宠物商店管理系统

丁志宏　PB23071426

2025 年 6 月 3 日

1　实验题目

宠物商店管理系统

2　小组名单及分工

(a) [**学院**] 地球与空间科学学院

(a) [**学号**] PB23071426

(a) [**姓名**] 丁志宏

(a) [**工作内容**] 完整项目

(a) [**工作占比**] 100%

3　核心问题分析

系统需要解决以下关键问题：

- **用户管理**：实现账号的注册与注销，区分管理员和普通用户
- **权限控制**：不同用户类型（管理员/用户）拥有不同操作权限
- **宠物信息管理**：对宠物信息进行增删改查，支持多态设计
- **文件存储**：宠物信息存储在文件中，支持通过文件更新系统
- **订单管理**：管理员可管理所有订单（查看、修改状态等）
- **GUI 实现**：所有操作通过图形界面完成，界面友好并提供操作反馈

3.1　面向对象设计

3.1.1　类层次结构设计

类设计采用分层架构，满足重用和移植需求：

表 1: 系统类层次结构

层级	类名	功能描述
实体层	User	用户类，包含用户名、密码、用户类型等公共属性
	Admin	管理员类，可查看宠物、下单拥有管理宠物、订单权限
	Pet	抽象宠物类，定义名字、性别、品种等基本属性
	Dog	继承 Pet，添加特有属性（如吠叫级别）
	Cat	继承 Pet，添加特有属性（如毛型）
	Order	订单类，包含订单 ID、用户、宠物、状态等信息
数据访问层	FileUtils	文件操作工具类，使用泛型实现多种数据类型存储
用户视图层	PetInfoGUI	显示所有宠物名界面，并显示查询框
	PetDetailGUI	点击宠物名会跳转至此显示宠物详细信息界面并显示购买按钮
管理员视图层	PetAdminGUI	显示所有宠物名界面，并显示管理员宠物增删，以及订单管理按钮
	petoperate	显示宠物详细信息界面并管理员可进行修改宠物信息
	PetInforGUIAdmin	显示订单信息界面并管理员可进行修改订单信息
主函数层	PetStoreGUI	主函数运行辨别用户与管理员权限跳转至用户或管理员界面 GUI

3.1.2 多态应用实现

系统使用多态实现宠物信息的差异化展示：

Listing 1: 多态应用示例

```
// 宠物类定义
public class Pet implements Serializable {
    private String name;
    private String gender;
    private String breed;
    private int age;
    private static final long serialVersionUID = 1L;
    private String classification;
    private int price;
}

// 狗子类实现
public class Dog extends Pet implements Serializable{
    public Dog(String name, String gender, String breed, int age) {
        super(name, gender, breed, age);
    }
    public Dog(String name, String gender, String breed, int age,String classification,int price) {
        super(name, gender, breed, age,classification,price);
    }
    @Override
    public void displayInfo() {
        super.displayInfo();
        System.out.println("This pet is a Dog.");
    }
}

// 猫类实现
public class Cat extends Pet implements Serializable{
    public Cat(String name, String gender, String breed, int age) {
        super(name, gender, breed, age);
    }
    public Cat(String name, String gender, String breed, int age,String classification,int price) {
```

```
        super(name, gender, breed, age,classification,price);
    }
    @Override
    public void displayInfo() {
        super.displayInfo();
        System.out.println("This pet is a Cat.");
    }
}
```

3.2 文件存储设计

系统使用.csv 文件格式存储数据：

表 2: 文件存储设计

数据类型	存储路径	格式说明
宠物数据	/pets.csv	存储宠物信息，包含类型字段
订单数据	/orders.csv	序列化对象存储

文件操作工具类 FileUtils 支持多种数据类型：

Listing 2: 文件工具类 FileUtils

```
public class FileUtils {

    public static void savePetsToFile(List<Pet> pets) ;

    public static void saveOrderToFile(Order order);

    public static List<Order> loadOrdersFromFile();

    public static List<Pet> loadPetsFromFile();

    public static void saveAllOrdersToFile(List<Order> orders);

}
```

3.3 权限控制系统

系统采用分层权限控制策略：
关键实现代码：

Listing 3: 权限验证逻辑

```
loginButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String username = usernameField.getText();
                String password = new String(passwordField.getPassword());
                String userType = (String) userTypeCombo.getSelectedItem();

                for (User user : users) {
                    if (user.getUsername().equals(username) && user.getPassword().equals(password)) {
                        if (userType.equals("Admin") && user instanceof Admin) {
                            JOptionPane.showMessageDialog(frame, "Welcome, Admin!");
                            Admin admin = (Admin) user;
                            admin.managePets();
                            admin.viewOrders();
```

```
                            // 登录成功后关闭当前窗口，打开宠物管理界面
                            frame.dispose();
                            new PetAdminGUI(); // 打开宠物信息界面
                            return;
                        } else if (userType.equals("User") && user instanceof User) {
                            JOptionPane.showMessageDialog(frame, "Welcome, User!");
                            // 登录成功后关闭当前窗口，打开宠物信息界面
                            frame.dispose();
                            new PetInfoGUI(user); // 打开宠物信息界面
                            return;
                        }
                    }
                }
                JOptionPane.showMessageDialog(frame, "Invalid login credentials.");
            }
        });
```

