

# 智能电话本项目

## 一、 项目介绍

### 1 项目描述

该项目是用于日常生活中记录联系人信息的一款智能小工具。

实现了对联系人的姓名、年龄、性别、电话号码、住址的添加及修改、查找、删除、排序等功能。该项目是以 Windows 控制台为运行平台，所有的操作都应在控制台上进行。

### 2 编写目的

通过该项目可以对以往所学过的知识点进行统一的复习，将平时所学的内容综合到一起并运用到实际的项目中。

该项目能够充分的体现出一些简单的业务流程处理，能够使同学们掌握基本的业务流程处理。

进一步理解什么是面相对象设计(OOD : Object Oriented Design)。充分理解面向对象设计的精髓。

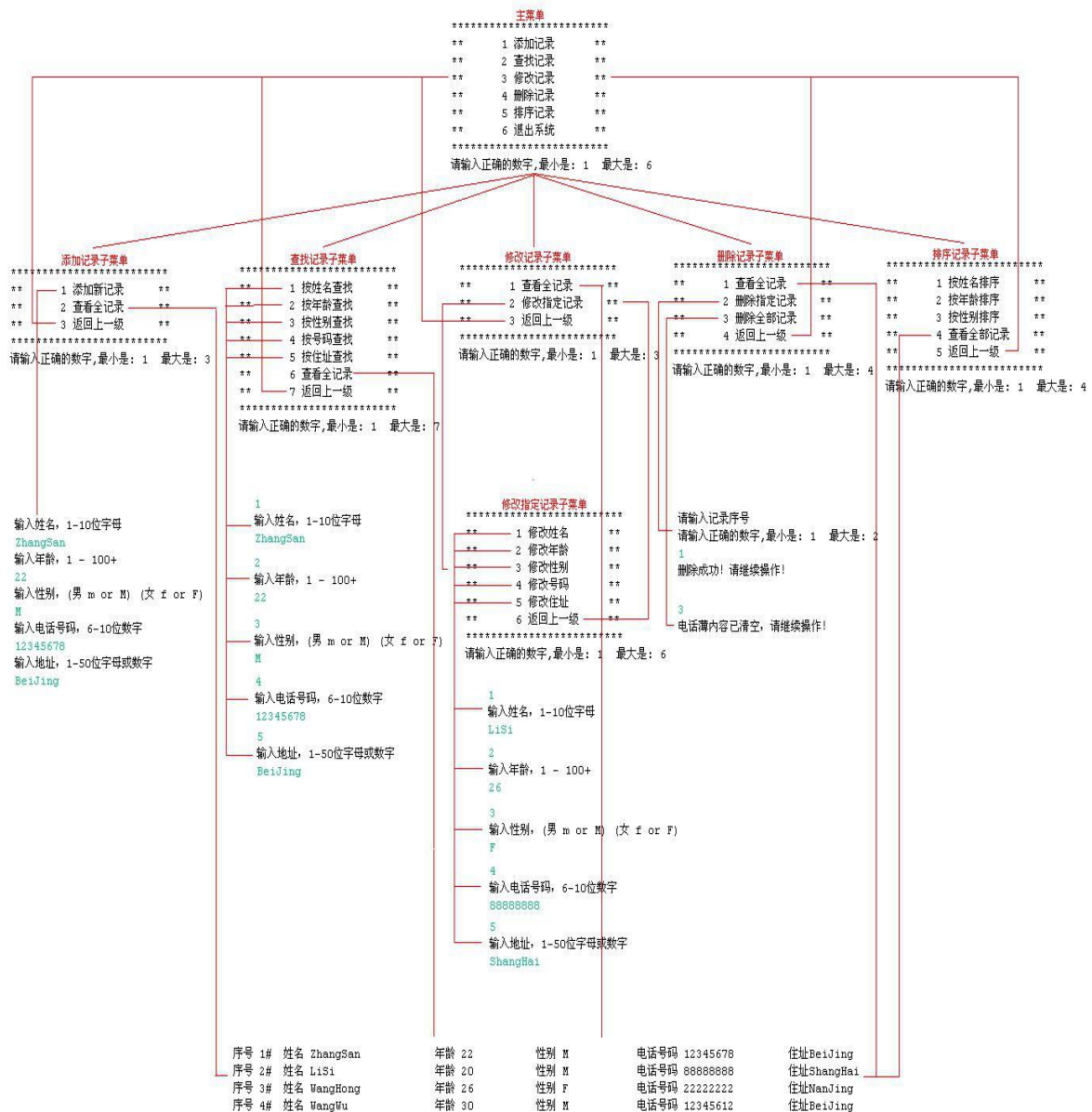
## 二、 项目演示

双击 start.bat 可启动电话本项目



### 三、 技术说明

#### 1 业务流程结构图



## 2 数据存储结构

id	序号
name	姓名
age	年龄
sex	性别
telNum	电话号码
address	地址

## 3 技术要求

- JDK 版本为 1.8 或以上
- 键盘输入 Scanner 类
- 正则表达式 Pattern 类
- 容器 ArrayList 类
- 排序方式（比较器排序）Comparator 接口，Collections 类的 sort 方法

## 4 开发环境

Windows 操作系统

## 5 开发工具

Idea

## 四、 编码规范

### 1 类名

App 对应 Application（程序入口类）

Menu 对应 Menu ( 菜单类 )

Operate 对应 Operate ( 业务处理类 )

Person 对应 Person ( 实体类 )

TelNoteRegex 对应 TelNoteRegex ( 用户输入验证类 )

OrderByName 对应 OrderByName (姓名排序比较器)

