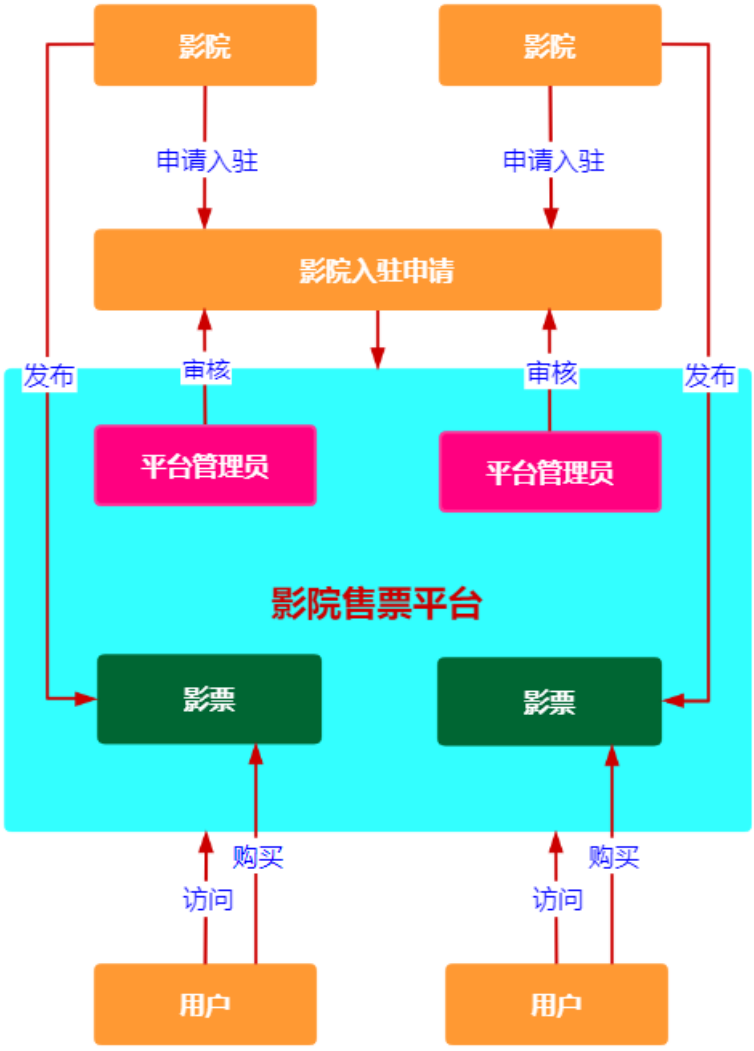
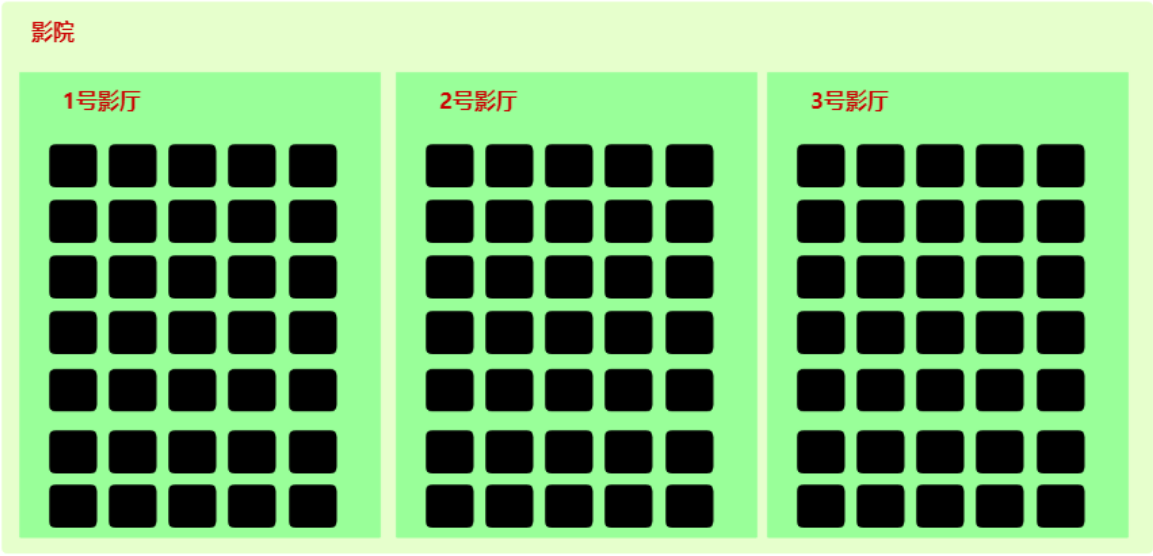
第三阶段项目-- 影院选票平台

需求分析

影院选票平台主要是为了解决影迷购票困难的问题，该平台集各大影院影票于一体，影迷可以很方便的在平台上购买影票。影院可以在申请入驻该平台，平台管理员对入驻申请进行审核，审核通过后，影院可以在该平台上发布影票售卖信息，随后，影迷可以在该平台上购买影票。平台结构如下：



影院结构如下：



影院在平台上可以管理该影院的影片信息、影厅信息、影片计划信息以及订单信息。

功能图

实体分析

1. 用户

用户分为管理员和普通用户。用户拥有账号、密码、安全码、角色和状态。其中账号唯一；账号和密码主要用于登录；安全码用于找回密码；角色用来区分普通用户和管理员；状态分为正常、冻结。

2. 商家

用户拥有账号、密码、安全码、角色、状态、影院名称和影院地址。其中账号唯一；账号和密码主要用于登录；安全码用于找回密码；角色用来区分商家和用户；状态分为待审核、正常、冻结。

3. 影片

影片具有编号、名称、主角、描述和所属商家。其中编号唯一；所属商家主要用来判断平台影片的归属者；其余信息只是用来展示

4. 座位

座位拥有排号、列号和所属用户。排号和列号决定座位所处位置；所属用户用来判定座位是否被订购。

5. 影厅

影厅拥有编号、名称、总排数、总列数、座位列表和所属商家。其中编号唯一，总排数和总列数决定了座位列表的大小，座位列表主要用于展示和计算余票，方便用户选购座位；所属商家主要用来判断平台影厅的归属者

6. 影片计划

影片计划拥有编号、使用影片、使用影厅、播放日期、开始时间、结束时间和所属商家。其中编号唯一；影片主要用于展示；开始时间和结束时间主要用于检测影片播放计划是否冲突；影厅主要用于计算余票；所属商家主要用来判断平台影片计划的归属者

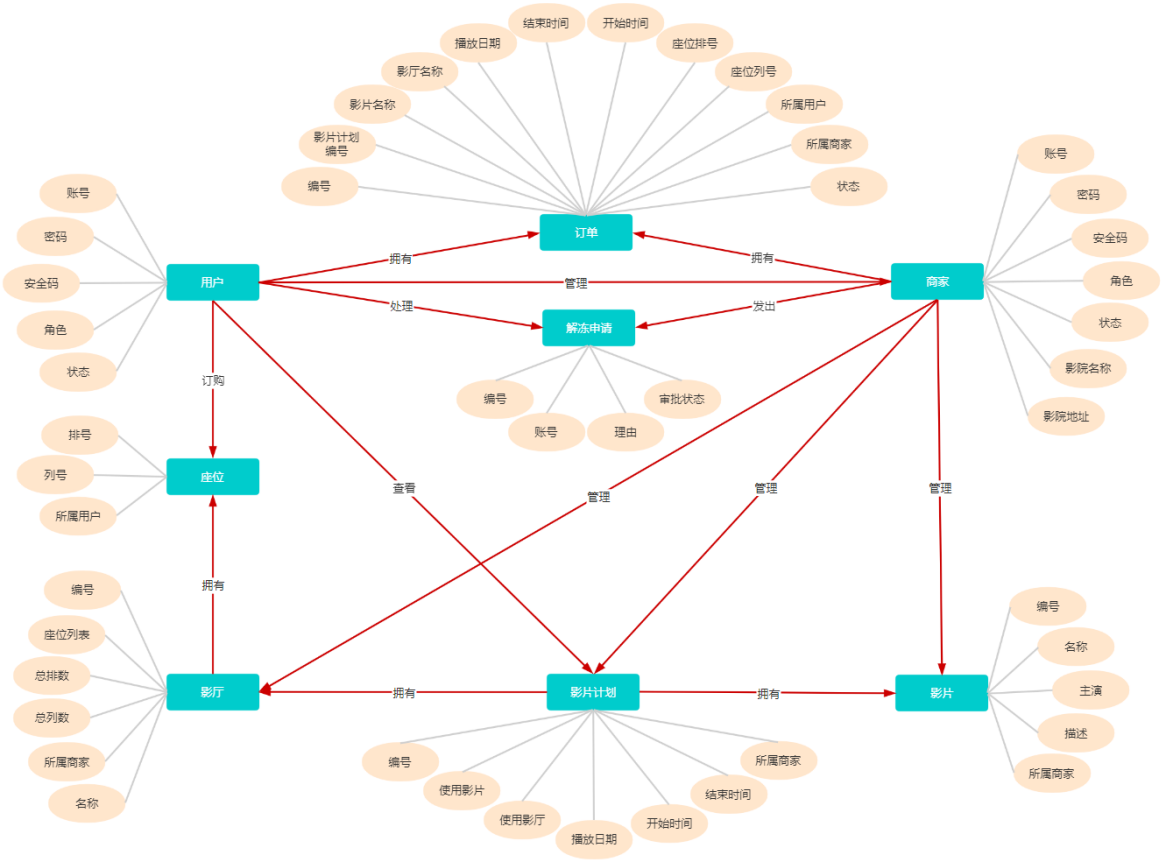
7. 订单

订单拥有编号、所属影片计划、影片名称、播放日期、开始时间、结束时间、座位排号、座位列号、所属用户、所属商家和状态。其中编号唯一；状态分为正常、已退订；其余信息只是用来展示