3.4 GUI 实现方案

系统采用 Swing 框架实现 GUI，主要界面包括：

1. **登录界面**：支持管理员/用户选择
2. **管理员控制台**：
   - 宠物管理（表格形式展示，支持增删改）
   - 订单管理（订单列表，状态修改功能）
   - 文件维护（配置文件导入导出）
3. **用户界面**：
   - 宠物浏览（卡片视图，分类筛选）
   - 下单功能（宠物选择 → 加入购物车 → 支付）
   - 订单查看（个人历史订单）

结果回显机制采用统一的消息处理接口：

Listing 4: 操作结果反馈

```
public class NotificationUtil {
    // 显示成功消息
    public static void showSuccess(String message) {
        JOptionPane.showMessageDialog(null, message, "操作成功",
            JOptionPane.INFORMATION_MESSAGE);
    }

    // 显示错误消息
    public static void showError(String message) {
        JOptionPane.showMessageDialog(null, message, "操作失败",
            JOptionPane.ERROR_MESSAGE);
    }

    // 数据显示面板
    public static void showDataPanel(Component parent, String title,
            List<?> data) {
        JDialog dialog = new JDialog();
        dialog.setTitle(title);
```

```
19          // 创建表格或列表展示数据
20          dialog.add(createDataDisplay(data));
21          dialog.pack();
22          dialog.setVisible(true);
23      }
24  }
```

3.5 系统架构总结

系统采用分层架构设计，各层职责明确：

1. **模型层（Model）**：User, Pet, Order 等实体类定义
2. **数据访问层（DAO）**：FileUtil 文件操作封装
3. **业务逻辑层**：用户/宠物/订单管理服务
4. **视图层（View）**：Swing 界面组件及事件监听

架构设计满足以下要求：

- 重用性：类层次设计支持宠物类型扩展
- 移植性：文件存储与 GUI 分离，便于跨平台
- 多态应用：宠物信息展示差异化实现
- 静态方法：工具类中合理使用静态方法
- 泛型技术：文件操作支持多种数据类型

3.6 分层权限控制机制

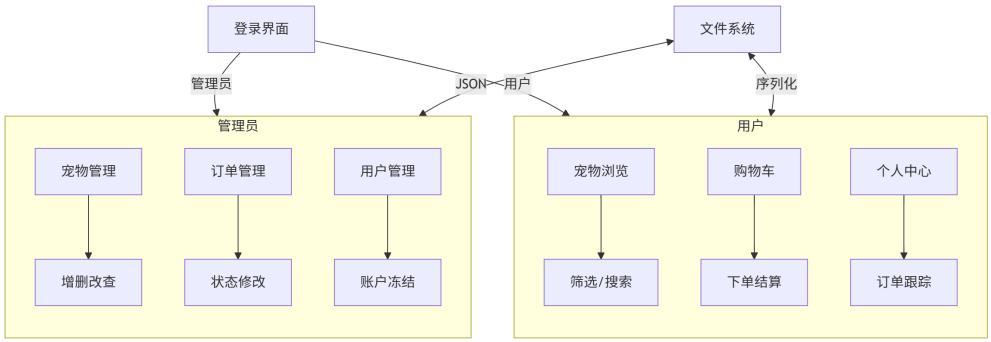

图 1: 分层权限控制机制

4 类关系说明

- **继承关系**:
  - `Admin` 和 `Customer` 继承自 `User`

  - Dog, Cat, Bird 继承自 Pet

- **关联关系**:
  - Order 关联 Customer 和 Pet
  - PetDetailGUI 关联特定 Pet 对象
- **依赖关系**:
  - 视图层依赖实体层获取数据（如 PetInfoGUI 依赖 Pet）
  - 视图层依赖数据访问层持久化数据（如 PetOperateGUI 依赖 FileUtils）
- **多态应用**:
  - Pet 类定义 getDetails() 抽象方法，不同子类实现各自的信息格式
  - FileUtils 使用泛型支持多种数据类型（用户、宠物、订单）的持久化
- **界面跳转流程**:
  1. PetStoreGUI 启动应用，显示登录界面
  2. 普通用户登录后进入 PetInfoGUI
     - 点击宠物列表项跳转到 PetDetailGUI
     - 点击购买按钮创建订单
  3. 管理员登录后进入 PetAdminGUI
     - 点击宠物操作按钮进入 PetOperateGUI
     - 点击订单管理按钮进入 OrderAdminGUI

