0);
}

@Override
public void movePrevious() {
   this.moveTo(tblCategories.getSelectedRow() - 1);
}

@Override
public void moveNext() {
   this.moveTo(tblCategories.getSelectedRow() + 1);
}

@Override
public void moveLast() {
   this.moveTo(tblCategories.getRowCount() - 1);
}

@Override
public void moveTo(int index) {
   if (index < 0) {
      this.moveLast();
   } else if (index >= tblCategories.getRowCount()) {
      this.moveFirst();
   } else {
      tblCategories.clearSelection();
      tblCategories.setRowSelectionInterval(index, index);
      this.edit();
   }
}
```

4.2 Gắn kết các event handler với các phương thức điều khiển

```
   private void formWindowOpened(java.awt.event.WindowEvent evt) {
      // TODO add your handling code here:
      this.open();
   }

   private void tblCategoriesMouseClicked(java.awt.event.MouseEvent evt) {
      // TODO add your handling code here:
      if (evt.getClickCount() == 2) {
         this.edit();
```

```
    }
}

private void btnCheckAllActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.checkAll();
}

private void btnUncheckAllActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.uncheckAll();
}

private void btnDeleteCheckedItemsActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.deleteCheckedItems();
}

private void btnCreateActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.create();
}

private void btnUpdateActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.update();
}

private void btnDeleteActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.delete();
}

private void btnClearActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.clear();
}

private void btnMoveFirstActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.moveFirst();
}

private void btnMovePreviousActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    // TODO add your handling code here:
    this.movePrevious();
  }

  private void btnMoveNextActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.moveNext();
  }

  private void btnMoveLastActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.moveLast();
  }
```