OrderByAge 对应 OrderByAge ( 年龄排序比较器 )

OrderBySex 对应 OrderBySex ( 性别排序比较器 )

## 2 方法名

### 2.1 App 类中方法

main() 启动程序方法

start() 主菜单控制

### 2.2 Operate 类中方法

private List<Person> list 容器

addLogic() 用户添加记录业务逻辑控制

searchLogic() 用户查询记录业务逻辑控制

modifyLogic() 修改记录业务逻辑控制

deleteLogic() 删除记录业务逻辑控制

orderLogic() 排序记录业务逻辑控制

addOperation () 添加新记录信息

showAll() 查询全部记录

searchByName() 按姓名查询记录

searchByAge() 按年龄查询记录

searchBySex() 按性别查询记录

searchByTelNum() 按电话号码查询记录

searchByAdd() 按地址查询记录

modifyOperation() 修改指定记录

deleteOperation() 删除指定记录

deleteAllOperation() 删除全部记录

orderName() 按用户姓名排序记录

orderAge() 按用户年龄排序记录

orderSex() 按用户性别排序记录

## 2.3 TelNoteRegex 类中方法

menuItemValidate (int min, int max ) 对菜单输入选项的验证

nameValidate ( ) 对用户输入姓名的验证

ageValidate ( ) 对用户输入年龄的验证

sexValidate ( ) 对用户输入性别的验证

telNumValidate ( ) 对用户输入电话号码的验证

addressValidate ( ) 对用户输入地址的验证

## 2.4 Menu 类中的方法

mainMenu() 主菜单

addMenu () 添加记录菜单

searchMenu () 查找记录菜单

modifyMenu () 修改记录主菜单

subModifyMenu () 修改记录子菜单

deleteMenu () 删除记录菜单

orderMenu () 排序记录菜单

## 2.5 Person 类中的方法

private int id; 用户序号属性

private String name; 用户姓名属性

private String age; 用户年龄属性

private String sex; 用户性别属性

private String telNum; 用户电话号码属性

private String address; 用户地址属性

Person() 无参数构造方法

Person(String name, String age, String sex, String telNum, String address) 有参数构造方法

getName() 读取用户名

setName(String name) 设置用户名

getAge() 读取用户年龄

setAge(String age) 设置用户年龄

getSex() 读取用户性别

setSex(String sex) 设置用户性别

getTelNum() 读取用户电话号码

setTelNum (String telNum) 设置用户电话号码

getAddress() 读取用户地址

setAddress(String address) 设置用户地址

getID () 读取用户 ID 号

setID (int ID) 设置用户 ID 号

toString() 连接字符串方法

## 2.6 OrderByName 类中的方法名

compare(Person p1, Person p2)根据姓名排序方法

## 2.7 OrderByAge 类中的方法名

compare(Person p1, Person p2)根据年龄排序方法

## 2.8 OrderBySex 类中的方法名

compare(Person p1, Person p2)根据性别排序方法

## 五、 项目开发

### 1 创建项目



### 2 创建类

#### 2.1 创建程序入口类

```
/**
 * 电话本项目入口类
 */
public class App {
    /**
     * 启动电话本项目
     * @param args
     */
    public static void main(String[] args) {

    }

    /**
     * 控制主菜单
     */
    public void start(){
    }
}
```



## 2.2 创建菜单类

```
public class Menu {

    //主界面

    public void mainMenu() {
        System.out.println("*****");

        System.out.println("**      1  添加记录      **");

        System.out.println("**      2  查找记录      **");

        System.out.println("**      3  修改记录      **");

        System.out.println("**      4  删除记录      **");

        System.out.println("**      5  排序记录      **");

        System.out.println("**      6  退出系统      **");

        System.out.println("*****");
    }

    //添加界面

    public void addMenu () {
        System.out.println("*****");

        System.out.println("**      1  添加新记录      **");

        System.out.println("**      2  查看全记录      **");

        System.out.println("**      3  返回上一级      **");

        System.out.println("*****");
    }

    //查找界面

    public void searchMenu () {
        System.out.println("*****");

        System.out.println("**      1  按姓名查找      **");
    }
}
```

```

System.out.println("**      2 按年龄查找      **");

System.out.println("**      3 按性别查找      **");

System.out.println("**      4 按号码查找      **");

System.out.println("**      5 按住址查找      **");

System.out.println("**      6 查看全记录      **");

System.out.println("**      7 返回上一级      **");

System.out.println("*****");
}

```

//修改界面

```

public void modifyMenu () {
    System.out.println("*****");

    System.out.println("**      1 查看全记录      **");

    System.out.println("**      2 修改指定记录      **");

    System.out.println("**      3 返回上一级      **");

    System.out.println("*****");
}

```

//修改子界面

```

public void subModifyMenu () {
    System.out.println("*****");

    System.out.println("**      1 修改姓名      **");

    System.out.println("**      2 修改年龄      **");

    System.out.println("**      3 修改性别      **");

    System.out.println("**      4 修改号码      **");

    System.out.println("**      5 修改住址      **");

    System.out.println("**      6 返回上一级      **");
}

```

```

        System.out.println("*****");
    }

    //删除界面

    public void deleteMenu () {
        System.out.println("*****");

        System.out.println("**      1 查看全记录      **");

        System.out.println("**      2 删除指定记录    **");

        System.out.println("**      3 删除全部记录    **");

        System.out.println("**      4 返回上一级      **");

        System.out.println("*****");
    }

    //排序界面

    public void orderMenu () {
        System.out.println("*****");

        System.out.println("**      1 按姓名排序      **");

        System.out.println("**      2 按年龄排序      **");

        System.out.println("**      3 按性别排序      **");

        System.out.println("**      4 查看全部记录    **");

        System.out.println("**      5 返回上一级      **");

        System.out.println("*****");
    }
}

