**家谱管理系统**

**一、功能**

（1）普通用户、超级管理员不同角色，不同角色登录后的权限各不相同，普通用户可以进行查询；超级管理员有对所有成员增加、删除和修改的权限。

现在的初始超级管理员：admin 123456

初始普通用户：user555 123123

（2）家谱中成员的信息中包含姓名、出生日期、婚否、地址、健在否、死亡日期（若其已死亡）等

（3）数据存储于文本文件中。（xxx.data）

每次正常关闭才可以保存修改成功

（4）增删查改。可按照姓名查询，输出成员信息（包括其本人、父亲、孩子的信息、所在辈分）；按照出生日期查询成员名单。

（5）按出生日期对家谱中所有人排序。

（6）打开家谱时，提示当天生日的健在成员。

（7）输入任意两个姓名，能够查询这两个姓名之间的是否具有直系祖先后裔关系。

（8）以图形方式显示家谱。

**二、数据结构说明**

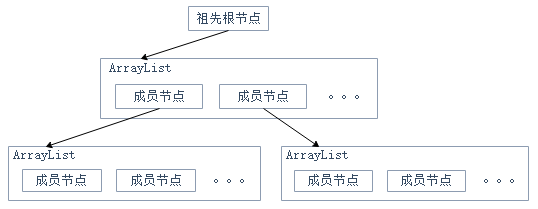


图10 树的简单图示

**三、算法设计**

1 总体流程设计

可视化界面跳转流程设计如图11

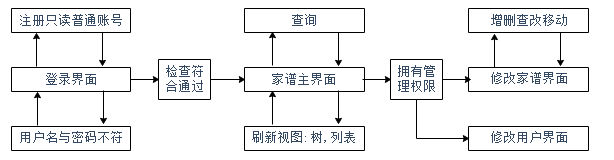


图11 总体流程设计

2 数据序列化和反序列化

这是一个对于普遍对象的写入和读取部分。

写：首先以文件输出位置为参数建立File类型对象，将其作为参数传入，建立文件输出流FileOutputStream对象，将其作为参数传入，建立对象输出流ObjectOutputStream对象，然后调用flush()方法把缓冲区的内容强制输出，最后将文件输出流close()，若不主动释放，它会持续占有资源。

读：创建流的方式类似写的前三步，然后调用ObjectInputStream实例的readObject()方法读取，最后关闭流。值得注意的是，在调用读取方法时读取Object类型，需要强制转换为存储时的类型。

在本程序中，每次进入登录界面读取可变数组类型的存储所有用户信息的数据，在登入后读取家谱树数据，在修改用户界面关闭时写入用户数据，在程序主界面正常关闭时进行两者的写入。

3 增删查改移动查询直系

首先要说明的是在实现中程序禁止了名字的重复，可重复部分在后面进行设计与分析。在对树结构有调整的操作中都需要再当前节点单独设置父母节点。在设置上父母是分离的两方，在需要绘制父母双方时可以只调整绘制程序实现。

查找：这是一个递归的过程，传递的参数为Member和名字，结束条件为要找的名字等同于当前Member的名字，内部递归的调用为对当前成员的每个子嗣作为Member传入参数。在此方法外部封装：设置重写方法传入家谱树根进行全局的查找。同时，因为我们计算过用于生日排序的链表，也可以通过链表查找。

插入（增加）：每次先确认要插入的父母节点存在，将其设为要插入节点的父或母，将其加入插入节点的子女数组。

删除：与上述类似，递归地调用自己，当名字符合时获得父母节点然后将自己从父母节点的子女数组删除。这个操作会删除所有下属的节点。同时，因为是一个无返回的方法，删除时会把所有此名的节点全部删除。

