实验 5：信息系统设计

一、实验基本情况

【目的要求】

本实验主要目的：

- 熟悉信息系统的系统结构与功能模块划分。
- 掌握数据的存储与查询功能的设计与实现。
- 掌握业务层功能的设计与实现。
- 掌握基本的用户界面设计。

本实验要求：

学生个人独立完成并且必需完成所有实验。

二、实验内容、实验用具

【环境用具】

Windows 10 及以上，JDK 21、IDEA 2023。

【实验内容】（以校园卡信息系统设计为例）

一、数据结构设计

1.创建程序。

打开 IDEA，创建一个名称为 Experiment5 的 Java 项目。

2.创建用户数据结构及其基本操作。

（1）创建 User 类。

定义一个实体类 User，用于表示系统的用户。用户角色分三种：卡用户，卡业务员，系统管理员，通过属性 role 来进行区分。

```
public class User {
    public String mobile; // 用户账号
    public String userName; // 用户名
    public String password; // 用户密码
    public int role ;// 0 卡用户, 1 卡业务员, 2 系统管理员
    public String cno; //校园卡号
    public User(String mobile, String password, String userName, int role, String
cno) {
        this.mobile = mobile;
        this.password = password;
        this.userName = userName;
        this.role = role;
        this.cno = cno;
    }
}
```

（2）创建 UserDatas 类。

定义一个用户数据类 UserDatas，用于表示对用户数据结构的基本操作。

```
// 此类完成对 users 对象中数据进行查找、添加、修改、删除等操作
public class UserDatas {
```

```
private static ArrayList<User> users = null; // 所有用户信息保存在 users 对象
// 根据 moblie 查找用户，返回用户的索引位置
// 此处采用的是最简单的顺序查找法，请将其升级为二分查找算法
public int findUser(String mobile) {
   if(users == null) return -1;
   if(mobile==null) return -1;
   int index = -1;
   for(int i=0; i<users.size(); i++) {
      if( mobile.equals(users.get(i).mobile)) {
         index = i;
         break;
      }
   }
   return index;
}
// 新增一个用户
// 要实现二分查找算法，则需要对用户数据进行排序，请自行补充完整
public void addUser(User user) {
   users.add(user);
}
// 修改 users 中一个用户信息
public void modifyUser(User user) {
   int index = findUser(user.mobile); // 查找 user 在 users 中的索引位置
   users.set(index, user); // 修改索引位置中数据
}
// 删除 users 中用户
public void removeUser(User user) {
   int index = findUser(user.mobile); // 查找 user 在 users 中的索引位置
   users.remove(index); // 删除
}
// users 对象数据保存到文件
public void saveUsers() {
   UserFile file = new UserFile();
   file.save(users);
}
// 从文件读取所有的用户信息到 users 对象
public static ArrayList<User> getUsers() {
   if(users == null) {
      UserFile file = new UserFile();
      users = file.acquire();
   }
   return users;
}
// 清除数据
```

```
    public static void clear() {
        users = null;
    }
}
```

3.创建××数据结构及其基本操作。

与创建用户数据结构类似，根据系统的功能定义，创建其他类型的数据结构，例如，校园卡数据结构 Card 类、卡交易数据结构 Transaction 类等等，请自行设计完成。