5 运行结果展示

5.1 用户功能展示

5.2 管理员功能展示

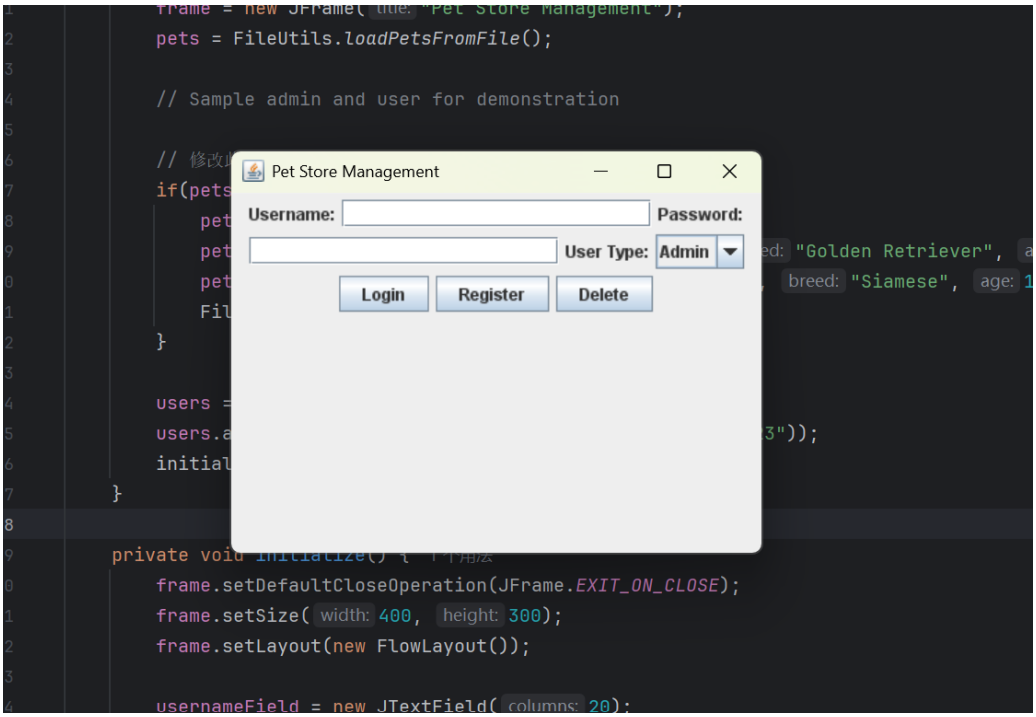

图 2: 登陆界面

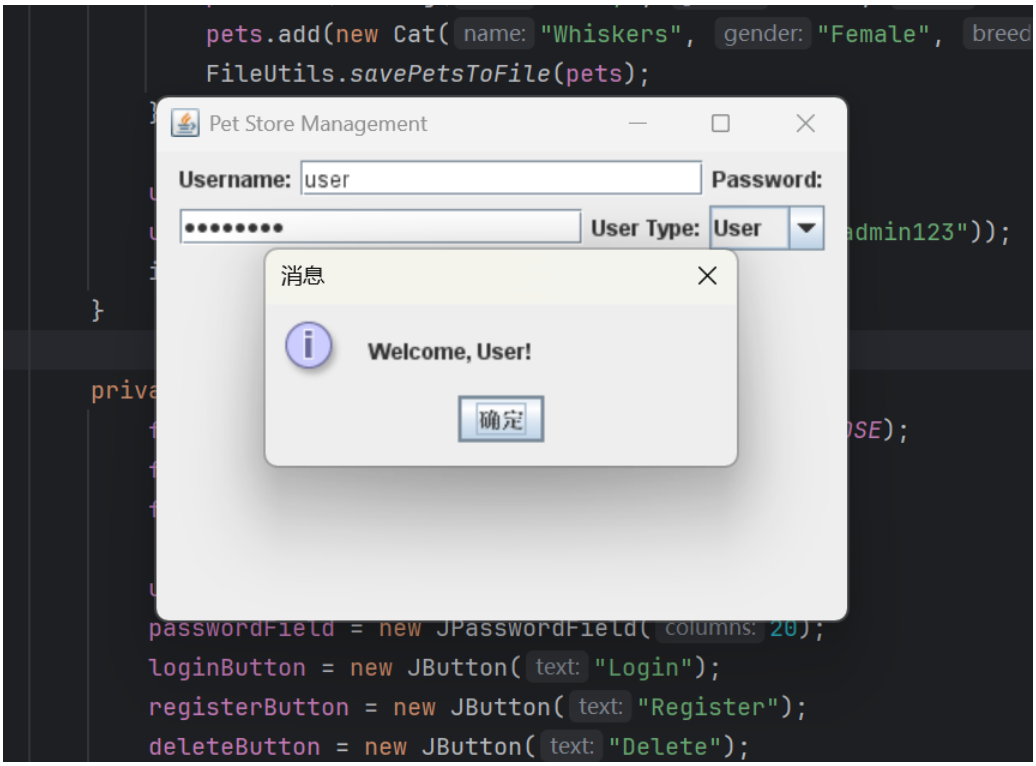

图 3: 用户登录

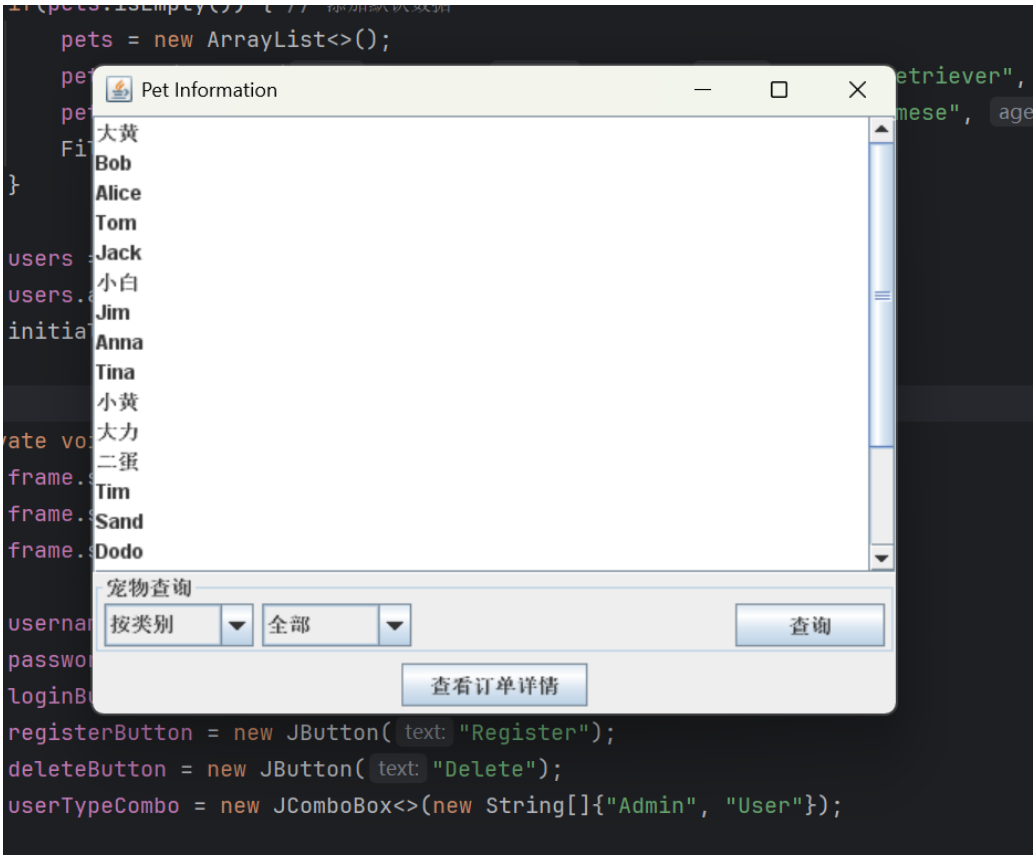

图 4: 用户界面

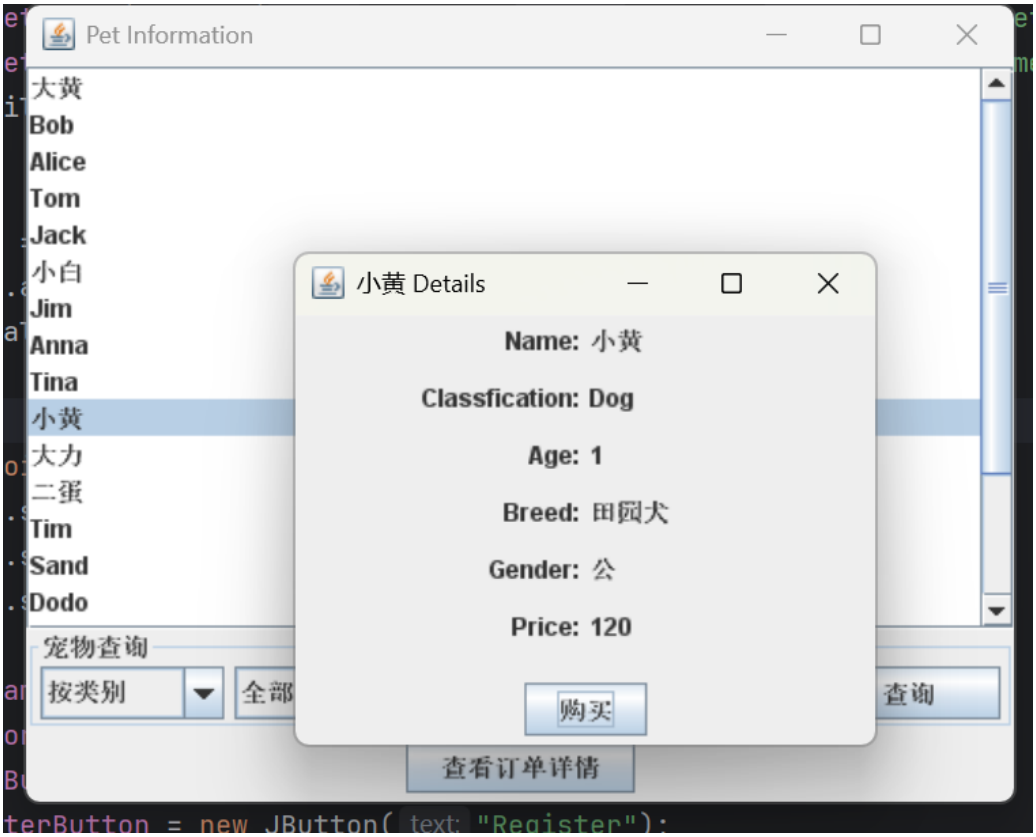

图 5: 用户宠物查询

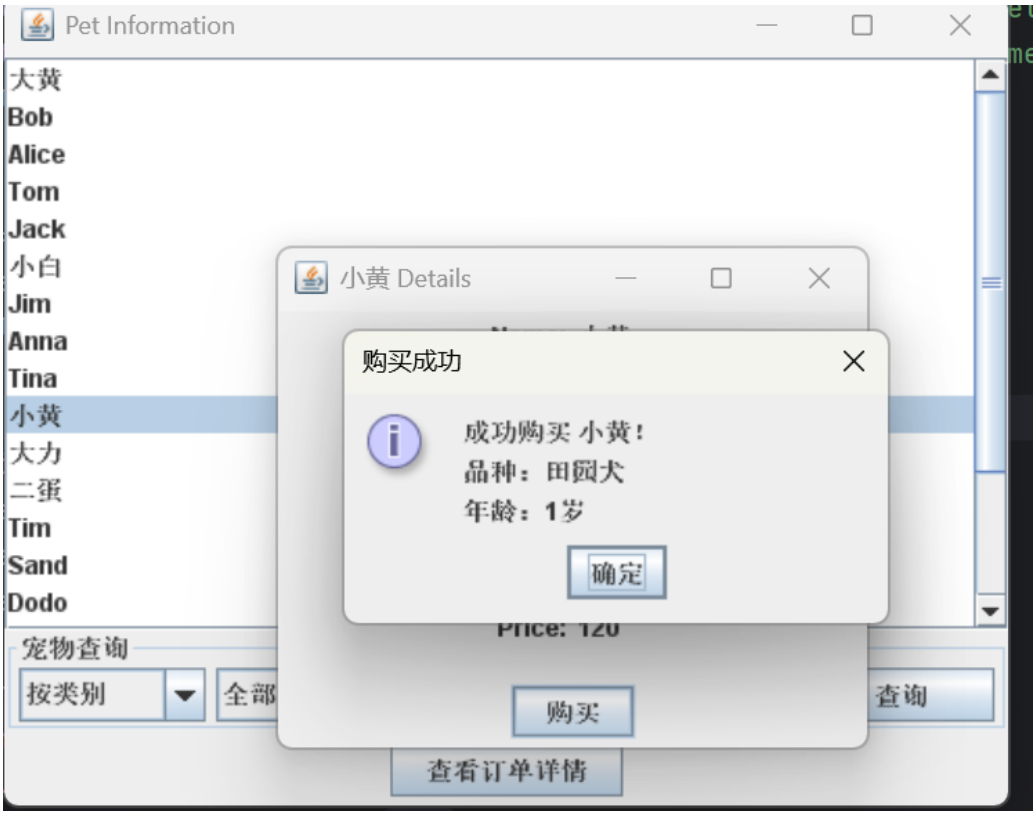