```

## 2.3 创建实体类

```

/**
 * 实体类
 */
public class Person {

```

```
private int id;
private String name;
private String age;
private String sex;
private String telNum;
private String address;

public Person() {
}

public Person(String name, String age, String sex, String telNum,
String address) {
    this.name = name;
    this.age = age;
    this.sex = sex;
    this.telNum = telNum;
    this.address = address;
}

public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getAge() {
    return age;
}

public void setAge(String age) {
    this.age = age;
}
```

```

public String getSex() {
    return sex;
}

public void setSex(String sex) {
    this.sex = sex;
}

public String getTelNum() {
    return telNum;
}

public void setTelNum(String telNum) {
    this.telNum = telNum;
}

public String getAddress() {
    return address;
}

public void setAddress(String address) {
    this.address = address;
}

@Override
public String toString() {
    StringBuffer sb = new StringBuffer();

    sb.append("序号").append(this.id).append("#").append("\t");

    sb.append("姓名").append(this.name).append("\t\t");

    sb.append("年龄").append(this.age).append("\t\t");

    sb.append("性别").append(this.sex).append("\t\t");

    sb.append("电话号码").append(this.telNum).append("\t\t");

    sb.append("地址").append(this.address);

    return sb.toString();
}
}

```

## 2.4 创建核心业务类

```
/**
 * 核心业务类
 */
public class Operate {
    private List<Person> list ;
    public Operate(){
        this.list = new ArrayList<>();
    }
    /**
     * 用户添加记录业务逻辑控制
     */
    public void addLogic(){

    }

    /**
     * 用户查询记录业务逻辑控制
     */
    public void searchLogic(){

    }

    /**
     * 修改记录业务逻辑控制
     */
    public void modifyLogic(){

    }

    /**
     * 删除记录业务逻辑控制
     */
    public void deleteLogic(){

    }
}
```

```

/**
 * 排序记录业务逻辑控制
 */
public void orderLogic() {

}

/**
 * 添加新记录信息
 */
public void addOperation() {

}

/**
 * 查询全部记录
 */
public void showAll() {

}

/**
 * 按姓名查询记录
 */
public void searchByName() {

}

/**
 * 按年龄查询记录
 */
public void searchByAge() {

}

/**
 * 按性别查询记录
 */
public void searchBySex() {

```

```

    }
    /**
     * 按电话号码查询记录
     */
    public void searchByTelNum() {

    }
    /**
     * 按地址查询记录
     */
    public void searchByAdd() {

    }
    /**
     * 修改指定记录
     */
    public void modifyOperation() {

    }
    /**
     * 删除指定记录
     */
    public void deleteOperation() {

    }
    /**
     * 删除全部记录
     */
    public void deleteAllOperation() {

    }
    /**
     * 按用户姓名排序记录
     */
    public void orderName() {

```



```

    }

    /**
     * 按用户年龄排序记录
     */
    public void orderAge() {

    }

    /**
     * 按用户性别排序记录
     */
    public void orderSex() {

    }
}

```

## 2.5 创建数据校验类

```

/**
 * 数据校验类
 */
public class TelNoteRegex {
    /**
     * 对菜单输入选项的验证
     * @param min
     * @param max
     * @return
     */
    public int menuItemValidate (int min, int max ){
        return 0;
    }

    /**
     * 对用户输入姓名的验证
     * @return
     */
}

```

```

public String nameValidate ( ){
    return null;
}

/**
 * 对用户输入年龄的验证
 * @return
 */
public String ageValidate ( ){
    return null;
}

/**
 * 对用户输入性别的验证
 * @return
 */
public String sexValidate ( ){
    return null;
}

/**
 * 对用户输入电话号码的验证
 * @return
 */
public String telNumValidate ( ){
    return null;
}

/**
 * 对用户输入地址的验证
 * @return
 */
public String addressValidate ( ){
    return null;
}
}

```

## 2.6 创建比较器类

排序器只能对在电话本中所存储的记录进行排序，不具备通用性，所以定义成 Operate 的内部类

### 2.6.1 按姓名排序的比较器

```
/**
 * 按姓名排序的比较器
 */
class OrderByName implements Comparator<Person>{

    @Override
    public int compare(Person o1, Person o2) {
        return 0;
    }
}
```

### 2.6.2 按年龄排序的比较器

```
/**
 * 按年龄排序的比较器
 */
class OrderByAge implements Comparator<Person>{

    @Override
    public int compare(Person o1, Person o2) {
        return 0;
    }
}
```

### 2.6.3 按性别排序的比较器

```
/**
 * 按性别排序的比较器
 */
```

```
class OrderBySex implements Comparator<Person>{
    @Override
    public int compare(Person o1, Person o2) {
        return 0;
    }
}
```

## 3 正则表达式

### 3.1 正则表达式介绍

#### 3.1.1 什么是正则表达式

正则表达式，又称规则表达式。（英语：Regular Expression，在代码中常简写为 regex、regexp 或 RE），是计算机科学的一个概念。正则表达式通常被用来检索、替换那些符合某个模式(规则)的文本。正则表达式并不仅限于某一种语言，但是在每种语言中有细微的差别。