修改不做赘述。

移动：先查找符合的需要转移的节点和目标父母节点，都存在时将其删除，而后加入目标子女，最后将改动成员及其子树递归重新计算世代数。

检查是否是直系亲属：两成员互相以对方为基点，在其子孙树下查找自己，如果有一方找到就是直系亲属。

4 可以重复的树的设计（未实现）：

增加与删除全部同名没有问题，只是禁止了。

查找：在tree下维护一个静态列表，记录每个名字的个数，每次加入删除时需要修改，同时被删除的子孙都需要修改个数，像维护每一代个数（已实现）一样做成方法每次刷新。

查找与删除的定位：将在editcontroller中调用的方法定向为记录的界面右侧的member实例，每次给用户看一个，确定后对当前member删除，然后查找时做成查找与查找下一个，如果有同名可以查找下一个实例，根据list中的个数，通过在每个member类中新添加一个flag记录有没有被查过来实现，每次结束需要将flag归位，可以再次遍历来全部重置或者再开一个member类型数组记录已经查过的实例。

查找与删除的另一个思路：在基础的树外部再建立二维都可变数组，每个名字为第一参数，内数组存同名个体们，然后都通过这个二维数组定位到记录。

修改：通过查找下一个定位到个体后没有问题。

移动：同上，定位两个个体让用户确定。

5 用户的增删查改

与上述类似，且使用数组，更简单。仅允许管理员操作，不可删除当前账户，对密码限制必须大于六位，用户名是否重复判断。

6 家谱排序以及提示健在生日者

在树的类中单独维护一个静态的链表来记录排序，在Member类中重写compareTo()方法，变成比较两个成员生日大小。已经有一个或多个的情况，比较第一个，比第一个小插入0位置；已经有多个的情况，比较前后相邻两个，大于前而小于后就插入其中；如果不满足前两个情况，在末尾插入。对树所有子女成员递归，排序所有人到这个链表。提示健在生日着只需要遍历，对每一个人调用getIsAlive()确认存活，然后确认生日是否一致，满足的打印提示。

7 图形化家谱树算法

每次首先清空画布，重新统计树每一层（代）的人数，记录在generationCount数组中，用以计算每一层节点之间的距离，统一计算，存储于xGap数组中。Count是在绘制时记录画的进度，每画一个节点在其层数已画计数加一。每次对所有子女先计算并画线，然后递归调用方法本身。

另外，在本方法实现中也判断了当日生日且存活的成员并将其框画为蓝色。

间隔具体计算式子：

yGap = ((int)canvasHeight-2\*startY)/(generationCount.size()+1)-gridHeight;

xGap[i] = (((int)canvasWidth-2\*startX))/(generationCount.get(i))-gridWidth;