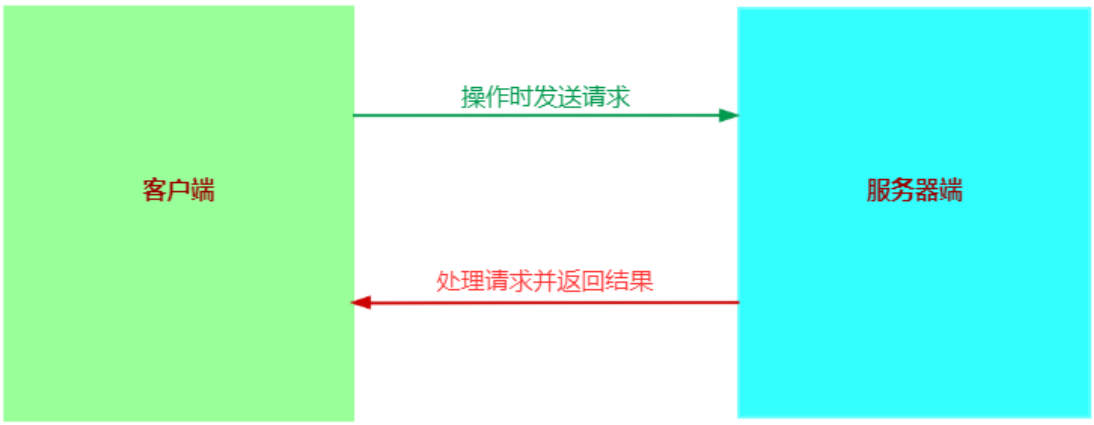
8. 解冻申请

解冻申请拥有编号、账号、理由和审批状态。其中编号唯一；账号主要用来确定解冻的用户；理由主要用来判定是否解冻；审批状态分为待处理、已通过、已驳回。

9. 实体关系图（ER图）



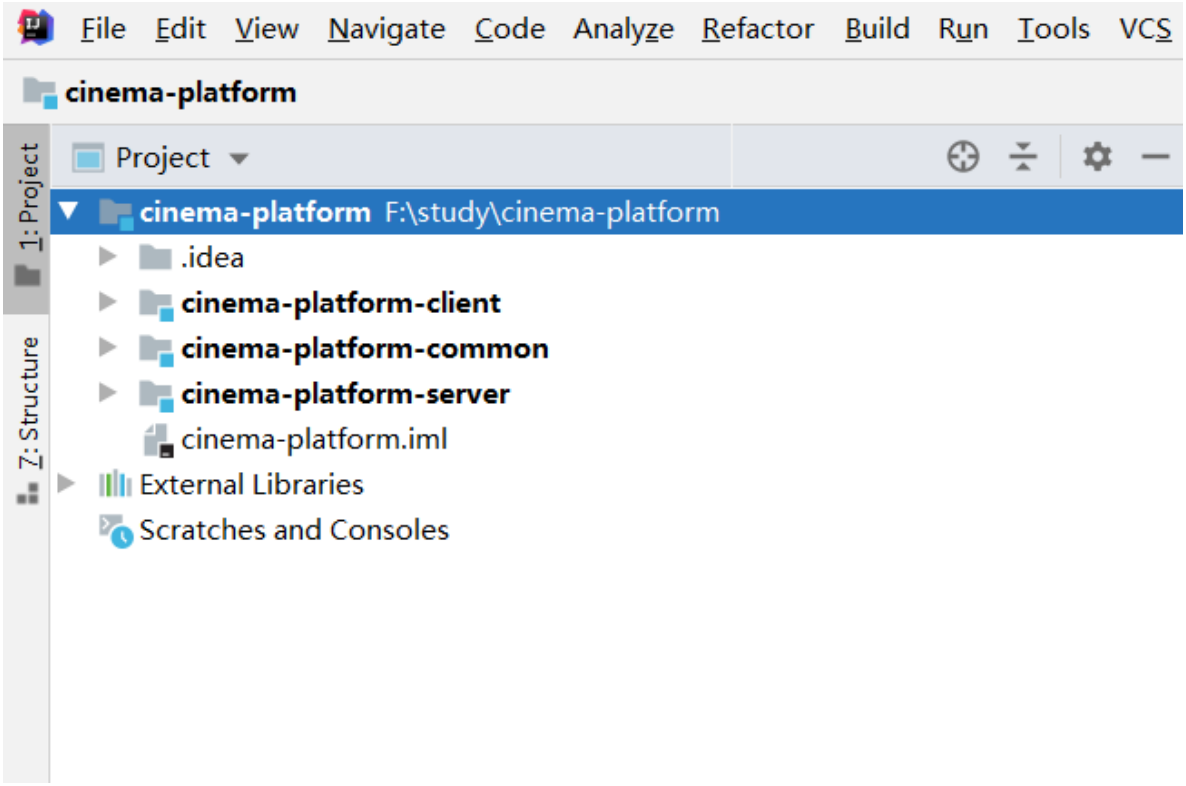
架构图



实现步骤

一、 C/S架构搭建

新建工程`cinema-platform`， 并在工程中分别创建`cinema-platform-client`，`cinema-platform-server`和`cinema-platform-common`三个模块。



其中`cinema-platform-client`为客户端，`cinema-platform-server`为服务器端，`cinema-platform-common`为客户端和服务器端共用的代码

二 `cinema-platform-common`设计

1. 实体构建

根据实体关系图将所有实体构建出来（此处贴出的代码省略了get/set方法）。

```java
package com.cyx.cinema.platform.entity;
//ER图 Entity Relational
//实体包常用包名：entity 实体 pojo(plain ordinary java object)简单java对象
//bo(business object) 业务对象 vo(view object) 视图对象 dto(data transfer object) 数据传输对象
//domain 领域模型  model 数据模型

/**
 * 用户
 */
public class User {
    /**
     * 用户角色：普通用户、管理员、商家
     */
    public enum Role{
        USER,MANAGER,SELLER
    }

    /**
     * 用户名
     */
    private String username;
    /**
     * 密码
     */
    private String password;
    /**
     * 安全码
     */
    private String securityCode;
    /**
     * 角色：默认为普通用户
     */
    private Role role = Role.USER;
    /**
     * 账号状态：-1-审核不通过，0-待审核，1-正常，2-冻结
     */
    private int state = 1;
}

package com.cyx.cinema.platform.entity;

/**
 * 商家
 */
public class Seller extends User {
    /**
     * 影院名称
     */
    private String cinemaName;
    /**
     * 影院地址
     */
    private String cinemaAddress;

}
package com.cyx.cinema.platform.entity;
```

```java

/**
 * 解冻申请
 */
public class UnfrozenApply {
    /**
     * 编号
     */
    private String id;
    /**
     * 账号
     */
    private String username;
    /**
     * 解冻理由
     */
    private String reason;
    /**
     * 解冻申请处理状态：0-待处理 1-已通过 2-已驳回
     */
    private int state;

}
package com.cyx.cinema.platform.entity;

/**
 * 影片
 */
public class Film {
    /**
     * 编号
     */
    private String id;
    /**
     * 名称
     */
    private String name;
    /**
     * 主演
     */
    private String actor;
    /**
     * 描述
     */
    private String description;
    /**
     * 拥有者
     */
    private String owner;
}
package com.cyx.cinema.platform.entity;

/**
 * 座位
 */
public class Seat {
    /**
     * 排号
```

```
     */
    private int row;
    /**
     * 列号
     */
    private int col;
    /**
     * 拥有者
     */
    private String owner;

}
package com.cyx.cinema.platform.entity;

/**
 * 影厅
 */
public class FilmHall {
    /**
     * 编号
     */
    private String id;
    /**
     * 名称
     */
    private String name;
    /**
     * 总排数
     */
    private int totalRow;
    /**
     * 总列数
     */
    private int totalCol;
    /**
     * 座位列表
     */
    private Seat[][] seats;
    /**
     * 拥有者
     */
    private String owner;
}
package com.cyx.cinema.platform.entity;

/**
 * 影片计划
 */
public class FilmPlan {
    /**
     * 编号
     */
    private String id;
    /**
     * 使用影片
     */
    private Film film;
    /**
```

```
     * 使用影厅
     */
    private FilmHall hall;
    /**
     * 播放日期
     */
    private String playDate;
    /**
     * 开始时间
     */
    private String beginTime;
    /**
     * 结束时间
     */
    private String endTime;
    /**
     * 拥有者
     */
    private Seller owner;
}
package com.cyx.cinema.platform.entity;

/**
 * 订单
 */
public class Order {
    /**
     * 编号
     */
    private String id;
    /**
     * 影片计划编号
     */
    private String filmPlanId;
    /**
     * 影片名称
     */
    private String filmName;
    /**
     * 影厅名称
     */
    private String filmHallName;
    /**
     * 播放日期
     */
    private String playDate;
    /**
     * 开始时间
     */
    private String beginTime;
    /**
     * 结束时间
     */
    private String endTime;
    /**
     * 排号
     */
    private int row;
```

```
    /**
     * 列号
     */
    private int col;
    /**
     * 订单所属用户
     */
    private String user;
    /**
     * 订单所属商家
     */
    private Seller seller;
    /**
     * 订单状态：0-已退订 1-正常
     */
    private int state = 1;
}
```