#### 3.1.2 正则表达式的作用

正则表达式是对字符串操作的一种逻辑公式，就是用事先定义好的一些特定字符、及这些特定字符的组合，组成一个“规则字符串”，这个“规则字符串”用来表达对字符串的一种过滤逻辑。使用特殊语法来表示字符类、数量限定符和位置关系,然后用这些特殊语法和普通字符一起表示一个模式。

#### 3.1.3 正则表达式特点

- 灵活性、逻辑性和功能性非常的强；
- 可以迅速地用极简单的方式达到字符串的复杂控制
- 对于刚接触的人来说，比较晦涩难懂

## 3.2 Java 中正则表达式的使用

在 Java 中正则表达式为 String 类型，被验证的内容同样为 String 类型。通过 String 类中的 matches 方法实现内容的匹配校验。如：“被验证内容”.matches(“正则表达式”)

## 3.3 正则表达式语法规则

正则表达式语法规则：[内容限定]{长度限定}

### 3.3.1 内容限定

在定义限定内容规则时，如果没有指定长度限定，那么默认长度为 1。

#### 3.3.1.1 单个字符限定

[a]:表示当前内容必须是字母 a

#### 3.3.1.2 范围字符限定

[a-z0-9]:表示内容可以是 a-z 之间的任意字母或者 0-9 之间的任意数字，不分先后。

#### 3.3.1.3 取反限定

[^abc]:表示内容不能是 a 或 b 或 c。

### 3.3.2 长度限定

在正则表达式中通过{ }来限定内容长度。

固定长度：{固定长度值}

范围长度：{最小长度值，最大长度值}

[a-z]{5}：表示内容范围为小写字母 a 到 z 且长度必须为 5

[a-z]{2,8} : 表示内容范围为小写字母 a 到 z 且长度在 2 到 8 之间, 包含 2 与 8

[a-z]{2,} : 表示内容范围为小写字母 a 到 z 且最小长度为 2, 最大长度无限制

[a-z]{0,2} : 表示内容范围为小写字母 a 到 z 且最小长度为 0, 最大长度为 2

### 3.3.3 长度限定符号

长度限定符号是指通过预定义符号来完成长度限定。

? : 零次或一次。等同于{0,1}

+ : 一次或多次。等同于{1,}

\* : 零次或多次。等同于{0,}

### 3.3.4 预定义字符

在正则表达式中可以通过一些预定义字符来表示内容限定。目的是为了简化内容限定的定义。

常见的预定义字符：

字符	描述
\d	匹配一个数字字符, 等价于[0-9]。
\D	匹配一个非数字字符, 等价于[^0-9]。
\n	匹配一个换行符。
\r	匹配一个回车符。
\s	匹配任何空白字符, 包括空格、制表符、换页符等等。
\S	匹配任何非空白字符。
\t	匹配一个制表符。
\w	匹配包括下划线的任何单词字符。等价于"[A-Za-z0-9_]"。
\W	匹配任何非单词字符。等价于"[^A-Za-z0-9_]"。

### 3.3.5 正则表达式的组合定义

在正则表达式中可以通过多个内容限定与长度限定来组合定义。

示例：

必须是以字母开头，最少长度为 4,最大长度为 8。

```
"[a-z]{1}\\w{3,8}"
```

校验带有区号的电话号码的正则表达式

```
"\\d{3,4}-\\d{7,8}"  
" (\\d{3,4})-(\\d{7,8}) "
```

## 3.4 常见的正则表达式

用户名	正则表达式
邮箱	([a-z0-9A-Z]+[- \\.]?)+[a-z0-9A-Z]@([a-z0-9A-Z]+(-[a-z0-9A-Z]+)?\\.)+[a-zA-Z]{2,}
IP 地址	(25[0-5] 2[0-4]\\d [0-1]\\d\\d 1[0-9]?\\d)
URL	http(s)?://([\\w-]+\\.)+[\\w-]+(/[\\w- ./?%&=]*)?
身份证号码	(^\\d{18}\$) (\\d{15}\$)

## 4 实现验证类中的方法

### 4.1 对菜单项的验证

```
/**  
 * 对菜单输入选项的验证  
 * @param min  
 * @param max  
 * @return  
 */  
public int menuItemValidate (int min, int max ) {  
    //定义验证菜单项的正则表达式
```

```
String regex = "[1-9]{1}";

//创建键盘输入对象

Scanner scanner = new Scanner(System.in);
while(true) {

    System.out.println("请输入正确数字, 最小是: "+min+"\t"+"最大是: "+max);
    String input = scanner.nextLine();
    if(input.matches(regex)) {
        int inputNum = Integer.parseInt(input);
        if(inputNum >= min && inputNum <= max) {
            return inputNum;
        }else{

            System.out.println("您输入的菜单项不符, 请重新输入!");

        }
    }else{

        System.out.println("输入数字错误, 请检查!");

    }
}
}
```

## 4.2 对姓名的验证

```
/**
 * 对用户输入姓名的验证
 * 字母可以是大写或者小写, 长度1-10之间的
 * @return
 */
public String nameValidate ( ){

    //验证姓名的正则表达式

    String regex = "[a-zA-Z]{1,10}";

    //创建 Scanner 对象

    Scanner scanner = new Scanner(System.in);
    while(true) {
```



```

        System.out.println("请输入姓名：格式为 1-10 之间的大写或小写字母");

        String input = scanner.nextLine();
        if(input.matches(regex)){
            return input;
        }else{

            System.out.println("您当前输入的姓名有误，请重新输入！");

        }
    }
}