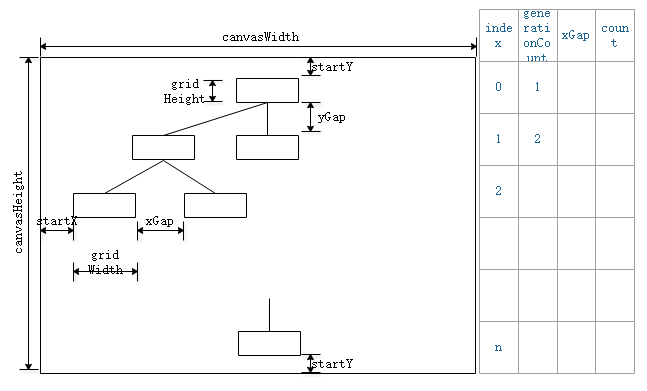


图12 布局与对应数组变量

8 初始化处理：

初始化都是内置的单独方法，在图形化界面不可调用。树的初始化建立固定的根节点，用户信息的初始化建立固定用户名为admin，且密码为123456的管理员账户。

1. **详细设计（核心部分）**

见代码

1. **测试数据及其结果分析**

在红楼梦贾家的基础上编造了相关信息并输入，信息部分如图13，之后是操作的截图。

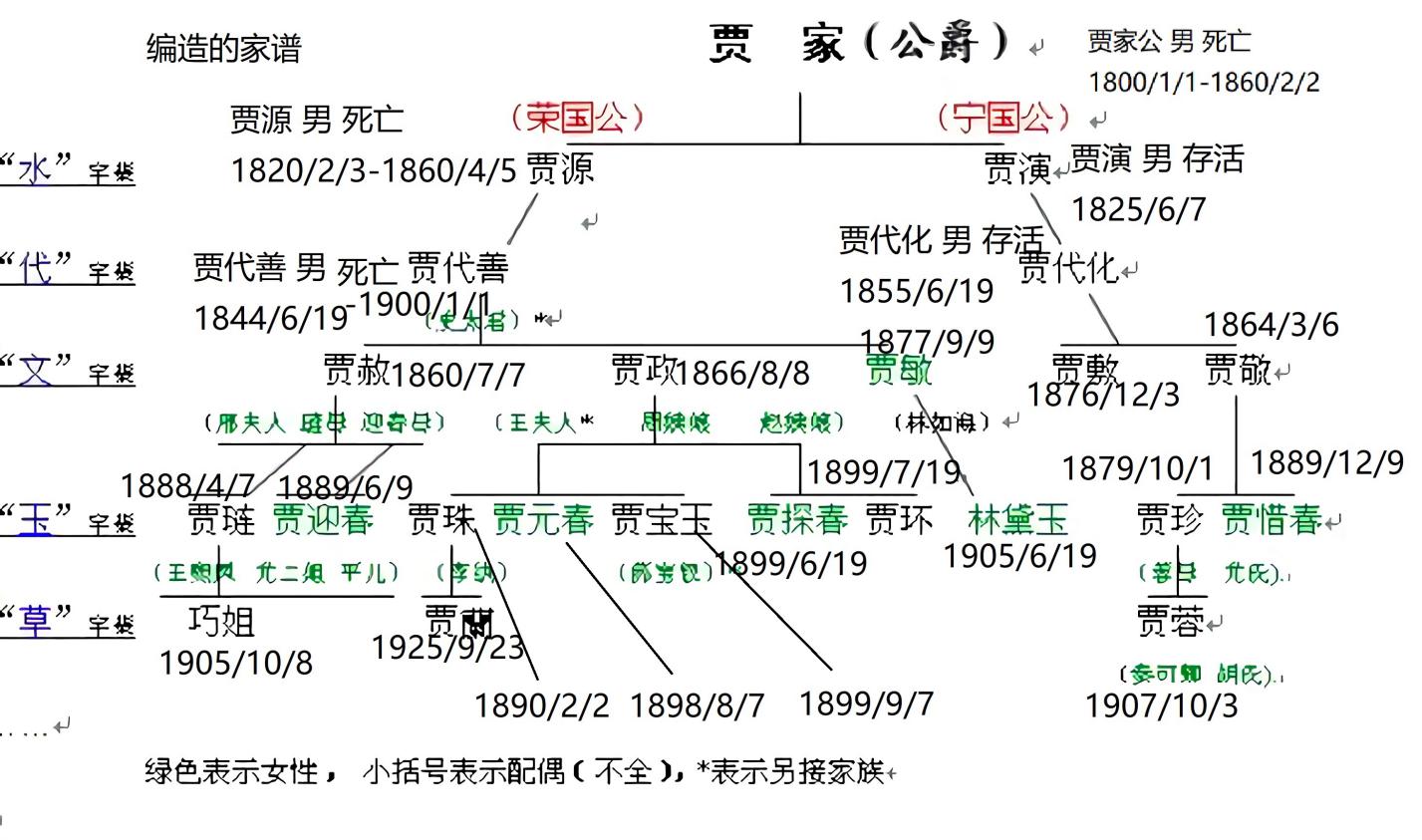


图13 在贾家基础上编造相关数据进行测试