2. ID生成器

大多数实体都有唯一编号，这些唯一的编号都是以字符串的方式出现，需要使用ID生成器来生成

```
package com.cyx.cinema.platform.util;
//工具类包名util

import java.util.Random;

/**
 * ID生成器
 */
public class IdGenerator {

    public static void main(String[] args) {
       for(int i=0; i<10; i++){
           String id = generateId(10);
           System.out.println(id);
       }
    }

    private static char[] characters = {
            'A','B','C','D','E','F','G','H','I',
            'J','K','L','M','N','O','P','Q','R',
            'S','T','U','V','W','X','Y','Z','0',
            '1','2','3','4','5','6','7','8','9'
    };

    /**
     * 生成指定长度的ID
     * @param length
     * @return
     */
    public static String generateId(int length){
        Random r = new Random();
        StringBuilder builder = new StringBuilder("CYX_");
        for(int i=0; i<length; i++){
            int index = r.nextInt(characters.length);
            builder.append(characters[index]);
```

```
        }
        return builder.toString();
    }

}
```

3. 信息交互分析

为了保证信息传输的可靠性，这里选用套接字Socket来实现可靠性信息传输。而Socket实现信息传输使用的是IO流通道进行，对于IO流，我们学过字节流、字符流、二进制流、对象流等，选用哪一种IO流来进行信息传输呢？在影院选票系统中，用户可以查看影片信息、影片播放计划、用户解冻申请信息等，这些信息都是以列表的形式展现。如果使用字节流和二进制流来实现系信息传输，在实现上存在诸多限制，因此首先排除。如果使用字符流来实现信息传输，那么接收端就必须对字符串进行解析，而字符串解析开销比较大，性能较为低下，所以也排除。因此，只能选择对象流来实现信息传输，但需要注意，**传输的对象必须实现序列化接口**

在影院选票系统中，客户端每次发送的信息必须包含用户的动作以及该动作携带的数据，服务器接到信息后，首先从信息中提取用户动作，然后再根据用户动作对数据进行处理。

```java
package com.cyx.cinema.platform.message;

import java.io.Serializable;

/**
 * 消息
 * @param <T>
 */
public class Message<T> implements Serializable {
    /**
     * 操作命令
     */
    private int command;
    /**
     * 发送的数据
     */
    private T data;

    public Message(int command, T data) {
        this.command = command;
        this.data = data;
    }

    public int getCommand() {
        return command;
    }

    public T getData() {
        return data;
    }

    @Override
    public String toString() {
        return command + "=>" + data;
    }
}
```

客户端与服务器端会进行频繁的交互，为了避免重复的编码，这些交互应该编写在一个工具类中。

```java
package com.cyx.cinema.platform.util;

import java.io.*;
import java.net.Socket;

/**
 * 套接字工具类
 */
public class SocketUtil {
    /**
     * 发送消息
     * @param socket
     * @param msg
     * @param <T>
     */
    public static <T> void sendMsg(Socket socket, T msg){
        try {
            //获取输出流
            OutputStream os = socket.getOutputStream();
            //将输出流包装为对象输出流
            ObjectOutputStream oos = new ObjectOutputStream(os);
            //对象输出流写对象，也就是发送消息
            oos.writeObject(msg);
            //强制将通道中的数据刷出
            oos.flush();
            //告知接收端，消息发送已经完毕
            socket.shutdownOutput();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    /**
     * 接收消息
     * @param socket
     * @param <T>
     * @return
     */
    public static <T> T receiveMsg(Socket socket){
        try {
            //获取输入流
            InputStream is = socket.getInputStream();
            //将输入流包装为对象输入流
            ObjectInputStream ois = new ObjectInputStream(is);
            //读取对象
            T t = (T) ois.readObject();
            //告诉发送端，信息读取已经完毕
            socket.shutdownInput();
            return t;
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

`cinema-platform-client`中添加新的消息发送器

```
package com.cinema.platform.client.interact;

import com.cyx.cinema.platform.message.Message;
import com.cyx.cinema.platform.util.SocketUtil;

import java.io.IOException;
import java.net.Socket;

/**
 * 消息发送器
 */
public class MessageSender {

    private static final String IP = "localhost";

    private static final int PORT = 8888;

    /**
     * 客户端发送消息
     * @param command
     * @param data
     * @param <T>
     * @param <V>
     * @return
     */
    public static <T,V> T sendMsg(int command, V data){
        try {
            Socket socket = new Socket(IP, PORT);
            Message<V> msg = new Message<>(command, data);
            SocketUtil.sendMsg(socket, msg);
            return SocketUtil.receiveMsg(socket);
        } catch (IOException e) {
            e.printStackTrace();
            return null;
        }
    }

}
```

三 `cinema-platform-client`设计

1. 用户行为分析

所有的实体都与用户息息相关，用户的行为决定了实体信息的变更。根据功能图将用户的所有行为全部构建出来。

```
package com.cinema.platform.client.action;

/**
 * 用户行为
 */
public class UserAction {
```

```
    /**
     * 注册
     */
    public static void register(){}
    /**
     * 登录
     */
    public static void login(){}
    /**
     * 入驻申请
     */
    public static void entryApply(){}
    /**
     * 查案入驻申请
     */
    public static void viewEntryApply(){}
    /**
     * 找回密码
     */
    public static void getPasswordBack(){}
    /**
     * 解冻申请
     */
    public static void unfrozenApply(){}
    /**
     * 查看商家
     */
    public static void viewSellers(){}
    /**
     * 审核商家
     */
    public static void auditSeller(){}
    /**
     * 冻结商家
     */
    public static void frozenSeller(){}
    /**
     * 查看用户
     */
    public static void viewUsers(){}
    /**
     * 冻结用户
     */
    public static void frozenUser(){}
    /**
     * 查看解冻申请
     */
    public static void viewUnfrozenApplies(){}
    /**
     * 审批解冻申请
     */
    public static void auditUnfrozenApply(){}
    /**
     * 查看影厅
     */
    public static void viewFilmHalls(){}
    /**
     * 添加影厅
```

```
     */
    public static void addFilmHall(){}
    /**
     * 修改影厅
     */
    public static void updateFilmHall(){}
    /**
     * 删除影厅
     */
    public static void deleteFilmHall(){}
    /**
     * 查看影片
     */
    public static void viewFilms(){}
    /**
     * 添加影片
     */
    public static void addFilm(){}
    /**
     * 修改影片
     */
    public static void updateFilm(){}
    /**
     * 删除影片
     */
    public static void deleteFilm(){}
    /**
     * 商家查看影片计划
     */
    public static void viewSellerFilmPlans(){}
    /**
     * 添加影片计划
     */
    public static void addFilmPlan(){}
    /**
     * 修改影片计划
     */
    public static void updateFilmPlan(){}
    /**
     * 删除影片计划
     */
    public static void deleteFilmPlan(){}
    /**
     * 查看订单
     */
    public static void viewOrders(){}
    /**
     * 用户查看影片计划
     */
    public static void viewUserFilmPlans(){}
    /**
     * 在线订座
     */
    public static void orderSeatOnline(){}
    /**
     * 修改订单
     */
    public static void updateOrder(){}
```

```
    /**
     * 取消订单
     */
    public static void cancelOrder(){}
}
```

2. 菜单分析

由于是控制台系统，功能图中的所有功能只能够以菜单的形式展示给用户。因此，需要对菜单进行设计。

菜单拥有编号、名称、触发的命令、子菜单列表。编号用于用户选择；名称用于展示；触发的命令决定用户的动作；子菜单列表主要用于选择该菜单后进行展示。

```
package com.cinema.platform.client.menu;

import java.util.ArrayList;
import java.util.List

/**
 * 菜单
 */
public class Menu {
    /**
     * 序号
     */
    private int order;
    /**
     * 名称
     */
    private String name;
    /**
     * 菜单命令
     */
    private int command;
    /**
     * 子菜单列表
     */
    private List<Menu> children = new ArrayList<>();

    public Menu(int order, String name, int command) {
        this.order = order;
        this.name = name;
        this.command = command;
    }

}
```

菜单分为登录菜单和主菜单，主菜单又分为普通用户主菜单、商家主菜单和管理员主菜单。因此，菜单应该被管理起来，方便使用

```
package com.cinema.platform.client.menu;

/**
 * 菜单管理器
 */
```

```
public class MenuManager {
    /**
     * 登录菜单
     */
    public static final Menu[] LOGIN_MENUS = {};
    /**
     * 用户菜单
     */
    public static final Menu[] USER_MENUS = {};
    /**
     * 管理员菜单
     */
    public static final Menu[] MANAGER_MENUS = {};
    /**
     * 商家菜单
     */
    public static final Menu[] SELLER_MENUS = {};

}
```

构建菜单对象时需要触发的命令，当客户端向服务器端发送该命令时，服务器端需要接收该命令，然后执行相应的处理。因此命令是服务器端和客户端共同使用。在`cinema-platform-common`工程添加`Command`接口

```
package com.cyx.cinema.platform.command;

/**
 * 命令接口
 */
public interface Command {
    //0123456789  9+1=>10
    //1+1 => 10
    //7+1 => 10
    //0123456789a(10)b(11)c(12)d(13)e(14)f(15)
    //15(f)+1 => 10
    /**
     * 注册
     */
    int REGISTER = 0x00;
    /**
     * 登录
     */
    int LOGIN = 0x01;
    /**
     * 入驻申请
     */
    int ENTRY_APPLY = 0x02;
    /**
     * 查看入驻申请
     */
    int VIEW_ENTRY_APPLY = 0x03;
    /**
     * 找回密码
     */
    int GET_PASSWORD_BACK = 0x04;
    /**
     * 申请解冻
```