二、数据的文件存储

为了防止每次退出程序后系统数据丢失，需要对系统的数据进行存储。要实现数据存储，可以采用数据库或者文件存储等方式。此处，为了简单起见，采用文件存储。

1.创建文件数据存取的基础类 FileHandler。

系统中为每一种数据结构都单独创建一个文件用于保存数据，虽然文件名称以及所操作的数据结构类型不同，但是都是属于对文件的操作，具有一些共同的属性，将这些属性提取出来创建一个基类 FileHandler，以方便后续使用。

```
//文件数据的读取与保存
public class FileHandler {
    protected static String separator = "\t";// 数据项之间的分隔符
    protected String filename = null; // 数据保存的文件名
    protected ArrayList<String> datas; // 每条记录数据以字符串的形式保存
    // 用于格式化时间
    protected SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    public FileHandler(String filename) {
        this.filename = filename;
        this.datas = new ArrayList<String>();
    }
    // 从文件中读取数据
    public void read() {
        BufferedReader reader = null;
        // 读取文件数据前，清除 datas 中的值
        datas.clear();
        try {
            // 以缓冲流方式读取文件中数据
            reader = new BufferedReader(new FileReader(filename));
            String line = null;
            while ((line = reader.readLine()) != null) {
                datas.add(line); // 每条记录（每行数据）保存为一个字符串对象
            }
        }
        catch(Exception ex) {
            ex.printStackTrace();
        }
        finally {
            if(reader != null)
```

```
            try {
               reader.close();
            } catch (Exception e) {
               e.printStackTrace();
            }
      }
   }
   // 将数据保存到文件中
   public void write() {
      // 文件内容
      String content = listToString(); // datas 对象数据转换成字符串保存
      if(content == null) return;
      FileWriter writer = null;
      try {
         writer = new FileWriter(filename);
         writer.write(content);
      } catch (Exception e) {
         e.printStackTrace();
      }
       finally {
          try {
           writer.close();
         } catch (Exception e) {
            e.printStackTrace();
         }
       }
   }
   // datas 对象数据转换成字符串
   public String listToString() {
      if(datas == null) return null;
      String content = "";
      for(int i=0; i<datas.size(); i++) {
         content += datas.get(i);
      }
      return content;
   }
}
```

2.创建用户数据结构的文件存储类 UserFile。

定义一个 UserFile 类，用于实现对用户数据的文件读取与保存操作，该类继承自父类 FileHandler。

```
// 用户文件数据的读取与保存
public class UserFile extends FileHandler{
   public UserFile() {
      super("users.dat");
   }
```

```
// 读取文件数据保存在 users 中
public ArrayList<User> acquire() {
   datas.clear(); // 读取数据前先清除 datas
   read(); // 读取数据
   if(datas == null) return null;
   ArrayList<User> users = new ArrayList<User>();
   for(int i=0; i<datas.size(); i++) {
      User user = dataFromString(datas.get(i));
      if(user != null)
         users.add(user);
   }
   return users;
}
// 将 users 中数据写入文件
public void save(ArrayList<User> users) {
   if(users.isEmpty()) return ;
   datas.clear(); // 先清除
   for(int i=0; i<users.size(); i++) {
      datas.add(dataToString(users.get(i)));
   }
   write(); // 写入文件
}
// 将读取的字符串数据转换成用户对象
private User dataFromString(String line) {
   User user = null;
   // 数据按照 separator 分隔符分隔
   String[] item = line.split(separator);
   String cno = "";
   if(item.length == 5) cno = item[4];
   // 用户没有卡号时，数组长度为 4
   if(item.length >=4 ) {
      user = new User(item[0],item[1],item[2],Integer.parseInt(item[3]),cno);
   }
   return user;
}
// 将用户对象数据转换成字符串
private String dataToString(User user) {
   String line=null;
   if(user != null) {
      // 使用 separator 符号连接数据项
      line = user.mobile+separator+user.password+separator
          +user.userName+separator+user.role+separator+user.cno+separator+'\n';
   }
   return line;
```

```
    }
}
```

3.创建××数据结构的文件存储类。

与创建用户数据结构的文件存储类相类似，根据系统的功能定义，创建其他数据类型的文件存储类，例如，校园卡数据结构文件存储类 CardFile、卡交易数据结构文件存储类 TransactionFile 等等，请自行设计完成。