图14 登录界面 图15 注册普通用户及错误提示

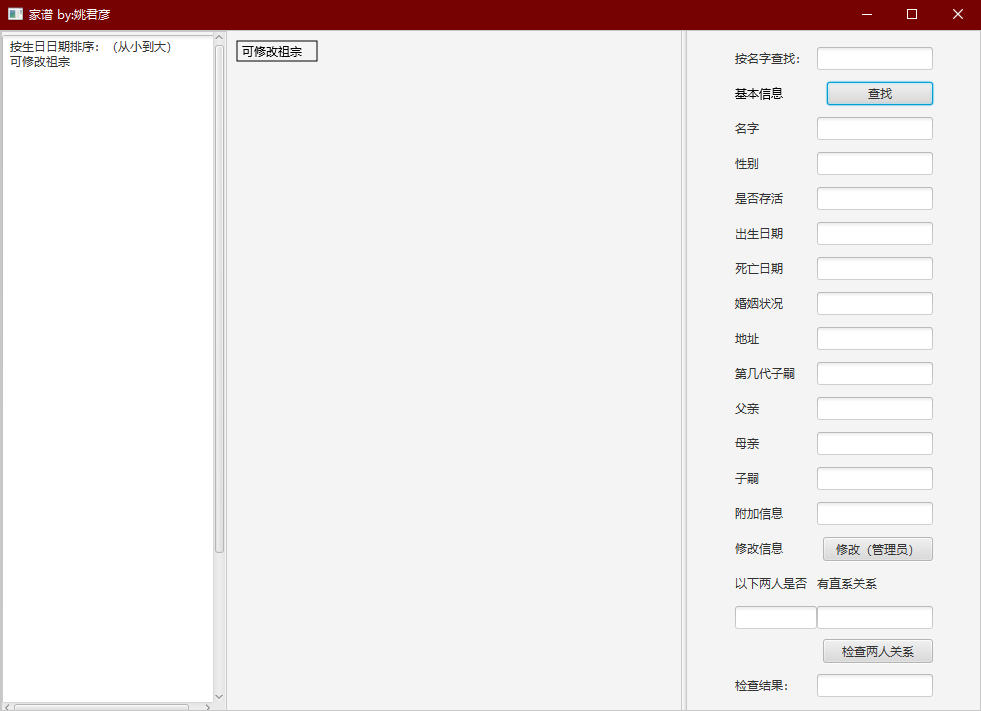


图16 初始化后的主界面



图17 修改界面以及添加操作（使用管理员账户登录）

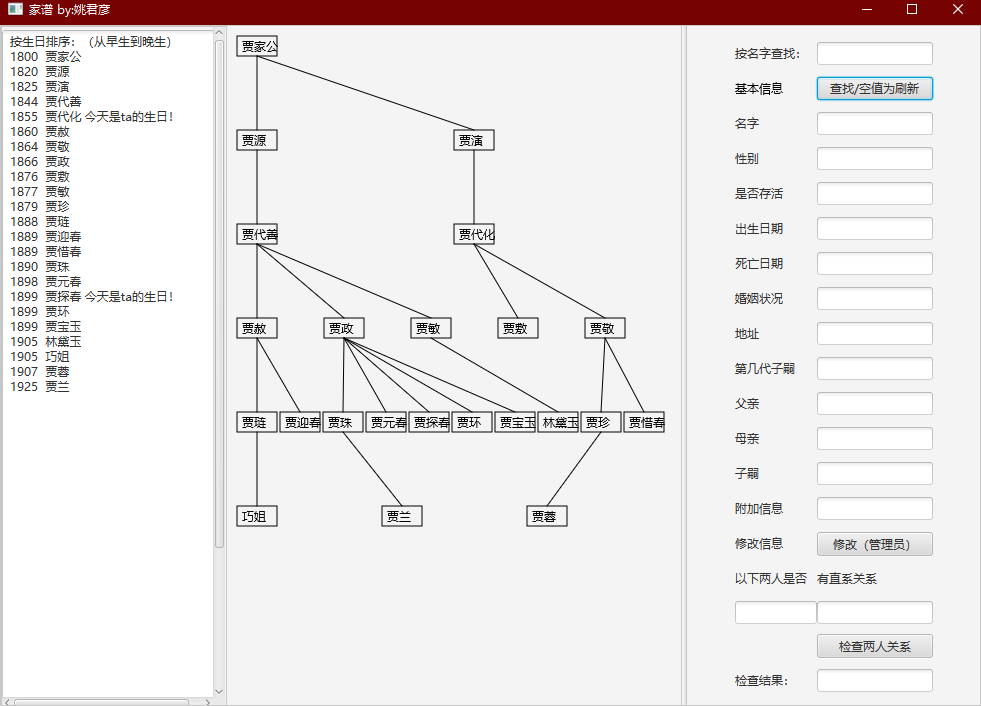


图18 全部添加完成，刷新后界面

图19 查找例子 图20 查询是否有直系关系

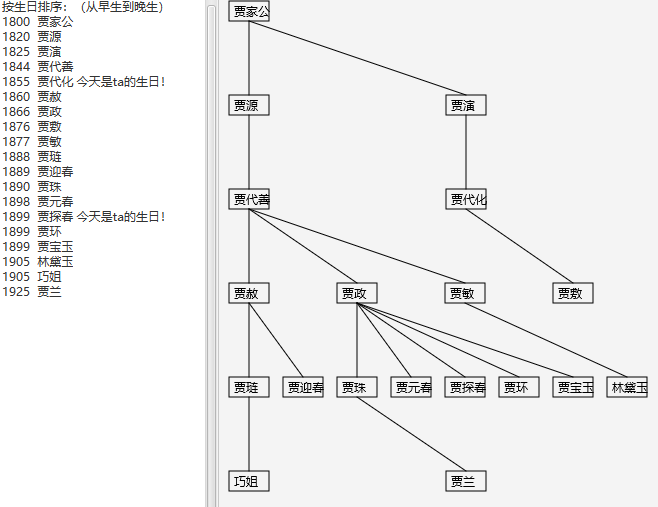


图21 删除贾敬结果

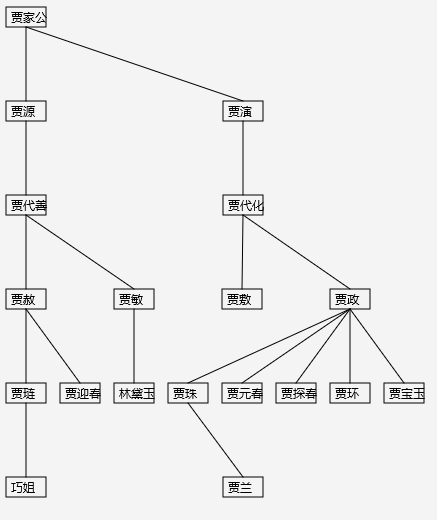