```
     */
    int UNFROZEN_APPLY = 0x05;
    /**
     * 退出
     */
    int QUIT = 0x06;
    /**
     * 显示子菜单
     */
    int SHOW_CHILDREN = 0x07;
    /**
     * 查看订单
     */
    int VIEW_ORDERS = 0x08;
    /**
     * 修改订单
     */
    int UPDATE_ORDER = 0x09;
    /**
     * 取消订单
     */
    int CANCEL_ORDER = 0x0a;
    /**
     * 返回主菜单
     */
    int GO_BACK_MAIN = 0x0b;
    /**
     * 查看影片计划
     */
    int VIEW_USER_FILM_PLANS = 0x0c;
    /**
     * 在线订座
     */
    int ORDER_SEAT_ONLINE = 0x0d;
    /**
     * 查看商家
     */
    int VIEW_SELLERS = 0x0c;
    /**
     * 审核商家
     */
    int AUDIT_SELLER = 0x0d;
    /**
     * 冻结商家
     */
    int FROZEN_SELLER = 0x0e;
    /**
     * 返回登录
     */
    int GO_BACK_LOGIN = 0x0f;
    /**
     * 查看用户
     */
    int VIEW_USERS = 0x10;
    /**
     * 冻结用户
     */
    int FROZEN_USER = 0x11;
```

```
    /**
     * 查看解冻申请
     */
    int VIEW_UNFROZEN_APPLIES = 0x12;
    /**
     * 审批解冻申请
     */
    int AUDIT_UNFROZEN_APPLY = 0x13;
    /**
     * 查看影厅
     */
    int VIEW_FILM_HALLS = 0x14;
    /**
     * 添加影厅
     */
    int ADD_FILM_HALL = 0x15;
    /**
     * 修改影厅
     */
    int UPDATE_FILM_HALL = 0x16;
    /**
     * 删除影厅
     */
    int DELETE_FILM_HALL = 0x17;
    /**
     * 查看影片
     */
    int VIEW_FILMS = 0x18;
    /**
     * 添加影片
     */
    int ADD_FILM = 0x19;
    /**
     * 修改影片
     */
    int UPDATE_FILM = 0x1a;
    /**
     * 删除影片
     */
    int DELETE_FILM = 0x1b;
    /**
     * 查看商家影片计划
     */
    int VIEW_SELLER_FILM_PLANS = 0x1c;
    /**
     * 添加影片计划
     */
    int ADD_FILM_PLAN = 0x1d;
    /**
     * 更新影片计划
     */
    int UPDATE_FILM_PLAN = 0x1e;
    /**
     * 删除影片计划
     */
    int DELETE_FILM_PLAN = 0x1f;
}
```

菜单类中应该添加子菜单的方法，然后完善菜单管理器中菜单内容

```
package com.cinema.platform.client.menu;

import com.cyx.cinema.platform.command.Command;

import java.util.Arrays;

/**
 * 菜单管理器
 */
public class MenuManager {
    /**
     * 登录菜单
     */
    public static final Menu[] LOGIN_MENUS = {
            new Menu(1, "注册", Command.REGISTER),
            new Menu(2, "登录", Command.LOGIN),
            new Menu(3, "入驻申请", Command.ENTRY_APPLY),
            new Menu(4, "查看入驻申请", Command.VIEW_ENTRY_APPLY),
            new Menu(5, "找回密码", Command.GET_PASSWORD_BACK),
            new Menu(6, "申请解冻", Command.UNFROZEN_APPLY),
            new Menu(7, "退出", Command.QUIT),
    };
    /**
     * 用户菜单
     */
    public static final Menu[] USER_MENUS;
    static {
            Menu menu1 = new Menu(1, "我的订单", Command.SHOW_CHILDREN);
            menu1.addChild(new Menu(1, "查看订单", Command.VIEW_ORDERS));
            menu1.addChild(new Menu(2, "修改订单", Command.UPDATE_ORDER));
            menu1.addChild(new Menu(3, "取消订单", Command.CANCEL_ORDER));
            menu1.addChild(new Menu(4, "返回主菜单", Command.GO_BACK_MAIN));

            Menu menu2 = new Menu(2, "购买影票", Command.SHOW_CHILDREN);
            menu2.addChild(new Menu(1, "查看影片计划",
Command.VIEW_USER_FILM_PLANS));
            menu2.addChild(new Menu(2, "在线订座",
Command.ORDER_SEAT_ONLINE));
            menu2.addChild(new Menu(3, "返回主菜单", Command.GO_BACK_MAIN));

            Menu menu3 = new Menu(3, "返回登录", Command.REGISTER);

            USER_MENUS = new Menu[]{menu1, menu2, menu3};
    };
    /**
     * 管理员菜单
     */
    public static final Menu[] MANAGER_MENUS;
    static {
        Menu menu1 = new Menu(1, "商家管理", Command.SHOW_CHILDREN);
        menu1.addChild(new Menu(1, "查看商家", Command.VIEW_SELLERS));
        menu1.addChild(new Menu(2, "审核商家", Command.AUDIT_SELLER));
        menu1.addChild(new Menu(3, "冻结商家", Command.FROZEN_SELLER));
        menu1.addChild(new Menu(4, "返回主菜单", Command.GO_BACK_MAIN));

```

```
        Menu menu2 = new Menu(2, "用户管理", Command.SHOW_CHILDREN);
        menu2.addChild(new Menu(1, "查看用户", Command.VIEW_USERS));
        menu2.addChild(new Menu(2, "冻结用户", Command.FROZEN_USER));
        menu2.addChild(new Menu(3, "返回主菜单", Command.GO_BACK_MAIN));

        Menu menu3 = new Menu(3, "解冻申请管理", Command.SHOW_CHILDREN);
        menu3.addChild(new Menu(1, "查看解冻申请",
Command.VIEW_UNFROZEN_APPLIES));
        menu3.addChild(new Menu(2, "审批解冻申请",
Command.AUDIT_UNFROZEN_APPLY));
        menu3.addChild(new Menu(3, "返回主菜单", Command.GO_BACK_MAIN));

        Menu menu4 = new Menu(4, "返回登录", Command.GO_BACK_LOGIN);
        MANAGER_MENUS = new Menu[]{menu1, menu2, menu3, menu4};
    }
    /**
     * 商家菜单
     */
    public static final Menu[] SELLER_MENUS;
    static {
        Menu menu1 = new Menu(1, "影厅管理", Command.SHOW_CHILDREN);
        menu1.addChild(new Menu(1, "查看影厅", Command.VIEW_FILM_HALLS));
        menu1.addChild(new Menu(2, "增加影厅", Command.ADD_FILM_HALL));
        menu1.addChild(new Menu(3, "修改影厅", Command.UPDATE_FILM_HALL));
        menu1.addChild(new Menu(4, "删除影厅", Command.DELETE_FILM_HALL));
        menu1.addChild(new Menu(5, "返回主菜单", Command.GO_BACK_MAIN));

        Menu menu2 = new Menu(2, "影片管理", Command.SHOW_CHILDREN);
        menu2.addChild(new Menu(1, "查看影片", Command.VIEW_FILMS));
        menu2.addChild(new Menu(2, "增加影片", Command.ADD_FILM));
        menu2.addChild(new Menu(3, "修改影片", Command.UPDATE_FILM));
        menu2.addChild(new Menu(4, "删除影片", Command.DELETE_FILM));
        menu2.addChild(new Menu(5, "返回主菜单", Command.GO_BACK_MAIN));

        Menu menu3 = new Menu(3, "影片计划管理", Command.SHOW_CHILDREN);
        menu3.addChild(new Menu(1, "查看影片计划",
Command.VIEW_SELLER_FILM_PLANS));
        menu3.addChild(new Menu(2, "增加影片计划", Command.ADD_FILM_PLAN));
        menu3.addChild(new Menu(3, "修改影片计划",
Command.UPDATE_FILM_PLAN));
        menu3.addChild(new Menu(4, "删除影片计划",
Command.DELETE_FILM_PLAN));
        menu3.addChild(new Menu(5, "返回主菜单", Command.GO_BACK_MAIN));

        Menu menu4 = new Menu(4, "订单管理", Command.SHOW_CHILDREN);
        menu4.addChild(new Menu(1, "查看订单", Command.VIEW_ORDERS));
        menu4.addChild(new Menu(2, "返回主菜单", Command.GO_BACK_MAIN));

        Menu menu5 = new Menu(5, "返回登录", Command.REGISTER);

        SELLER_MENUS = new Menu[]{menu1, menu2, menu3, menu4, menu5};
    }
}
```

菜单管理器中的菜单主要用于展示，因此，需要提供一个展示的方法。

```
/**
```

```
     * 展示给定的菜单数组
     * @param menus 菜单数组
     */
    public static void showMenu(Menu[] menus){
        System.out.println("============================");
        Arrays.stream(menus).forEach(System.out::println);
        System.out.println("============================");
    }

    /**
     * 展示给定的菜单数组
     * @param menus
     */
    public static void showMenus(Menu[] menus){
        System.out.println("============================");
        Stream.of(menus).forEach(System.out::println);
        System.out.println("============================");
    }
```

编写启动类，测试菜单展示

```
package com.cyx.cinema.platform.client.starter;

import com.cyx.cinema.platform.client.menu.MenuManager;

/**
 * 影院平台客户端
 */
public class CinemaPlatformClient {

    public static void main(String[] args) {
        MenuManager.showMenus(MenuManager.MANAGER_MENUS);
    }
}
```

展示时发现展示信息带有内存地址，原因是没有重写`Object`类中的`toString()`方法

```
package com.cinema.platform.client.menu;

import java.util.ArrayList;
import java.util.List;

/**
 * 菜单
 */
public class Menu {
    /**
     * 序号
     */
    private int order;
    /**
     * 名称
     */
    private String name;
    /**
```

```
     * 菜单命令
     */
    private int command;
    /**
     * 子菜单列表
     */
    private List<Menu> children = new ArrayList<>();

    public Menu(int order, String name, int command) {
        this.order = order;
        this.name = name;
        this.command = command;
    }