图 6: 宠物购买

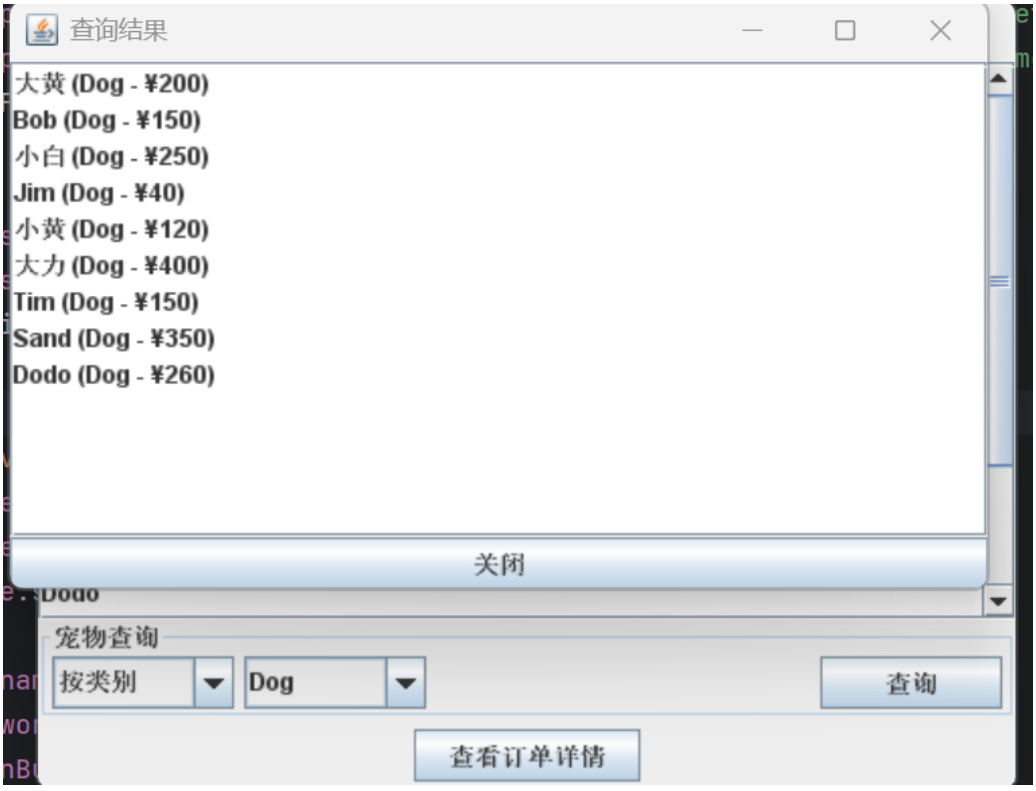

图 7: 宠物查询（按类别）

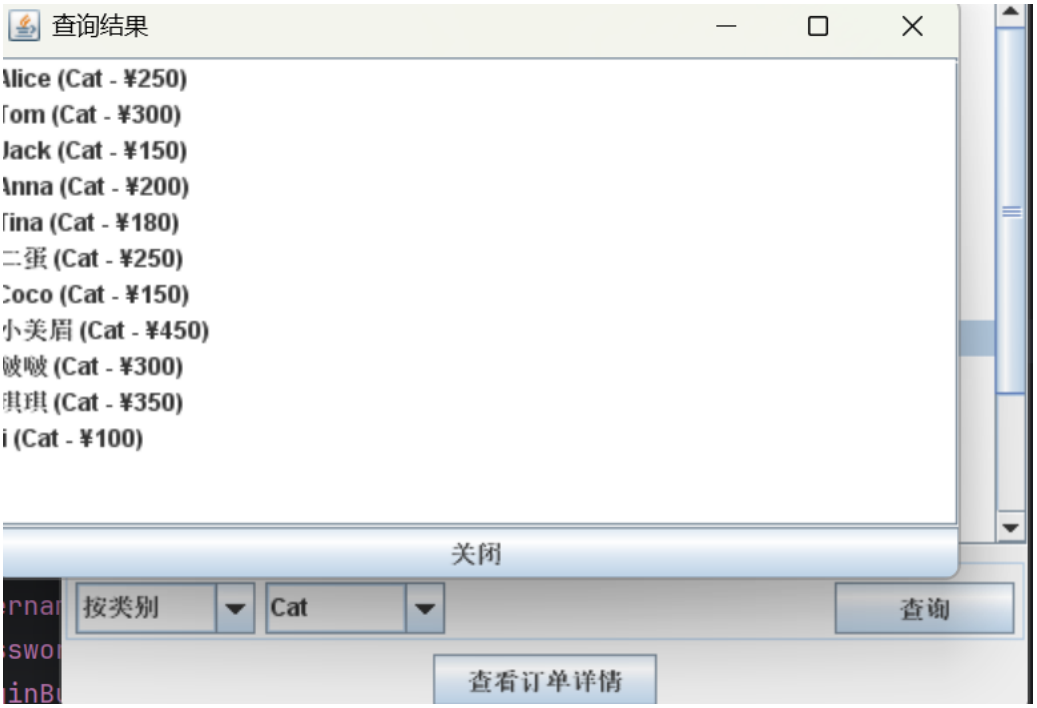

图 8: 宠物查询（按价格）

图 9: 用户订单信息

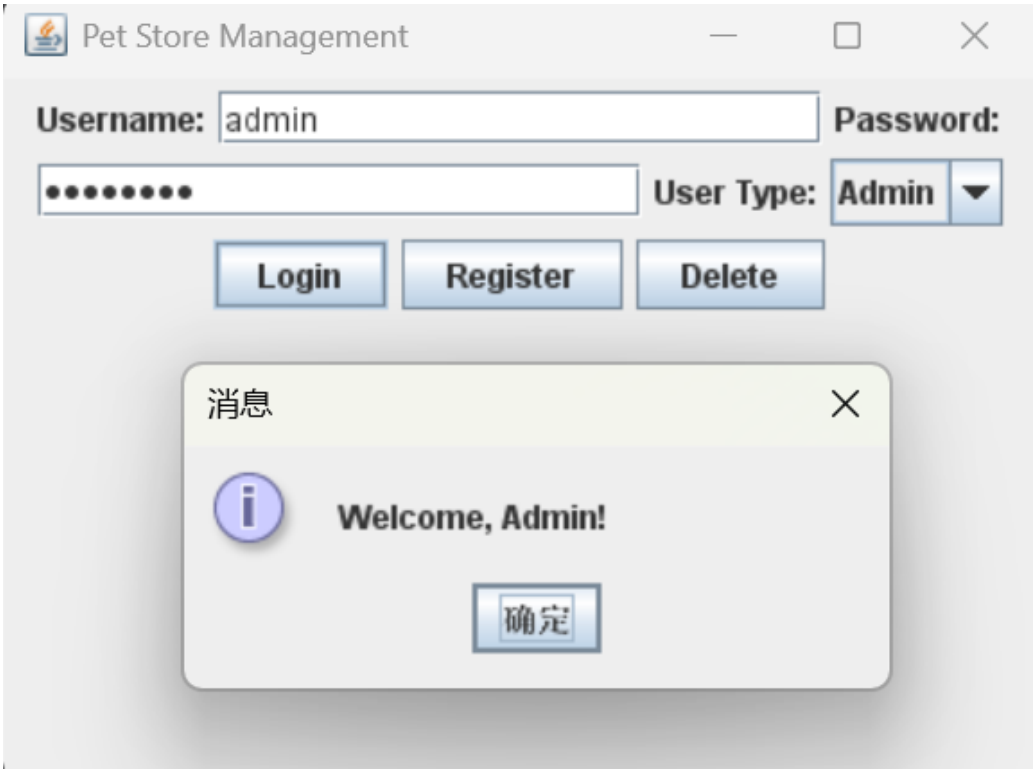

图 10: 管理员登陆

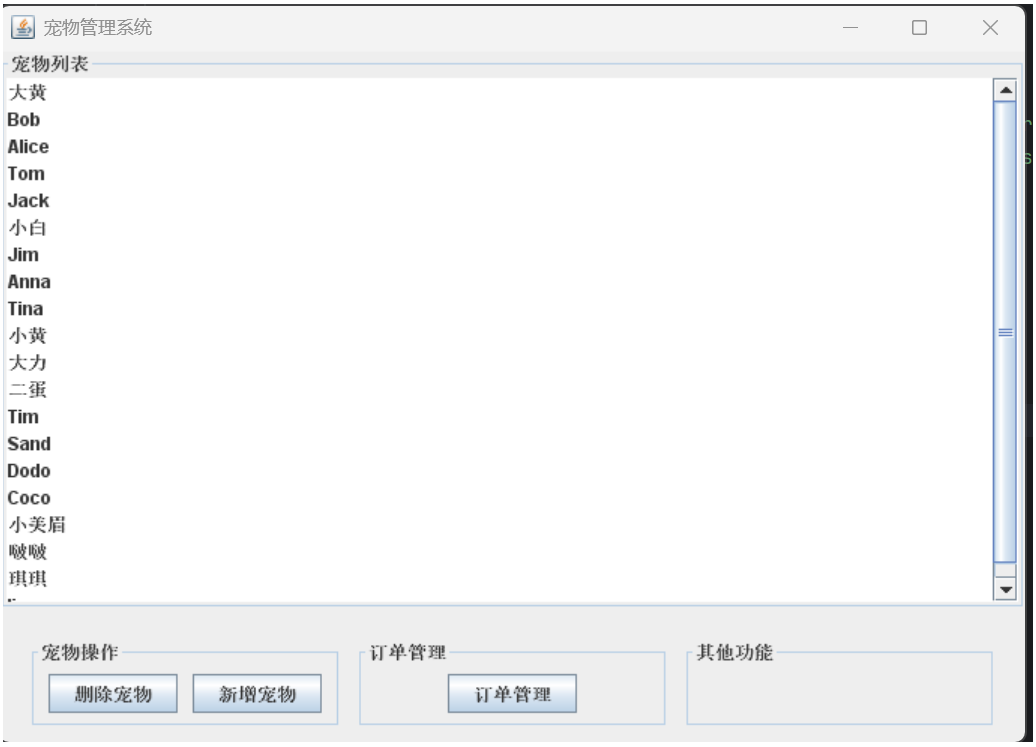

图 11: 管理员界面

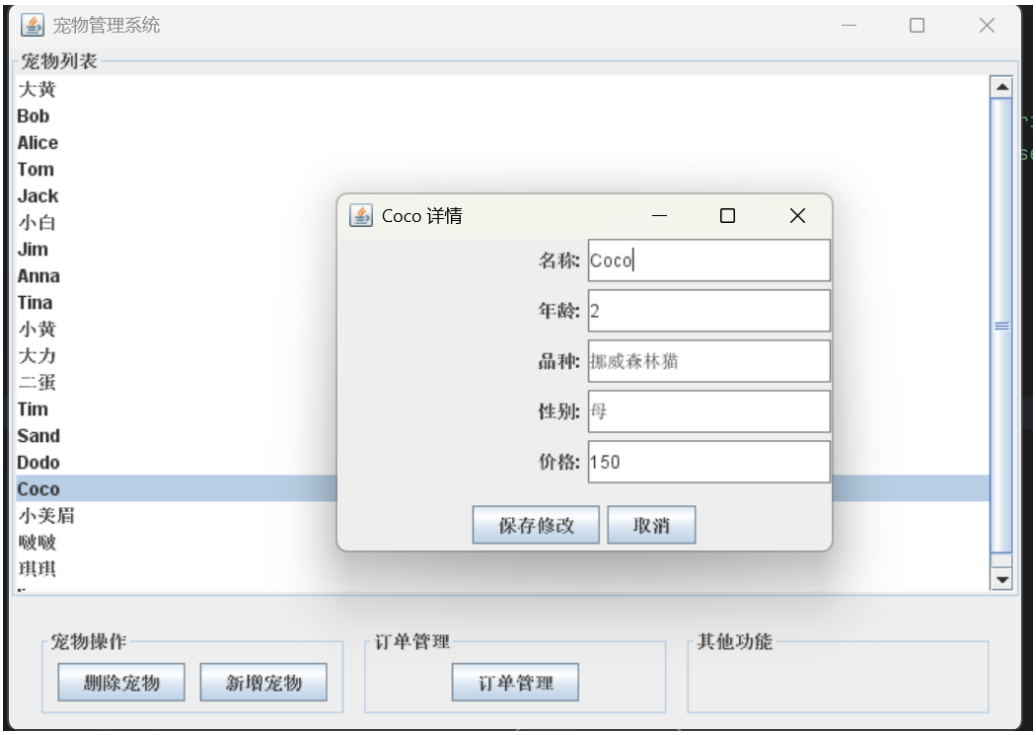