三、业务层设计

1.创建用户业务基础类 UserService。

系统共有 3 种不同的用户角色：系统管理员、卡业务员、卡用户。系统为每一种用户角色都单独创建一个业务类用于实现该用户角色的相关业务。虽然不同的用户角色业务有所不同，但是也具有一些共同的业务（例如：登录、注销、修改密码等），将这些属性提取出来创建一个基类 UserService，以方便后续使用。

```
//用户业务基础类，它是 ManagerService、OperatorService 和 CustomerService 的父类
public class UserService {
    public static ArrayList<User> users = null;
    public static ArrayList<Card> cards = null;
    public static ArrayList<Transaction> transactions = null;
    public static int curUserIndex = -1;// 当前操作用户索引
    public boolean change = false; // 标记业务操作是否更改数据  false:未更改  true: 更改
    private Scanner scanner = new Scanner(System.in);
    protected UserDatas userDatas = new UserDatas();
    protected CardDatas cardDatas = new CardDatas();
    protected TransactionDatas transDatas = new TransactionDatas();
    public DecimalFormat df = new DecimalFormat("#.00"); // 用于格式化金额
    // 用于格式化时间
    public SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    public void getDatas() {
        users = UserDatas.getUsers();
        cards = CardDatas.getCards();
        transactions = TransactionDatas.getTransactions();
    }
    public int getCurUserIndex() {
        return curUserIndex;
    }
    // 用户登录
    public boolean login() {
        System.out.println("mobile:");
        String mobile = scanner.next();
        System.out.println("password:");
        String password = scanner.next();
        int index = userDatas.findUser(mobile);
        if(index < 0) {
            System.out.println("用户不存在！");
            return false;
        }
```

```
        User user =  users.get(index);
        boolean pass =user.password.equals(password);
        if(!pass) {
            System.out.println("用户密码错误！");
            return false;
        }
        curUserIndex = index;
        return true;
    }
    // 修改密码
    public boolean modifyPassword() {
        User user = users.get(curUserIndex);
        System.out.println("请输入原密码：");
        String oldPwd = scanner.next();
        if(!oldPwd.equals(user.password)) {
            System.out.println("原密码不正确！");
            return false;
        }
        System.out.println("请输入新密码：");
        String newPwd = scanner.next();
        System.out.println("再次确认密码：");
        String newPwd2 = scanner.next();
        if(!newPwd.equals(newPwd2)) {
            System.out.println("两个输入的密码不同，修改密码失败\n");
            return false;
        }
        user.password = newPwd;
        users.set(curUserIndex, user);
        System.out.println("密码修修改成功！");
        change = true; // 数据更改
        return true;
    }
    // 用户注销
    public void logout() {
        // 保存数据
        saveDatas();
        // 清空全局变量/对象数据
        UserDatas.clear();
        CardDatas.clear();
        TransactionDatas.clear();
        curUserIndex = -1;
    }
    // 用户退出
    public void exit() {
```

```
        // 保存数据
        saveDatas();
        // 程序终止
        System.exit(0);
    }
    /** 保存数据，由于系统管理员、卡业务员、卡用户保存数据的操作不一样
     *  程序逻辑在各个子类中实现
     */
    public void saveDatas() {
    }
}
```