图22 移动贾政结果

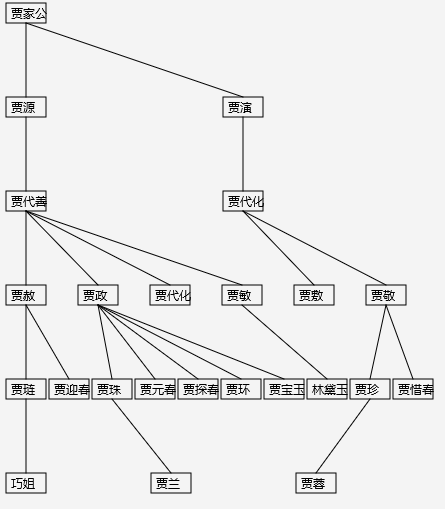


图23 添加重复的贾代化（在可执行文件已禁止）

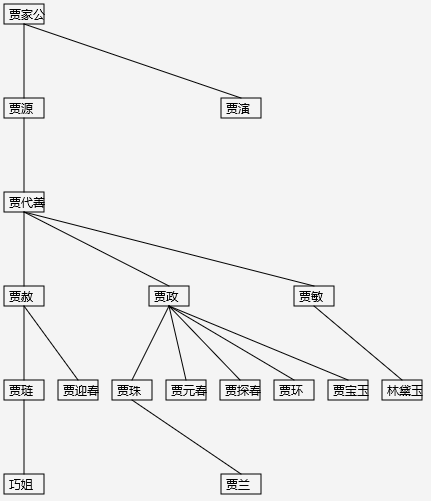


图24 删除重复的贾代化

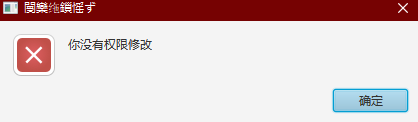


图25 以普通用户尝试进入家谱修改界面或用户修改界面的弹窗



图26 在修改界面建立新账户

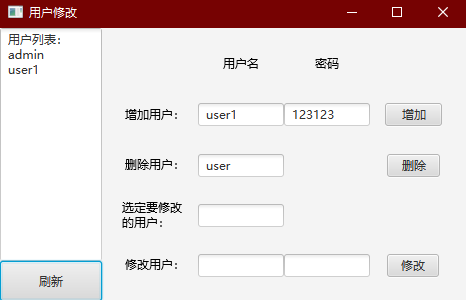


图27 删除账户



图28 选定后修改用户及密码