```

### 4.3 对年龄的验证

```

/**
 * 对用户输入年龄的验证
 * 年龄的格式要求：10-99 之间的
 * @return
 */
public String ageValidate ( ){

    //对年龄校验的正则表达式
    String regex = "[1-9]{1}[0-9]{1}";

    //创建 Scanner 对象
    Scanner scanner = new Scanner(System.in);
    while(true){

        System.out.println("请输入年龄：格式为 10-99 之间");

        String input = scanner.nextLine();
        if(input.matches(regex)){
            return input;
        }else{

            System.out.println("您输入的年龄格式有误，请重新输入！");

        }
    }
}

```

## 4.4 对性别的验证

```
/**
 * 对用户输入性别的验证
 * 性别的输入要求：男(m 或 M) 女(f 或 F)
 * @return
 */
public String sexValidate ( ){
    //对性别验证的正则表达式
    String regex ="[m|M|f|F]{1}";

    //创建 Scanner 对象
    Scanner scanner = new Scanner(System.in);
    while(true){
        System.out.println("输入性别：(男 m or M) (女 f or F)");
        String input = scanner.nextLine();
        if(input.matches(regex)){
            return input;
        }else{
            System.out.println("输入性别格式有误，请重新输入！");
        }
    }
}
```

## 4.5 对电话号码的验证

```
/**
 * 对用户输入电话号码的验证
 * 电话号码要求：允许带有区号的座机号，允许手机号
 * @return
 */
public String telNumValidate ( ){
```

```
//对电话号码验证的正则表达式

String regex = "(\\d{3,4}-\\d{7,8})|([1]{1}\\d{10})";

//创建 Scanner 对象

Scanner scanner = new Scanner(System.in);
while(true){

    System.out.println("请输入电话号码：可以是座机号或者是手机号。");

    String input = scanner.nextLine();
    if(input.matches(regex)){
        return input;
    }else{

        System.out.println("输入的电话号码有误，请重新输入！");

    }

}

}
```

## 4.6对地址的验证

```
/**
 * 对用户输入地址的验证
 * 地址格式要求：字母或者数字，长度1,50
 * @return
 */
public String addressValidate ( ){

    //对地址验证的正则表达式

    String regex = "\\w{1,50}";

    //创建 Scanner 对象

    Scanner scanner = new Scanner(System.in);
    while (true){

        System.out.println("请输入地址：格式为字母或数字，长度为1,50");

        String input = scanner.nextLine();
        if(input.matches(regex)){
            return input;
        }else{


```

```

        System.out.println("您输入的地址格式有误，请重新输入。");
    }
}
}

```

## 4.7 测试验证类中的方法

```

public static void main(String[] args) {
    TelNoteRegex regex = new TelNoteRegex();
    /*int menuItem = regex.menuItemValidate(1,6);
    System.out.println(menuItem);*/
    /*String name = regex.nameValidate();
    System.out.println(name);*/
    /*String age = regex.ageValidate();
    System.out.println(age);*/
    /*String sex = regex.sexValidate();
    System.out.println(sex);*/
    /*String telNum = regex.telNumValidate();
    System.out.println(telNum);*/
    String address = regex.addressValidate();
    System.out.println(address);
}

```

## 5 实现 APP 类中的方法

### 5.1 实现 start 方法

```

/**
 * 控制主菜单
 */
public void start(){
    Menu menu = new Menu();
    TelNoteRegex regex = new TelNoteRegex();
    Operate operate = new Operate();
    while(true){
        menu.mainMenu();
        int item = regex.menuItemValidate(1,6);
        switch(item){
            case 1: operate.addLogic();break;

```

```

        case 2: operate.searchLogic();break;
        case 3: operate.modifyLogic();break;
        case 4: operate.deleteLogic();break;
        case 5: operate.orderLogic();break;
        case 6: System.exit(0);
    }
}
}

```

## 5.2 实现 main 方法

```

/**
 * 启动电话本项目
 * @param args
 */
public static void main(String[] args) {
    App app = new App();
    app.start();
}

```

## 6 实现 Operate 类中的方法

### 6.1 实现添加记录业务逻辑方法

```

/**
 * 用户添加记录业务逻辑控制
 */
public void addLogic() {
    Menu menu = new Menu();
    TelNoteRegex telNoteRegex = new TelNoteRegex();
    while(true) {
        menu.addMenu();
        int item = telNoteRegex.menuItemValidate(1, 3);
        switch (item) {
            case 1: this.addOperation();break;
            case 2: this.showAll();break;
            case 3: return;
        }
    }
}

```

```
}
```

## 6.2 实现查找记录业务逻辑方法

```
/**
 * 用户查询记录业务逻辑控制
 */
public void searchLogic(){
    Menu menu = new Menu();
    TelNoteRegex telNoteRegex = new TelNoteRegex();
    while(true){
        menu.searchMenu();
        int item = telNoteRegex.menuItemValidate(1,7);
        switch(item){
            case 1: this.searchByName();break;
            case 2: this.searchByAge();break;
            case 3: this.searchBySex();break;
            case 4: this.searchByTelNum();break;
            case 5: this.searchByAdd();break;
            case 6: this.showAll();break;
            case 7: return;
        }
    }
}
```

## 6.3 实现修改记录业务逻辑方法

```
/**
 * 修改记录业务逻辑控制
 */
public void modifyLogic(){
    Menu menu = new Menu();
    TelNoteRegex telNoteRegex = new TelNoteRegex();
    while(true){
        menu.modifyMenu();
        int item = telNoteRegex.menuItemValidate(1,3);
        switch (item){
            case 1: this.showAll();break;
            case 2: this.modifyOperation();break;
        }
    }
}
```

```
        case 3: return;
    }
}
}
```