2.创建系统管理员业务类 ManagerService。

定义一个 ManagerService 类，用于系统管理员的业务功能，该类继承自父类 UserService。

```
//实现系统管理员的业务功能
public class ManagerService extends UserService{
// 删除用户时，需要同时删除用户的卡号和交易记录，需要单独标记
    public boolean delete = false;
    private Scanner scanner = new Scanner(System.in);
    // 新增用户
    public void addUser() {
        String mobile;
        int index = -1;
        while (true){
            System.out.println("手机号(11 位):");
            mobile = scanner.next();
            index = userDatas.findUser(mobile);
            if(index < 0)
                break;
            System.out.println("用户["+mobile+"]已存在，请重新输入");
        }
        System.out.println("登录密码:");
        String password = scanner.next();
        System.out.println("用户名:");
        String userName = scanner.next();
        System.out.println("角色(0 卡用户, 1 卡业务员, 2 系统管理员):");
        int role = scanner.nextInt();
        User user = new User(mobile, password, userName, role, "");
        userDatas.addUser(user);
        change = true; // 数据更改
    }
    // 删除用户
    public void removeUser() {
        //系统中只有 admin 用户，不进行删除
        if(users.size() ==1) {
```

```
            return;
        }
        String mobile;
        int index = -1;
        while (true){
            System.out.println("手机号(11 位):");
            mobile = scanner.next();
            index = userDatas.findUser(mobile);
            if(index >= 0)
                break;
            System.out.println("用户["+mobile+"]不存在，请重新输入");
        }
        User user = users.get(index);
        if(user.mobile.equals(users.get(curUserIndex).mobile)) {
            System.out.println("不能删除当前用户!");
            return;
        }
        if(users.get(curUserIndex).role == 2) {
            System.out.println("不能删除[admin]系统管理员用户!");
            return;
        }
        userDatas.removeUser(user);
        change = true; // 数据更改
        //删除用户时，需要删除对应的卡号和交易记录
        if(!user.cno.isEmpty()) {
            cardDatas.removeCard(user.cno);
            transDatas.removeTransactionByCno(user.cno);
            delete = true; // 删除用户的卡号和交易记录，需要标记
        }
    }
    // 显示所有用户
    public void showUsers() {
        System.out.println("\n 手机号\t\t 密码\t\t 用户名\t\t 身份\t\t 卡号");
        for(int i=0; i<users.size(); i++) {
            User user = users.get(i);
            String roleName ;
            switch(user.role) {
                case 0:
                    roleName = "卡用户";
                    break;
                case 1:
                    roleName = "卡业务员";
                    break;
                case 2:
```

```
                roleName = "系统管理员";
                break;
            default:
                roleName = "卡用户";
        }
        System.out.println(user.mobile+"\t\t"+user.password+"\t\t"
                +user.userName+"\t\t"+roleName+"\t\t"+user.cno);
    }
  }
  @Override
  public void saveDatas() {
    if(!change) return ;
    System.out.println("用户信息已经被修改，是否保存？(Y/N)");
    String yn = scanner.next();
    if (yn.equalsIgnoreCase("Y")) {
      userDatas.saveUsers();
      if(delete) { //如果是删除用户，还需要删除卡、交易记录
        cardDatas.saveCards();
        transDatas.saveTransaction();
      }
    }
    // 恢复标记
    change = false;
    delete = false;
  }
}
```

3.创建××业务类。

与创建系统管理员业务类相类似，根据系统的功能定义，创建其他用户角色的业务类，例如，卡业务员业务类 OperatorService、卡用户业务类 CustomerService 等等，请自行设计完成。

四、界面设计与系统集成

1.创建系统菜单的显示内容。

定义一个 Menu 类，用于显示系统菜单的内容，由于不同的用户角色所对应的业务功能有所不同，因此需要建立多个不同的菜单。

```
public class Menu {
  // 登录前菜单
  public static void preLoginMenu(){
    System.out.println("\n\n\t1. 登录");
    System.out.println("\t2. 退出");
    System.out.println("\n\t 请选择您的操作：");
  }
  // 系统管理员菜单
  public static void managerMenu(){
    System.out.println("\n\n\t1. 修改密码");
```

```
        System.out.println("\t2. 浏览用户信息");
        System.out.println("\t3. 添加新用户");
        System.out.println("\t4. 删除用户");
        System.out.println("\t5. 注销");
        System.out.println("\t6. 退出");
        System.out.println("\n\t 请选择您的操作：");
    }
    // 卡业务员菜单
    public static void operatorMenu(){
        System.out.println("\n\n\t1. 修改密码");
        System.out.println("\t2. 浏览校园卡信息");
        System.out.println("\t3. 开卡");
        System.out.println("\t4. 挂失");
        System.out.println("\t5. 解挂");
        System.out.println("\t6. 注销");
        System.out.println("\t7. 退出");
        System.out.println("\n\t 请选择您的操作：");
    }
    // 卡用户菜单
    public static void customerMenu(){
        System.out.println("\n\n\t1. 修改密码");
        System.out.println("\t2. 充值");
        System.out.println("\t3. 消费");
        System.out.println("\t4. 我的校园卡");
        System.out.println("\t5. 查看交易记录");
        System.out.println("\t6. 注销");
        System.out.println("\t7. 退出");
        System.out.println("\n\t 请选择您的操作：");
    }
}
```