    /**
     * 添加子菜单
     * @param child
     */
    public void addChild(Menu child){
        children.add(child);
    }

    @Override
    public String toString() {
        return order + "." + name;
    }
}
```

3. 输入分析

菜单展示后，用户需要选择菜单进行操作，因此需要提供输入交互，交互时应保证输入的正确性，而这种交互操作在退出系统之前会反复出现。因此，输入应该编写在一个工具类中，在减少重复代码的同时还能对异常情况进行统一处理。

```
package com.cinema.platform.client.util;

import com.cinema.platform.client.menu.MenuManager;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Scanner;

/**
 * 输入工具类
 */
public class InputUtil {

    private static final Scanner SCANNER = new Scanner(System.in);

    /**
     * 从控制台获取一个给定范围区间内的整数
     * @param tip
     * @param min
     * @param max
     * @return
     */
    public static int getInputInteger(String tip, int min, int max){
```

```
        System.out.println(tip);
        while (true){
            if(SCANNER.hasNextInt()){
                int number = SCANNER.nextInt();
                if(number >= min && number <= max){
                    return number;
                } else {
                    System.out.println("输入错误，请输入" + min + "~" + max + "之间的整数");
                }
            } else {
                System.out.println("输入错误，请输入" + min + "~" + max + "之间的整数");
                SCANNER.next();
            }
        }
    }

    /**
     * 从控制台获取一个字符串
     * @param tip
     * @return
     */
    public static String getInputText(String tip){
        System.out.println(tip);
        return SCANNER.next();
    }

    /**
     * 从控制台获取一个给定日期格式的日期
     * @param tip
     * @param format
     * @return
     */
    public static String getInputDate(String tip, String format){
        System.out.println(tip);
        while (true){
            String dateStr = SCANNER.next();
            SimpleDateFormat sdf = new SimpleDateFormat(format);
            boolean valid = true;
            try {
                sdf.parse(dateStr);
            } catch (ParseException e) {
                valid = false;
            }
            if(valid){
                return dateStr;
            } else {
                System.out.println("日期格式错误，请重新输入：（" + format + "）");
            }
        }
    }
}
```

4. 界面展示分析

以登录界面为例，用户在注册成功、登录失败、找回密码、申请解冻等操作后可以继续登录，换言之，登录界面需要反复的展示，可以使用循环来实现。但为了加深对于知识点的应用，这里采用递归的方式实现。主界面设计也是一样。而递归只能使用方法来实现，因此，界面展示需要编写成一个方法。

```
package com.cinema.platform.client.starter;

import com.cinema.platform.client.menu.Menu;
import com.cinema.platform.client.menu.MenuManager;
import com.cinema.platform.client.util.InputUtil;
import com.cyx.cinema.platform.command.Command;


/**
 * 影院平台客户端
 */
public class CinemaPlatformClient {

    public static void main(String[] args) {
        showLoginMenu();
    }


    private static void showLoginMenu(){
        MenuManager.showMenu(MenuManager.LOGIN_MENUS);
        int number = InputUtil.getInputInteger("请选择菜单编号：", 1,
MenuManager.LOGIN_MENUS.length);
        Menu select = MenuManager.LOGIN_MENUS[number-1];
        switch (select.getCommand()){
            case Command.REGISTER:
                System.out.println("你选择了注册");
                showLoginMenu();
                break;
            case Command.LOGIN:
                System.out.println("你选择了登录");
                showMainMenu();
                break;
        }
    }

    private static void showMainMenu(){
        MenuManager.showMenu(MenuManager.MANAGER_MENUS);
        int number = InputUtil.getInputInteger("请选择菜单编号：", 1,
MenuManager.MANAGER_MENUS.length);
        Menu select = MenuManager.MANAGER_MENUS[number-1];
        switch (select.getCommand()){
            case Command.SHOW_CHILDREN:
                System.out.println("你选择了显示子菜单");
                showChildMenu(select.getChildren());
                break;
            case Command.GO_BACK_LOGIN:
                System.out.println("你选择了返回登录");
                showLoginMenu();
                break;
        }
    }
```

```
50
51      private static void showChildMenu(Menu[] menus){
52          MenuManager.showMenu(menus);
53          int number = InputUtil.getInputInteger("请选择菜单编号：", 1,
menus.length);
54          Menu select = menus[number-1];
55          switch (select.getCommand()){
56              case Command.GO_BACK_MAIN:
57                  System.out.println("你选择了返回主菜单");
58                  showMainMenu();
59                  break;
60              default:
61                  System.out.println(select);
62          }
63      }
64
65  }
```

代码优化

```
1   package com.cinema.platform.client.starter;
2
3   import com.cinema.platform.client.menu.Menu;
4   import com.cinema.platform.client.menu.MenuManager;
5   import com.cinema.platform.client.util.InputUtil;
6   import com.cyx.cinema.platform.command.Command;
7
8
9   /**
10   * 影院平台客户端
11   */
12  public class CinemaPlatformClient {
13
14      public static void main(String[] args) {
15  //          showLoginMenu();
16          showInterface(MenuManager.LOGIN_MENUS);
17      }
18
19      /**
20       * 展示给定的菜单
21       * @param menus
22       */
23      private static void showInterface(Menu[] menus){
24          MenuManager.showMenu(menus);
25          int number = InputUtil.getInputInteger("请选择菜单编号：", 1,
menus.length);
26          Menu select = menus[number-1];
27          switch (select.getCommand()){
28              case Command.REGISTER:
29                  System.out.println("你选择了注册");
30                  showInterface(menus);
31                  break;
32              case Command.LOGIN:
33                  System.out.println("你选择了登录");
34                  showInterface(MenuManager.MANAGER_MENUS);
35                  break;
36              case Command.GO_BACK_MAIN:
```

```
                System.out.println("你选择了返回主菜单");
                showInterface(MenuManager.MANAGER_MENUS);
                break;
            case Command.SHOW_CHILDREN:
                System.out.println("你选择了显示子菜单");
                showInterface(select.getChildren());
                break;
            case Command.GO_BACK_LOGIN:
                System.out.println("你选择了返回登录");
                showInterface(MenuManager.LOGIN_MENUS);
                break;
            case Command.QUIT:
                System.out.println("感谢使用影院选票平台");
                System.exit(0);
                break;
            default:
                System.out.println(select);
                showInterface(menus);
        }
    }


//    private static void showLoginMenu(){
//        MenuManager.showMenu(MenuManager.LOGIN_MENUS);
//        int number = InputUtil.getInputInteger("请选择菜单编号：", 1,
MenuManager.LOGIN_MENUS.length);
//        Menu select = MenuManager.LOGIN_MENUS[number-1];
//        switch (select.getCommand()){
//            case Command.REGISTER:
//                System.out.println("你选择了注册");
//                showLoginMenu();
//                break;
//            case Command.LOGIN:
//                System.out.println("你选择了登录");
//                showMainMenu();
//                break;
//        }
//    }
//
//    private static void showMainMenu(){
//        MenuManager.showMenu(MenuManager.MANAGER_MENUS);
//        int number = InputUtil.getInputInteger("请选择菜单编号：", 1,
MenuManager.MANAGER_MENUS.length);
//        Menu select = MenuManager.MANAGER_MENUS[number-1];
//        switch (select.getCommand()){
//            case Command.SHOW_CHILDREN:
//                System.out.println("你选择了显示子菜单");
//                showChildMenu(select.getChildren());
//                break;
//            case Command.GO_BACK_LOGIN:
//                System.out.println("你选择了返回登录");
//                showLoginMenu();
//                break;
//        }
//    }
//
//    private static void showChildMenu(Menu[] menus){
//        MenuManager.showMenu(menus);
```

```
//        int number = InputUtil.getInputInteger("请选择菜单编号：", 1,
menus.length);
//        Menu select = menus[number-1];
//        switch (select.getCommand()){
//            case Command.GO_BACK_MAIN:
//                System.out.println("你选择了返回主菜单");
//                showMainMenu();
//                break;
//            default:
//                System.out.println(select);
//        }
//    }
}
```

三、 `cinema-platform-server`设计

1. 服务器套接字

```
package com.cyx.cinema.platform.server.starter;

import com.cyx.cinema.platform.command.Command;
import com.cyx.cinema.platform.message.Message;
import com.cyx.cinema.platform.util.SocketUtil;

import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

/**
 * 影院平台服务器
 */
public class CinemaPlatformServer {

    private ServerSocket serverSocket;

    public CinemaPlatformServer(int port) throws IOException {
        this.serverSocket = new ServerSocket(port);
    }

    /**
     * 服务器启动
     */
    public void start(){
        while (true){
            try {
                //等待客户端连接
                Socket socket = serverSocket.accept();
                Message msg = SocketUtil.receiveMsg(socket);
                switch (msg.getCommand()){
                    case Command.REGISTER:
                        System.out.println("接到客户端注册请求");
                        break;
                }
                SocketUtil.sendMsg(socket, "");
            } catch (IOException e) {
```

```
                e.printStackTrace();
            }
        }
    }

    public static void main(String[] args) {
        try {
            CinemaPlatformServer server= new CinemaPlatformServer(8888);
            server.start();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