图 12: 宠物信息修改

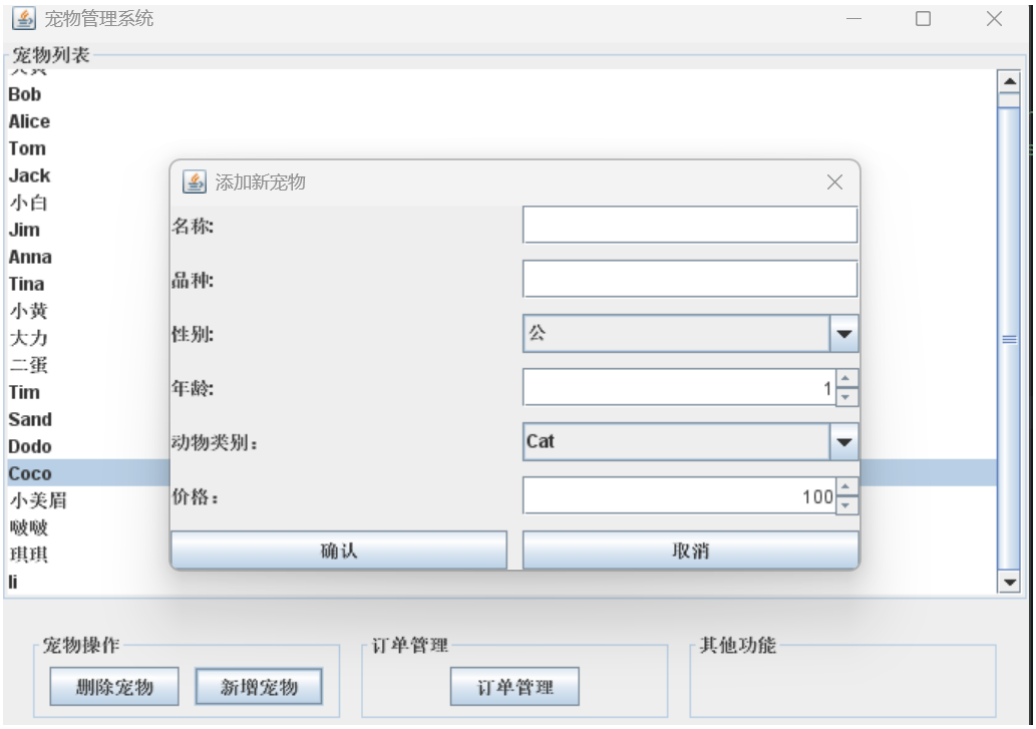

图 13: 宠物信息新增

图 14: 订单管理查询

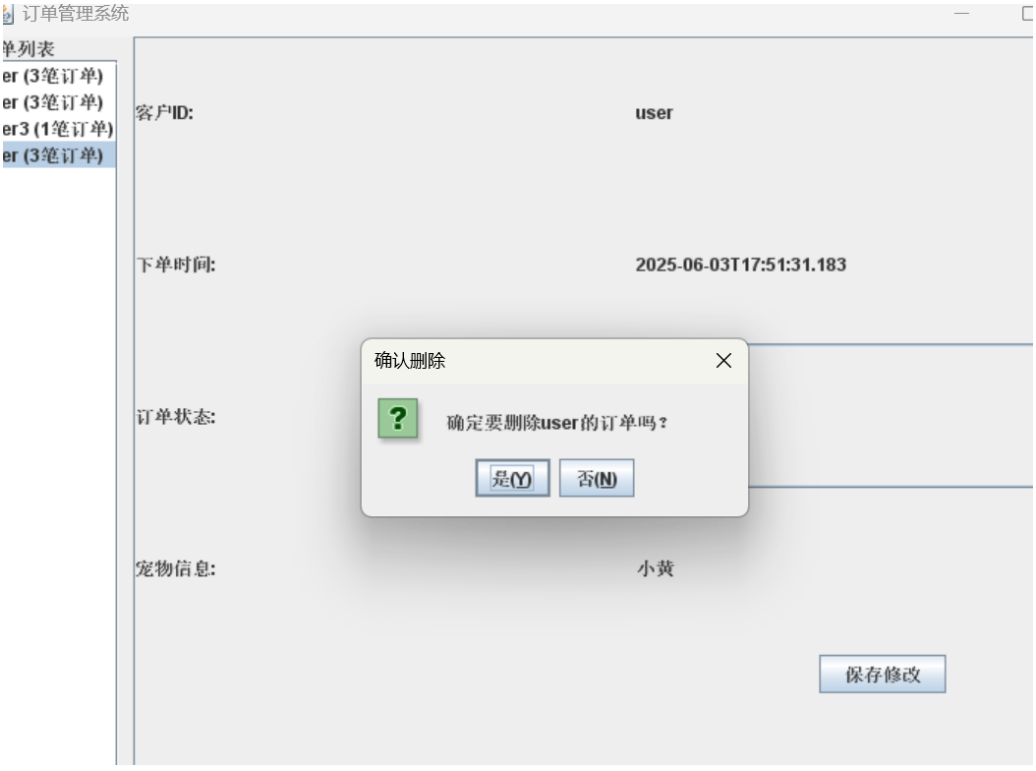

图 15: 订单修改

6 总结

宠物商店管理系统 GUI 实验项目总结

一、项目核心收获 Java GUI 开发能力提升

熟练掌握 Swing 组件库：JFrame, JTable, JList, JOptionPane 等

复杂布局设计：嵌套使用 BorderLayout、GridLayout、FlowLayout