## 6.4 实现删除记录业务逻辑方法

```
/**
 * 删除记录业务逻辑控制
 */
public void deleteLogic(){
    Menu menu = new Menu();
    TelNoteRegex telNoteRegex = new TelNoteRegex();
    while(true){
        menu.deleteMenu();
        int item = telNoteRegex.menuItemValidate(1,4);
        switch(item){
            case 1: this.showAll();break;
            case 2: this.deleteOperation();break;
            case 3: this.deleteAllOperation();break;
            case 4: return;
        }
    }
}
```

## 6.5 实现排序记录业务逻辑控制

```
/**
 * 排序记录业务逻辑控制
 */
public void orderLogic(){
    Menu menu = new Menu();
    TelNoteRegex telNoteRegex = new TelNoteRegex();
    while(true){
        menu.orderMenu();
        int item = telNoteRegex.menuItemValidate(1,5);
        switch(item){
            case 1: this.orderName();break;
            case 2: this.orderAge();break;
        }
    }
}
```

```

        case 3: this.orderSex();break;
        case 4: this.showAll();break;
        case 5: return;
    }
}
}

```

## 6.6 实现添加记录业务功能

```

/**
 * 添加新记录信息
 */
public void addOperation(){
    TelNoteRegex telNoteRegex = new TelNoteRegex();
    String name = telNoteRegex.nameValidate();
    String age = telNoteRegex.ageValidate();
    String sex = telNoteRegex.sexValidate();
    String telNum = telNoteRegex.telNumValidate();
    String address = telNoteRegex.addressValidate();
    Person person = new Person(name,age,sex,telNum,address);
    this.list.add(person);
    person.setId(this.list.size());
}

```

## 6.7 实现查看全记录业务功能

```

/**
 * 查询全部记录
 */
public void showAll(){
    if(this.list.size() == 0){
        System.out.println("没有任何记录");
        return ;
    }
    for(int i=0;i<this.list.size();i++){
        System.out.println(this.list.get(i));
    }
}

```



## 6.8 实现按姓名查找记录业务功能

```
/**
 * 按姓名查询记录
 */
public void searchByName() {
    TelNoteRegex telNoteRegex = new TelNoteRegex();
    String name = telNoteRegex.nameValidate();
    boolean flag = true;
    for(int i = 0; i < this.list.size(); i++) {
        if(name.equals(this.list.get(i).getName())) {
            System.out.println(this.list.get(i));
            flag = false;
        }
    }
    if(flag) {
        System.out.println("没有此人记录");
    }
}
```

## 6.9 实现按年龄查找记录业务功能

```
/**
 * 按年龄查询记录
 */
public void searchByAge() {
    TelNoteRegex telNoteRegex = new TelNoteRegex();
    String age = telNoteRegex.ageValidate();
    boolean flag = true;
    for(int i = 0; i < this.list.size(); i++) {
        if(age.equals(this.list.get(i).getAge())) {
            System.out.println(this.list.get(i));
            flag = false;
        }
    }
    if(flag) {
```

```
        System.out.println("没有此人记录");
    }
}
```

## 6.10 实现按性别查找记录业务功能

```
/**
 * 按性别查询记录
 */
public void searchBySex() {
    TelNoteRegex telNoteRegex = new TelNoteRegex();
    String sex = telNoteRegex.sexValidate();
    boolean flag = true;
    for(int i=0;i<this.list.size();i++){
        if(sex.equalsIgnoreCase(this.list.get(i).getSex())){
            System.out.println(this.list.get(i));
            flag = false;
        }
    }
    if(flag){
        System.out.println("没有此人记录");
    }
}
```

## 6.11 实现按电话号码查找记录业务功能

```
/**
 * 按电话号码查询记录
 */
public void searchByTelNum() {
    TelNoteRegex telNoteRegex = new TelNoteRegex();
    String telNum = telNoteRegex.telNumValidate();
    boolean flag = true;
    for(int i=0;i<this.list.size();i++){
        if(telNum.equals(this.list.get(i).getTelNum())){
            System.out.println(this.list.get(i));
        }
    }
}
```

```

        flag = false;
    }
}
if(flag) {
    System.out.println("没有此人记录");
}
}

```

## 6.12 实现按地址查找记录业务功能

```

/**
 * 按地址查询记录
 */
public void searchByAdd() {
    TelNoteRegex telNoteRegex = new TelNoteRegex();
    String address = telNoteRegex.addressValidate();
    boolean flag = true;
    for(int i=0;i<this.list.size();i++){
        if(address.equals(this.list.get(i).getAddress())){
            System.out.println(this.list.get(i));
            flag = false;
        }
    }
    if(flag) {
        System.out.println("没有此人记录");
    }
}

```

## 6.13 实现修改指定记录业务功能

```

/**
 * 修改指定记录
 */
public void modifyOperation() {
    TelNoteRegex telNoteRegex = new TelNoteRegex();
    Menu menu = new Menu();
}

```

//对被修改的记录的进行验证，可以使用对菜单项验证的方法来完成

```
System.out.println("请输入记录的序号");

int itemNum = telNoteRegex.menuItemValidate(1, this.list.size());
menu.subModifyMenu();
int menuItem = telNoteRegex.menuItemValidate(1, 6);
switch(menuItem) {
    case 1: String name = telNoteRegex.nameValidate();
    (this.list.get(itemNum - 1)).setName(name); break;
    case 2: String age =
telNoteRegex.ageValidate(); (this.list.get(itemNum-1)).setAge(age)
; break;
    case 3: String sex =
telNoteRegex.sexValidate(); (this.list.get(itemNum -
1)).setSex(sex); break;
    case 4: String telNum =
telNoteRegex.telNumValidate(); (this.list.get(itemNum-1)).setTelNu
m(telNum); break;
    case 5: String address =
telNoteRegex.addressValidate(); (this.list.get(itemNum
-1)).setAddress(address); break;
    case 6: return;

}
}
```