2. 并发处理

服务器可以同时处理多个用户的操作请求，比如用户A在注册的同时用户B在登录。然而根据Java代码按顺序逐行执行的特征，用户B必须等待用户A注册完成后才能登录，这明显不符合用户的实际需求。为了解决这个问题，这里需要使用线程来完成

```
package com.cyx.cinema.platform.server.task;

import com.cyx.cinema.platform.command.Command;
import com.cyx.cinema.platform.message.Message;
import com.cyx.cinema.platform.util.SocketUtil;

import java.net.Socket;

/**
 * 消息处理任务
 */
public class MessageProcessTask implements Runnable{

    private Socket socket;

    public MessageProcessTask(Socket socket) {
        this.socket = socket;
    }

    @Override
    public void run() {
        Message msg = SocketUtil.receiveMsg(socket);
        switch (msg.getCommand()){
            case Command.REGISTER:
                System.out.println("接到客户端注册请求");
                break;
        }
        SocketUtil.sendMsg(socket, "");
    }
}

package com.cyx.cinema.platform.server.starter;

import com.cyx.cinema.platform.server.task.MessageProcessTask;

import java.io.IOException;
```

```java
import java.net.ServerSocket;
import java.net.Socket;

/**
 * 影院平台服务器
 */
public class CinemaPlatformServer {

    private ServerSocket serverSocket;

    public CinemaPlatformServer(int port) throws IOException {
        this.serverSocket = new ServerSocket(port);
    }

    /**
     * 服务器启动
     */
    public void start(){
        while (true){
            try {//用户A注册，此时用户B登录
                //等待客户端连接
                Socket socket = serverSocket.accept();
                //启动线程执行消息处理任务
                new Thread(new MessageProcessTask(socket)).start();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    public static void main(String[] args) {
        try {
            CinemaPlatformServer server= new CinemaPlatformServer(8888);
            server.start();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

3. 信息读取与存档

对于用户的注册信息、解冻申请、订单、影片、影片计划、影厅信息等都需要存档，用户查询时再读取出来。针对这些频繁发生的操作，可以使用工具类来完成。

```java
package com.cyx.cinema.platform.server.util;

import com.cyx.cinema.platform.entity.Film;

import java.io.*;
import java.util.ArrayList;
import java.util.List;

/**
 * 文件操作工具类
 */
```

```java
public class FileUtil {
    /**
     * 用户存档文件
     */
    public static final String USER_FILE = "data/user.obj";
    /**
     * 影片存档文件
     */
    public static final String FILM_FILE = "data/film.obj";
    /**
     * 影厅存档文件
     */
    public static final String FILM_HALL_FILE = "data/filmHall.obj";
    /**
     * 影片计划存档文件
     */
    public static final String FILM_PLAN_FILE = "data/filmPlan.obj";
    /**
     * 订单存档文件
     */
    public static final String ORDER_FILE = "data/order.obj";
    /**
     * 解冻申请存档文件
     */
    public static final String UNFROZEN_APPLY_FILE =
"data/unfrozenApply.obj";

    /**
     * 保存给定的列表数据至给定的路径文件中
     * @param dataList
     * @param path
     * @param <T>
     * @return
     */
    public static <T> boolean saveData(List<T> dataList, String path){
        File file = new File(path);
        File parent = file.getParentFile();
        if(parent != null && !parent.exists()){
            parent.mkdirs();
        }
        try (OutputStream os =new FileOutputStream(file);
             ObjectOutputStream oos = new ObjectOutputStream(os)){
            oos.writeObject(dataList);
            oos.flush();
            return true;
        } catch (Exception e) {
            return false;
        }
    }

    /**
     * 读取给定路径文件中存储的数据
     * @param path
     * @param <T>
     * @return
     */
    public static <T> List<T> readData(String path){
        try (InputStream is = new FileInputStream(path);
```