事件处理机制：深入理解 ActionListener, MouseListener 等接口软件工程实践

MVC 模式落地：实现视图 (View)-控制器 (Controller)-模型 (Model) 分离

异常处理体系：建立健壮的 IO 操作和业务逻辑错误处理机制

数据持久化：设计 CSV 文件存储方案实现 CRUD 操作对象导向设计能力

领域模型抽象：User, Pet, Order 等实体类的合理封装

关注点分离：文件操作 (FileUtils) 与业务逻辑解耦

二、功能实现亮点用户管理模块

宠物展示系统

响应式宠物列表：带缩略信息的 JList 展示

智能搜索：种类/价格区间联合查询

订单管理核心

状态驱动 UI：不同订单状态对应不同操作按钮

安全取消机制：

三、克服的关键技术挑战 CSV 数据持久化难点

转义字符处理：解决, 和

跨对象关系管理

用户-订单-宠物关联：通过 ID 建立实体间关联关系

数据一致性：通过事务性文件写入保证复杂界面状态管理

动态 UI 组件：根据业务状态自动显隐组件

四、待改进方向架构优化

引入数据库替换文件存储

实现观察者模式实现数据-UI 同步更新用户体验增强

添加分页组件解决大数据加载问题

实现拖拽排序、收藏功能安全强化

五、项目价值提炼技术栈整合能力

将 OOP、GUI、文件 IO、异常处理等知识点融合应用

掌握从需求分析到可运行产品的完整开发流程工程化思维培养

版本控制：Git 管理项目演化

模块化设计：高内聚低耦合组件开发

防御式编程：校验所有外部输入

六、结论

本宠物商店管理系统的开发是一次完整的 Java GUI 应用开发实践。通过项目实现：深入掌握了 Swing 框架的核心组件和布局管理

构建了健壮的数据持久化方案和异常处理体系

实践了面向对象设计原则在实际项目中的应用

培养了完整软件开发生命周期的项目管理能力