2.定义菜单的跳转与操作逻辑。

定义一个 Action 类，用于表示不同菜单之间的跳转以及各个菜单的操作逻辑。

```
// 系统中菜单操作的逻辑代码
public class Action {
    UserService usersService = new UserService();
    ManagerService managerService = new ManagerService();
    OperatorService operatorSerivce = new OperatorService();
    CustomerService customerService = new CustomerService();
    private Scanner scanner = new Scanner(System.in);
    // 系统启动时，显示菜单，并提供相应操作
    public void start() {
        while(true) {
            Menu.preLoginMenu(); // 显示登录前菜单项
            int item = scanner.nextInt();
```

```
        if(item == 1 ) {
            usersService.getDatas(); // 获取数据
            boolean pass = usersService.login(); // 用户登录
            if(!pass)
                continue;
            User user = usersService.users.get(usersService.getCurUserIndex());
            if(user.role == 0) { // 卡用户登录
                customerAction();
            }
            else if(user.role == 1){ // 卡业务员登录
                operatorAction();
            }
            else if(user.role == 2){ // 系统管理员登录
                managerAction();
            }
        }
        else if(item ==2) {
            usersService.logout();
            break;
        }
    }
}
// 系统管理员的菜单项及其操作逻辑
public void managerAction() {
    boolean flag = false;
    while(true) {
        Menu.managerMenu(); // 显示系统管理员菜单项
        int item = scanner.nextInt();
        switch(item) {
            case 1:
                managerService.modifyPassword(); // 修改密码
                break;
            case 2:
                managerService.showUsers(); // 显示所有用户
                break;
            case 3:
                managerService.addUser(); // 新增用户
                break;
            case 4:
                managerService.removeUser(); // 删除用户
                break;
            case 5:
                managerService.logout(); // 注销
                flag = true;
```

```
            break;
        case 6:
            managerService.exit(); // 退出程序
            flag = true;
            break;
      }
      if(flag)
        break;
    }
  }
  // 卡业务员菜单项及其操作逻辑
  public void operatorAction() {
    boolean flag = false;
    while(true) {
      Menu.operatorMenu(); // 卡业务员菜单项
      int item = scanner.nextInt();
      switch(item) {
      case 1:
        operatorSerivce.modifyPassword(); // 修改密码
        break;
      case 2:
        operatorSerivce.showCards(); // 显示所有卡信息
        break;
      case 3:
        operatorSerivce.createCard(); // 开卡
        break;
      case 4:
        operatorSerivce.frozenCard(); // 挂失
        break;
      case 5:
        operatorSerivce.activateCard();    // 解挂
        break;
      case 6:
        operatorSerivce.logout(); // 注销
        flag = true;
        break;
      case 7:
        operatorSerivce.exit(); // 退出
        flag = true;
        break;
    }
    if(flag)
      break;
    }
```

```
    }
    // 卡用户菜单及其操作逻辑
    public void customerAction() {
        boolean flag = false;
        while(true) {
            Menu.customerMenu(); // 卡用户菜单
            int item = scanner.nextInt();
            switch(item) {
            case 1:
                customerService.modifyPassword(); // 修改密码
                break;
            case 2:
                customerService.recharge(); // 充值
                break;
            case 3:
                customerService.consume(); // 消费
                break;
            case 4:
                customerService.showCard(); // 显示卡信息
                break;
            case 5:
                customerService.showTransaction(); // 显示交易记录
                break;
            case 6:
                customerService.logout(); // 注销
                flag = true;
                break;
            case 7:
                customerService.exit(); // 退出
                flag = true;
                break;
        }
        if(flag)
            break;
        }
    }
}
```

3.创建主类。

定义一个主类，在主类中启动系统。如果是首次运行，需要进行一次初始化，目的是创建一个系统管理员账号，以方便后续操作。

```
public class Main {
    public static void main(String[] args) {
        //系统首次运行时进行一次初始化，后续不要再重复运行此函数
        //init();
```

```
        Action action = new Action();
        action.start();
    }
    // 首次使用系统时初始化数据
    public static void init() {
        User user = new User("admin", "admin", "admin", 2, "");
        ArrayList<User> users = new ArrayList<User>();
        users.add(user);
        UserFile userDatas = new UserFile();
        userDatas.save(users);
    }
}
```