```
69              ObjectInputStream ois = new ObjectInputStream(is)){
70                  return (List<T>) ois.readObject();
71          } catch (Exception e) {
72                  return new ArrayList<>();
73          }
74      }
75  }
```

四、功能实现

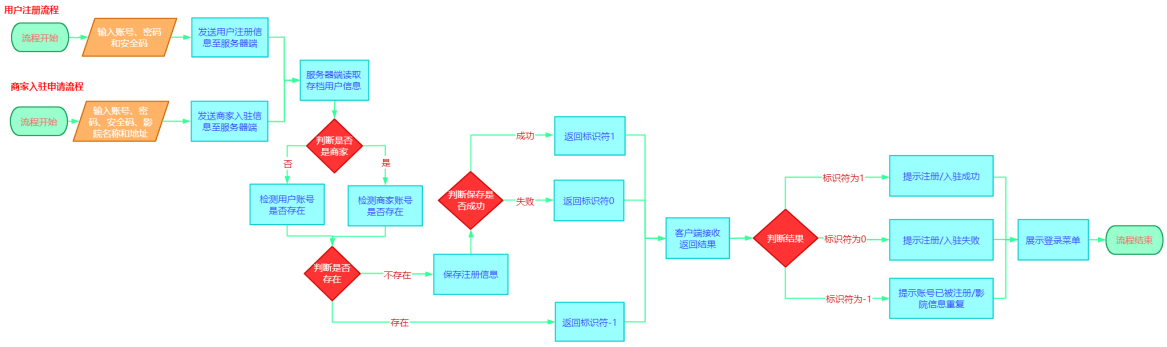
1. 登录界面

注册

注册需要输入账号、密码、安全码。服务器端处理注册请求，处理结果有：注册成功、注册失败、账号已被注册。可以使用标识符来标识：1-成功，0-失败，-1 - 账号已被注册

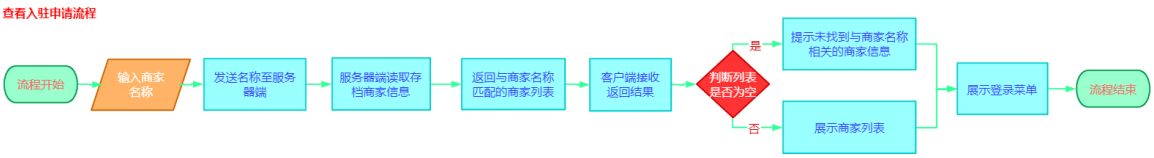
入驻申请

影院入驻申请需要输入账号、密码、安全码、影院名称和影院地址。服务器端处理入驻申请请求，处理结果有：申请成功、申请失败、账号/影院重复申请。可以使用标识符来标识：1-成功，0-失败，-1 - 账号/影院重复申请



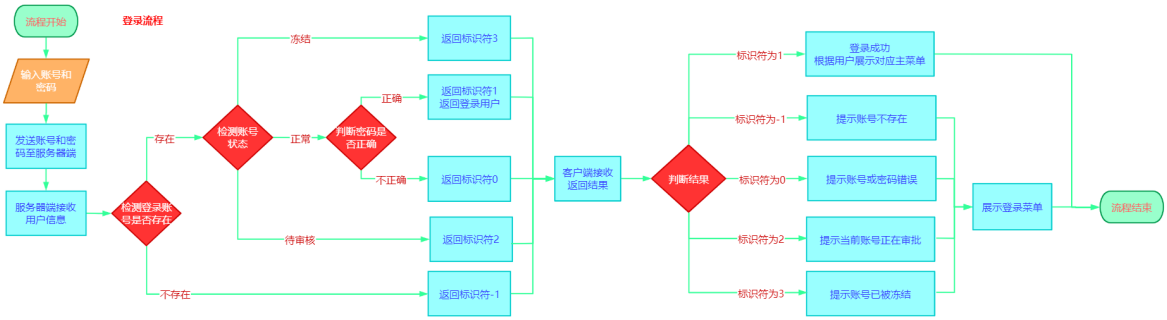
查看入驻申请

查看影院入驻申请需要输入影院名称。



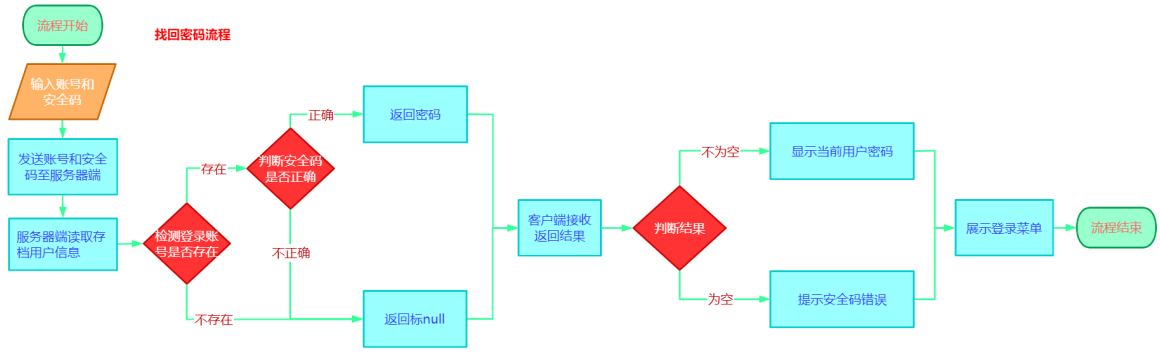
登录

登录需要输入账号、密码。服务器端处理登录请求，处理结果有：登录成功、登录失败、账号不存在、账号被冻结、账号正在审批。可以使用标识符来标识：1-成功，0-失败，-1 - 账号不存在，2-账号正在审批， 3 - 账号被冻结。服务器端还需要将用户的身份反馈回来，方便客户端展示主菜单



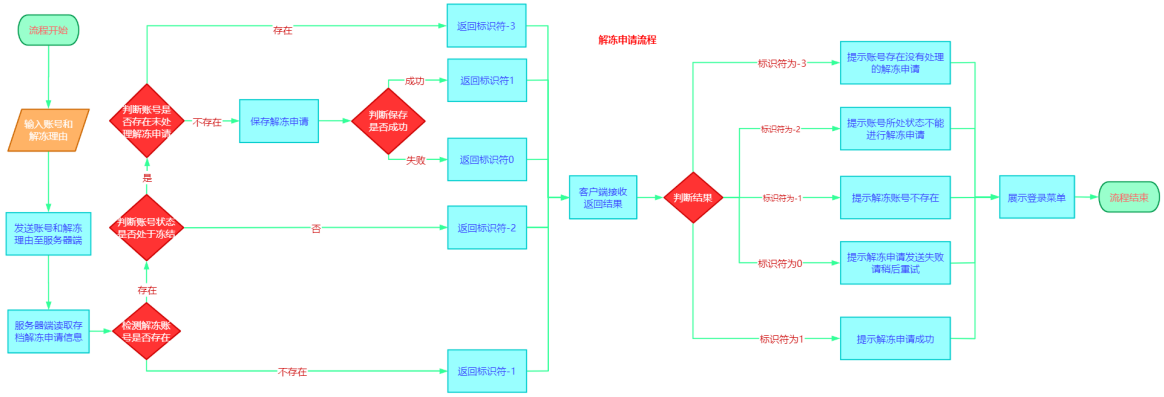
找回密码

找回密码需要输入账号、安全码。服务器端处理找回密码请求，处理结果有：安全码正确，返回密码；安全码错误。可以使用标识符来标识：1-安全码正确，0-安全码错误



申请解冻

申请解冻需要输入账号、解冻原因。服务器端处理申请解冻请求，处理结果有：成功、失败、解冻账号存在未处理的解冻申请、账号所处状态不能进行解冻申请、解冻账号不存在。可以使用标识符来标识：1-成功，0-失败，-1 -解冻账号不存在，-2 -账号所处状态不能进行解冻申请，-3 - 解冻账号存在未处理的解冻申请

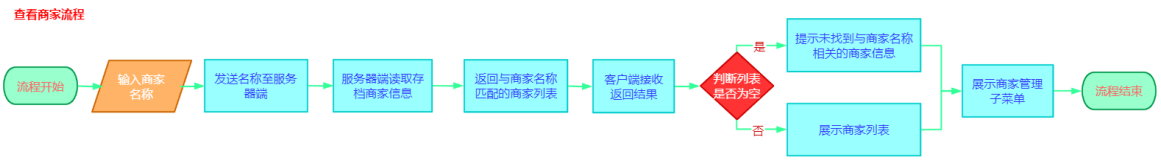


退出系统