## 6.14 实现删除指定记录业务功能

```
/**
 * 删除指定记录
 */
public void deleteOperation() {
    TelNoteRegex telNoteRegex = new TelNoteRegex();

    System.out.println("请输入记录序号");

    int itemNum = telNoteRegex.menuItemValidate(1, this.list.size());
    this.list.remove(itemNum - 1);

    //重新为记录设置新的序号

    for(int i=0; i<this.list.size(); i++) {
```

```
(this.list.get(i)).setId(i+1);
}

System.out.println("删除成功！请继续操作！");
}
```

## 6.15 实现删除全部记录业务功能

```
/**
 * 删除全部记录
 */
public void deleteAllOperation() {
    this.list.clear();

    System.out.println("电话中的记录已清空，请继续操作！");
}
```

## 6.16 实现比较器的排序规则

### 6.16.1 按姓名排序比较规则

```
/**
 * 按姓名排序的比较器
 */
class OrderByName implements Comparator<Person>{
    @Override
    public int compare(Person o1, Person o2) {
        return o1.getName().compareTo(o2.getName());
    }
}
```

### 6.16.2 按年龄排序比较规则

```
/**
 * 按年龄排序的比较器
 */
```

```
class OrderByAge implements Comparator<Person>{
    @Override
    public int compare(Person o1, Person o2) {
        return o1.getAge().compareTo(o2.getAge());
    }
}
```

### 6.16.3 按性别排序比较规则

```
/**
 * 按性别排序的比较器
 */
class OrderBySex implements Comparator<Person>{
    @Override
    public int compare(Person o1, Person o2) {
        return o1.getSex().compareTo(o2.getSex());
    }
}
```

## 6.17 实现排序记录

### 6.17.1 实现按姓名排序业务功能

```
/**
 * 按用户姓名排序记录
 */
public void orderName() {
    Collections.sort(this.list, new OrderByName());
    for(int i=0; i<this.list.size(); i++){
        (this.list.get(i)).setId(i+1);
    }
}
```

### 6.17.2 实现按年龄排序业务功能

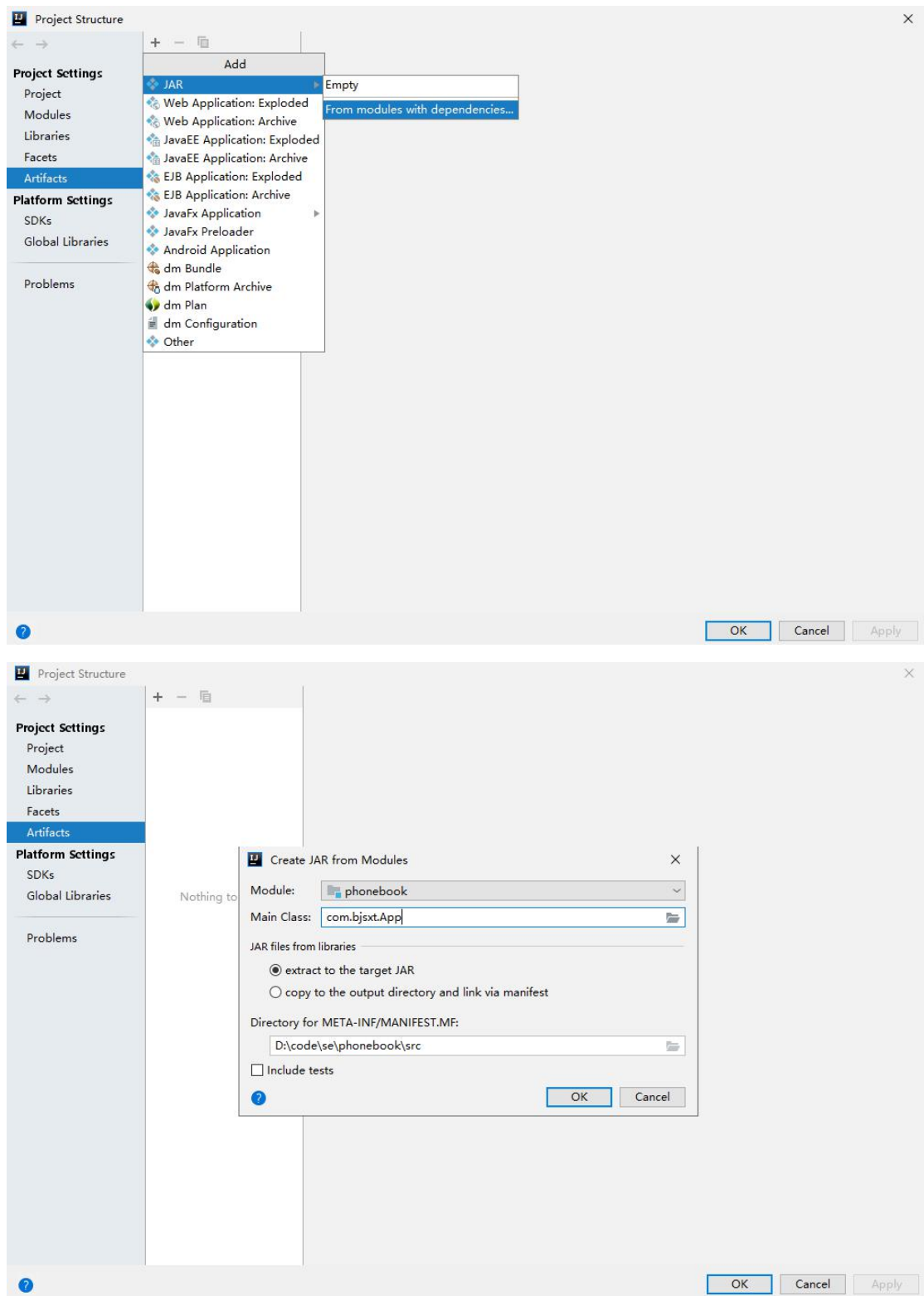
```
/**
 * 按用户年龄排序记录
 */
public void orderAge() {
    Collections.sort(this.list, new OrderByAge());
    for (int i = 0; i < this.list.size(); i++) {
        (this.list.get(i)).setId(i + 1);
    }
}
```