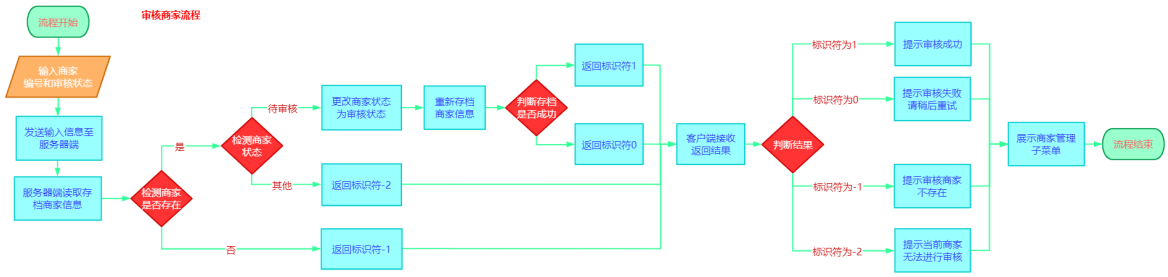
2. 管理员主界面

商家管理

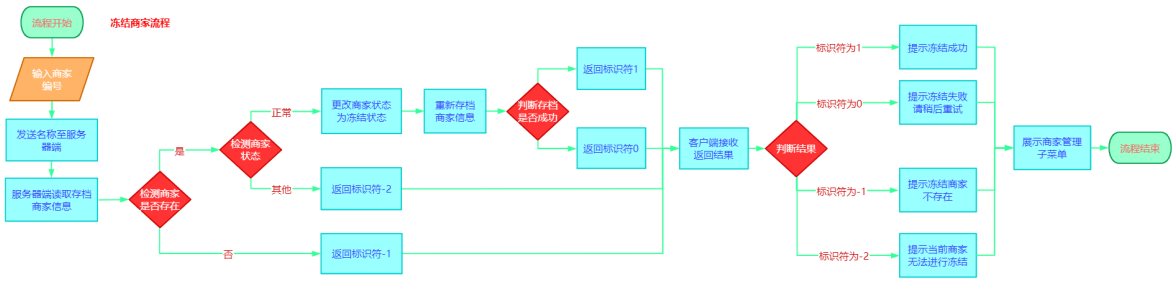
查看商家：需要输入商家名称（也就是影院名称）



审核商家：需要输入商家账号和审核状态



冻结商家：需要输入商家编号



用户管理（学员完成）

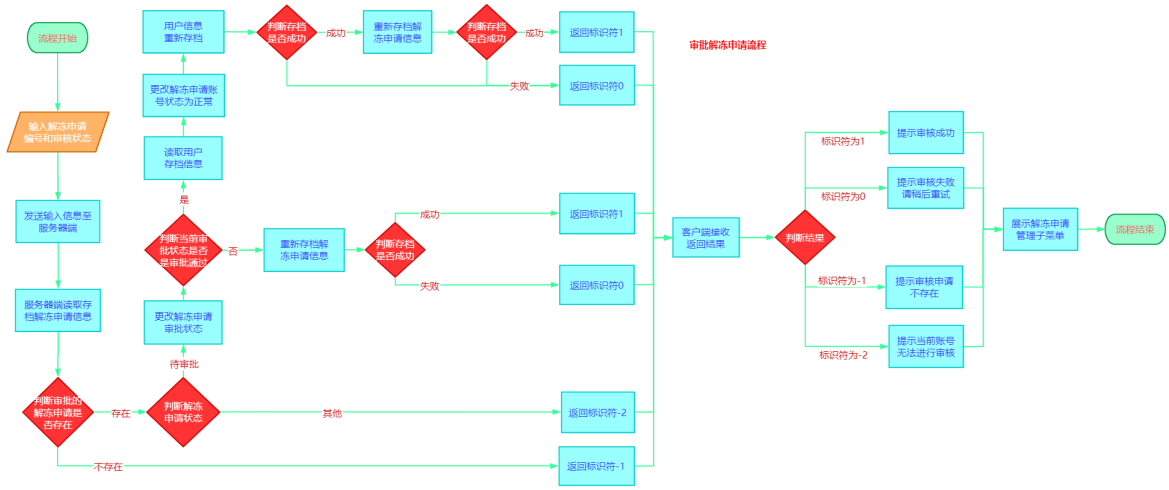
查看用户信息：查看所有用户

冻结用户：需要输入用户名

解冻申请管理

查看解冻申请：查看所有解冻申请（学员完成）

审批解冻申请：需要输入解冻申请编号，处理状态



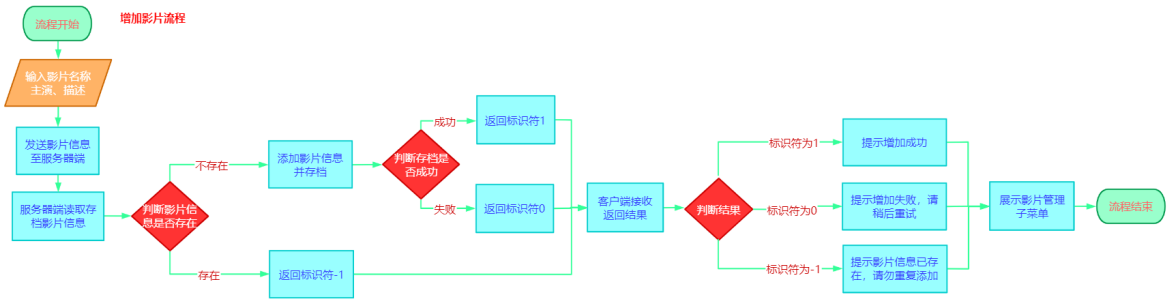
返回登录

3. 商家主界面

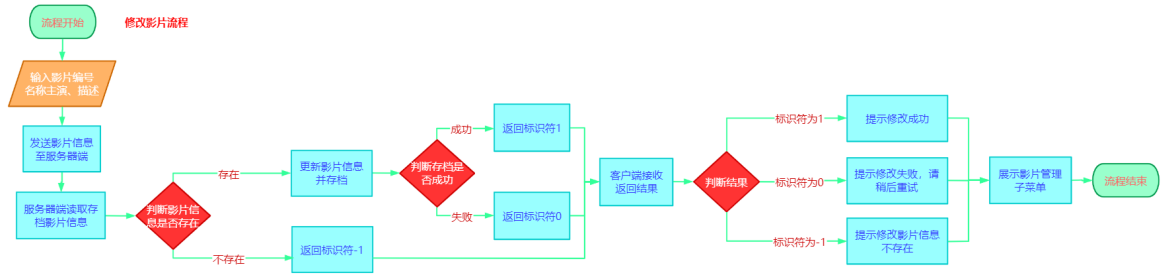
影片管理

包含影片的增删改查

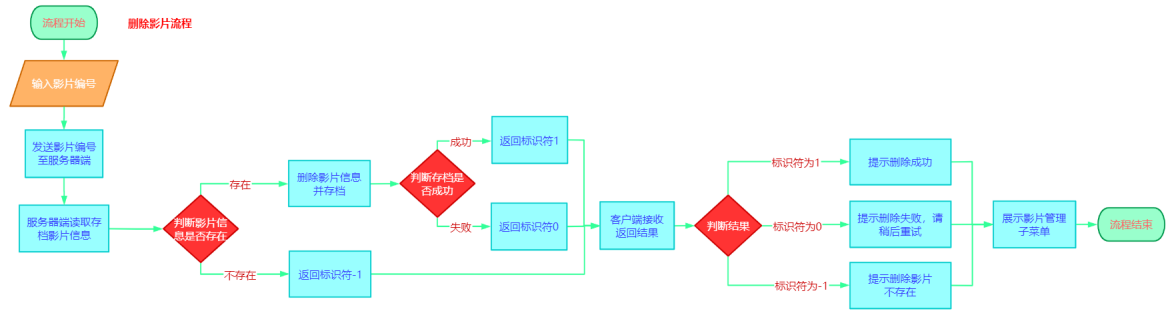
增加影片：需要输入影片名称、主演、影片描述



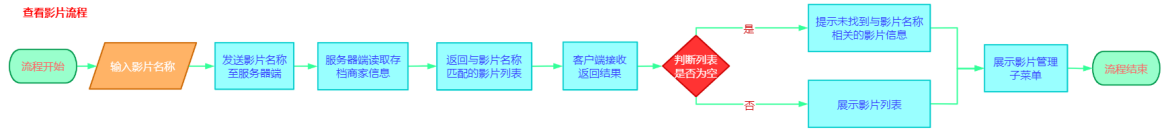
修改影片：需要输入影片的编号、影片名称、主演、影片描述



删除影片：需要输入影片的编号



查看影片：输入影片名称，可以进行模糊匹配



影厅管理（学员完成）

包含影厅的增删改查

增加影厅：需要输入影厅名称、总排数、总列数

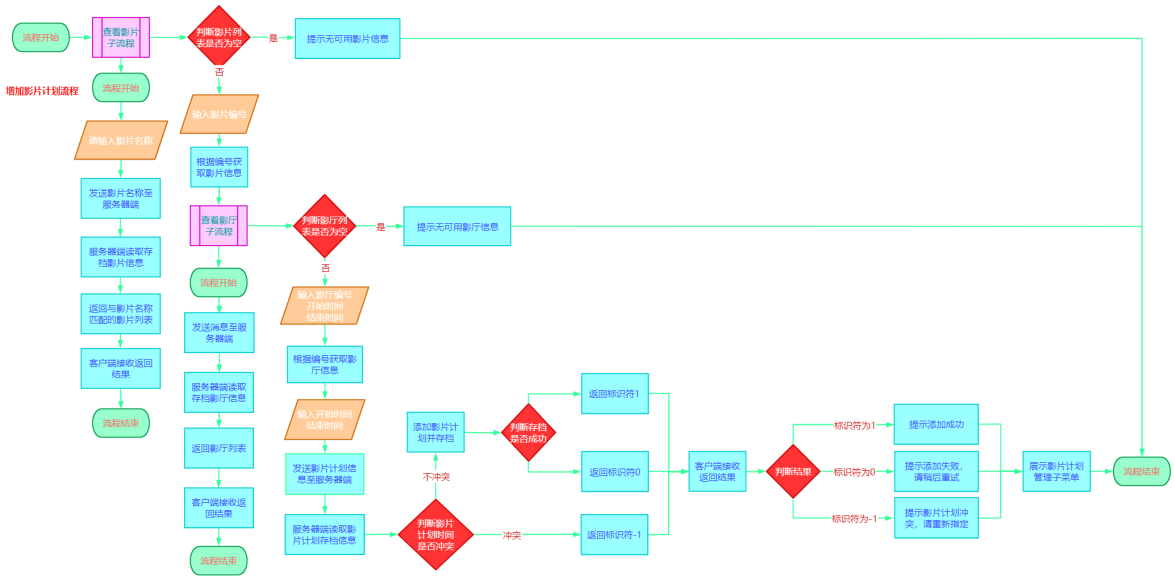
修改影厅：需要输入的编号、影片名称、总排数、总列数

删除影厅：需要输入影厅的编号

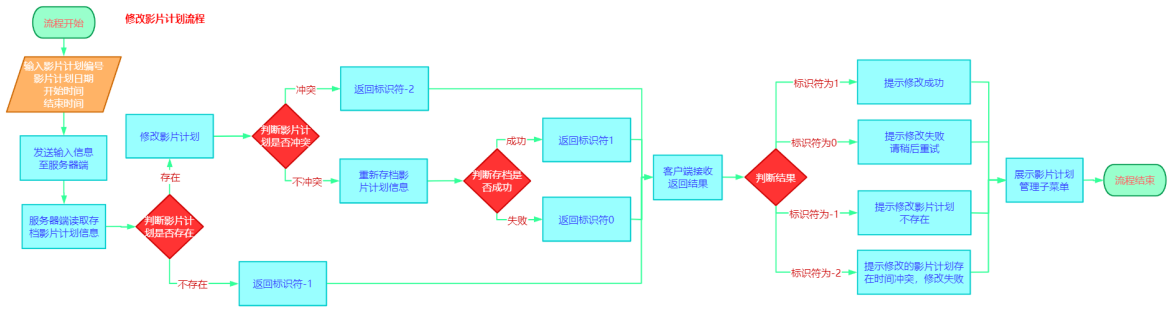
查看影厅：查看所有影厅信息

影片计划管理

增加影片计划：需要输入影片编号、影厅编号、播放日期、开始时间、结束时间



修改影片计划：需要输入影片计划的编号、播放日期、开始时间、结束时间



删除影片计划：需要输入影片计划的编号（学员完成）

查看影片计划：输入影片名称，可以进行模糊匹配（学员完成）

订单管理（学员完成）

订单查看：只能查看商家自己的订单

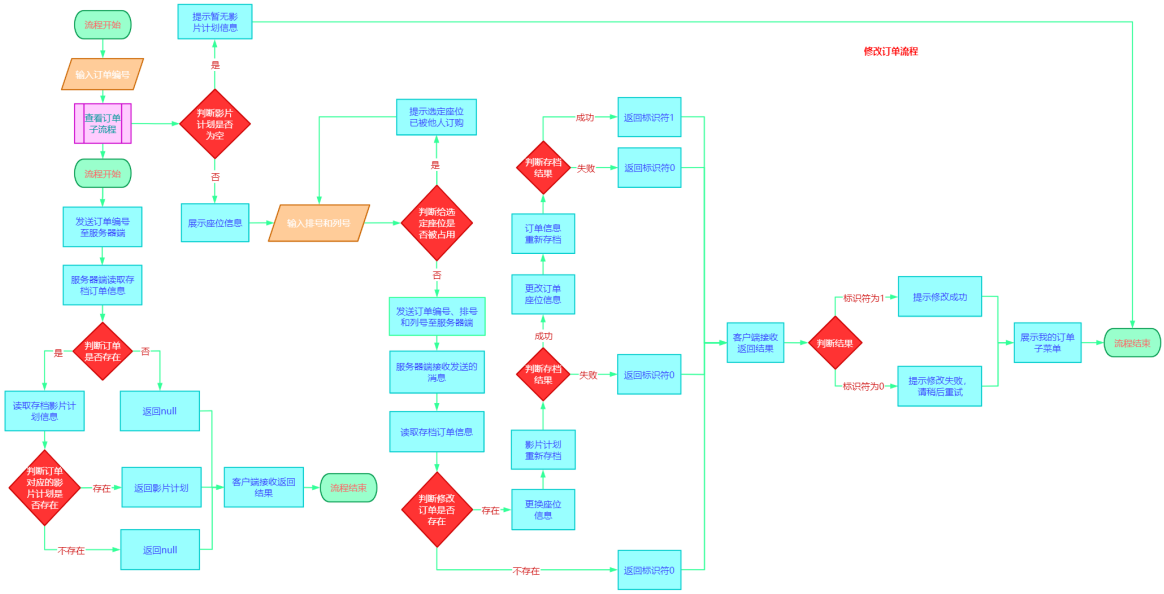
4. 普通用户主界面

查看订单（学员完成）

只能查看用户自己的订单

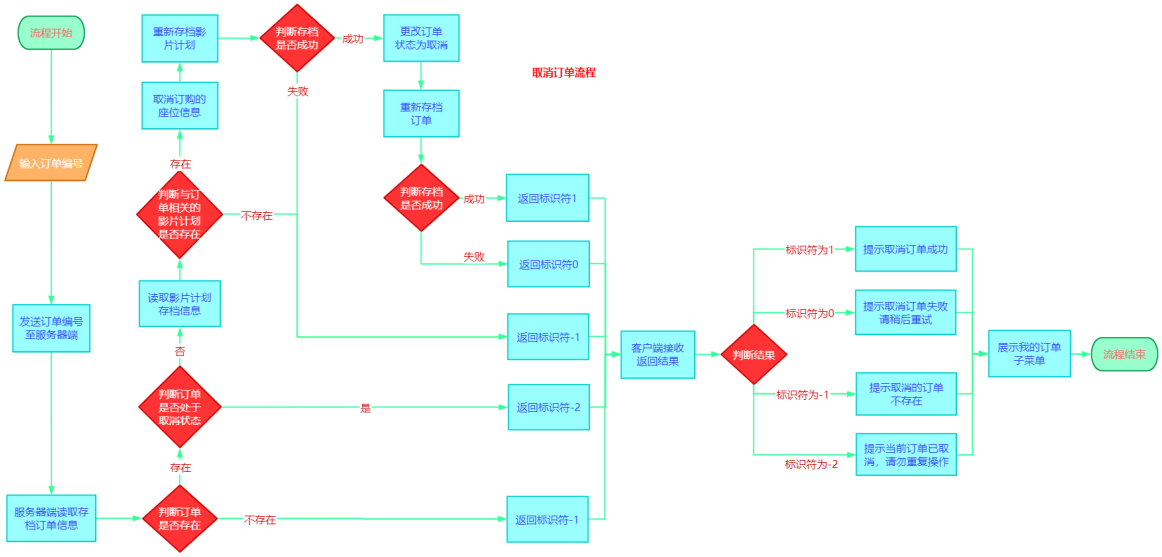
修改订单

只能修改用户自己的订单



取消订单

只能取消用户自己的订单



在线订座

首先需要查看影票信息（也就是影片计划），然后选择影片计划，展示座位信息，再选择座位。