### 6.17.3 实现按性别排序业务功能

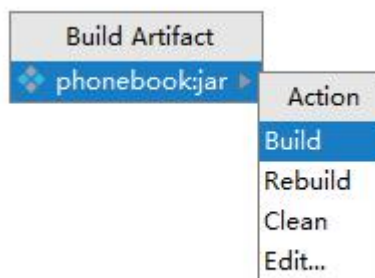
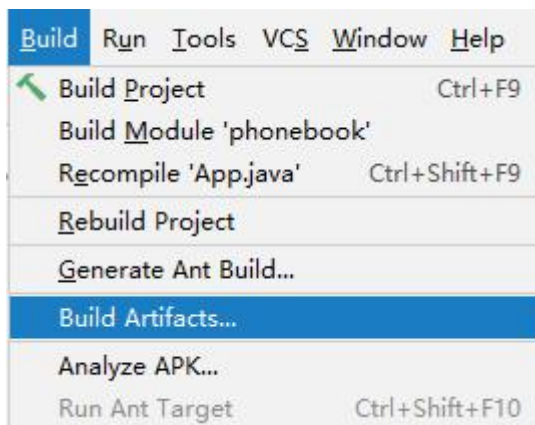
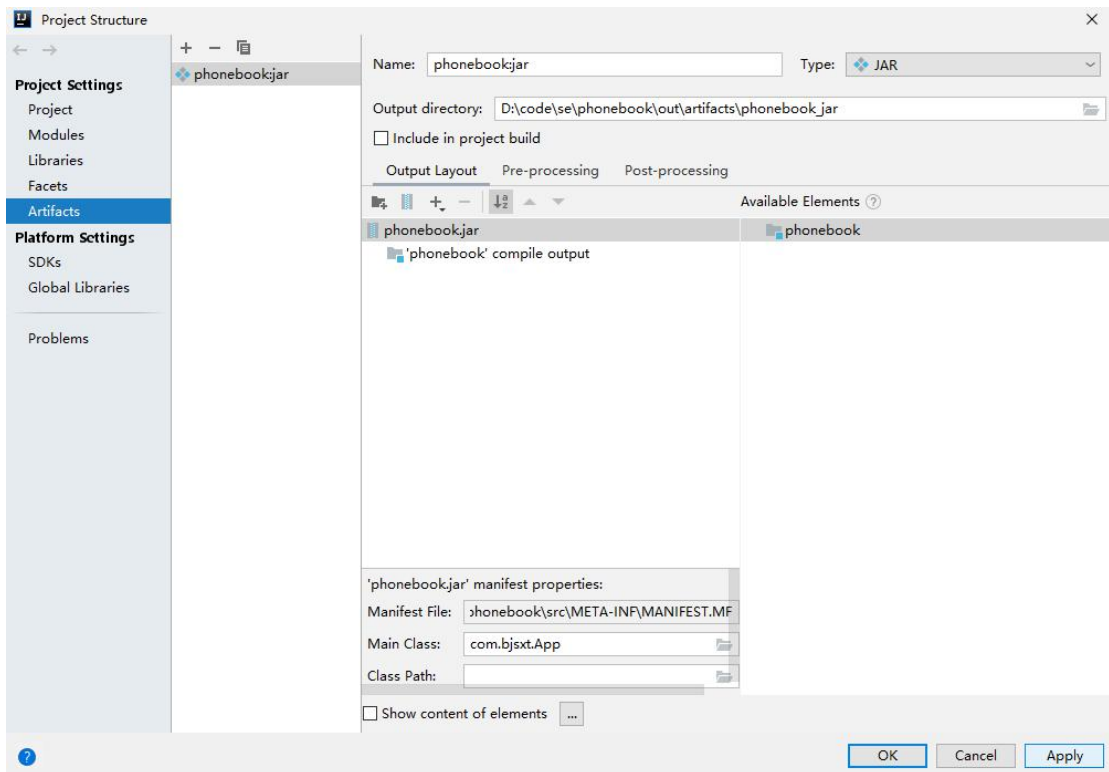
```
/**
 * 按用户性别排序记录
 */
public void orderSex() {
    Collections.sort(this.list, new OrderBySex());
    for (int i = 0; i < this.list.size(); i++) {
        (this.list.get(i)).setId(i + 1);
    }
}
```

## 7 导出项目


### 7.1 导出电话本项目的 jar 包








## 7.2 创建启动项目的批处理文件

 phonebook.jar

 start.bat

批处理文件中的内容：

```
java -jar phonebook.jar
```