系统概要：

这是可以辨别直系（爷爷-爸爸-兄弟（哥哥-弟弟）-儿子-孙子）和师生关系（老师-同学-学生）的简单系统。

数据结构：二叉树

思路：将关系分成亲属（先不区分哥哥弟弟）/非亲属两棵大树分别建树。查询时分为亲属树/非亲属分别得到关系，特别的是：当亲属关系为兄弟时 再次判断是否能确定为哥哥/弟弟（结构：链表）

建树：

结点代码：

Node（data）{

this.data = data;

Node left = null;

Node right = null;

Node parent = null;

}//三叉链表

其中：Left:儿子 right：兄弟 这样关系和结构之间存在一一映射。

兄弟

爷爷

儿子

爸爸

第一步：将三种句型：

**<人名>是<人名>的<人物关系>**

**<人名>的<人物关系>是<人名>**

**<人名>的<人物关系>是<人名>的<人物关系>**

统一为第一句的句型

然后将句子切割成**<人名> <人名> <人物关系> 用String**数组保存。

特别的：对于跨代/第三举行引入”未知者”进而达到统一处理的目的

Eg：小明的爸爸是小花的爷爷

* 未知者是小明的爸爸
* 未知者是小花的爷爷

这样有效降低了人为去判断关系逻辑 接口也比较好写

第二步：遍历root数组（存每棵树的头结点）判断String[0]和String[1]是否在树上。

通过反馈的结果进行下面三种业务：

1：两个点其中之一之前已经存在->旧点为锚点 新点附进去

2：两个点都是新点->新建一棵树

3：两个点都是旧点且不同一棵树->两棵树合并（建树最难点 下谈）。

查询：总体思路对每个点查询：

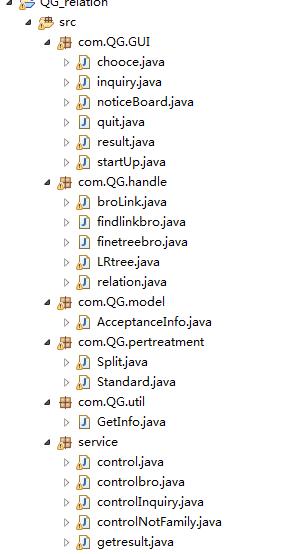
1：向右找（兄弟->进而再判断是哥哥还是/弟弟）

2：往下找（儿子）。

下面是程序的一些截图/演示：

功能：仅查询

文件夹结构：



GUI包：noticeBoard ---- 公告牌 主要是提醒用户一些事项。 startup --- 启动程序（读取文本）

chooce – 选择功能（目前仅支持查询）

inquiry ---查询

result – 结果显示

handle包： LRtree -- 亲属/非亲属树

broLink – 兄弟链表（判断哥哥/弟弟）

relation – 判断结果时用的类

findtreebro –- 判断结果的类 方法是基于树的

findlinkbro -- 判断结果的类 方法是基于链表的

model 包 ： 接收文本句子时用的

pretreatment 包：Spilt – 将句子切割

Standard –将句式统一

Service 包： control – 控制建立亲属树

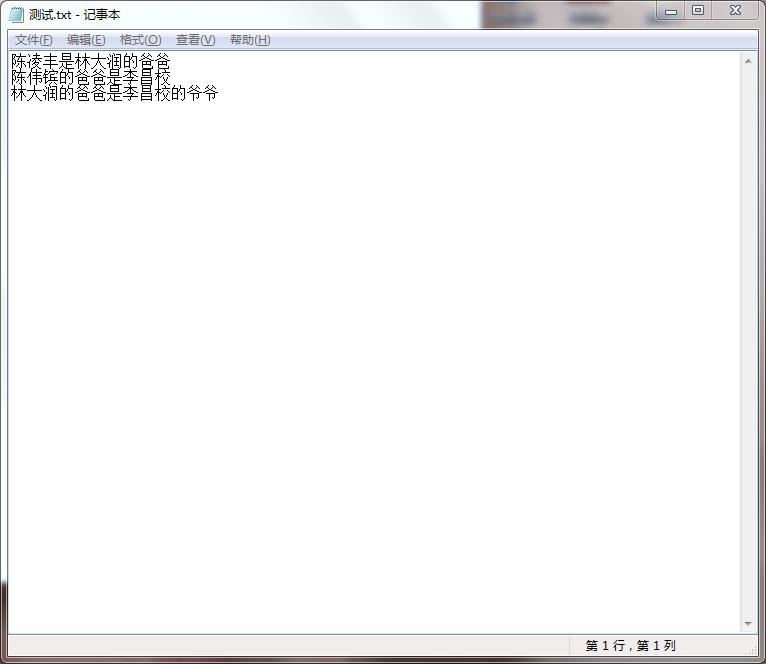
controlNotFamily – 控制非亲属树

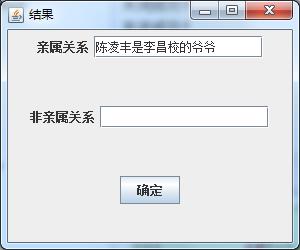
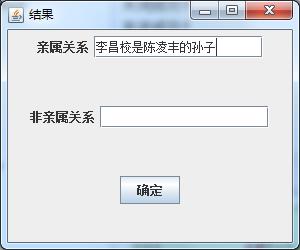
controlbro – 控制兄弟链

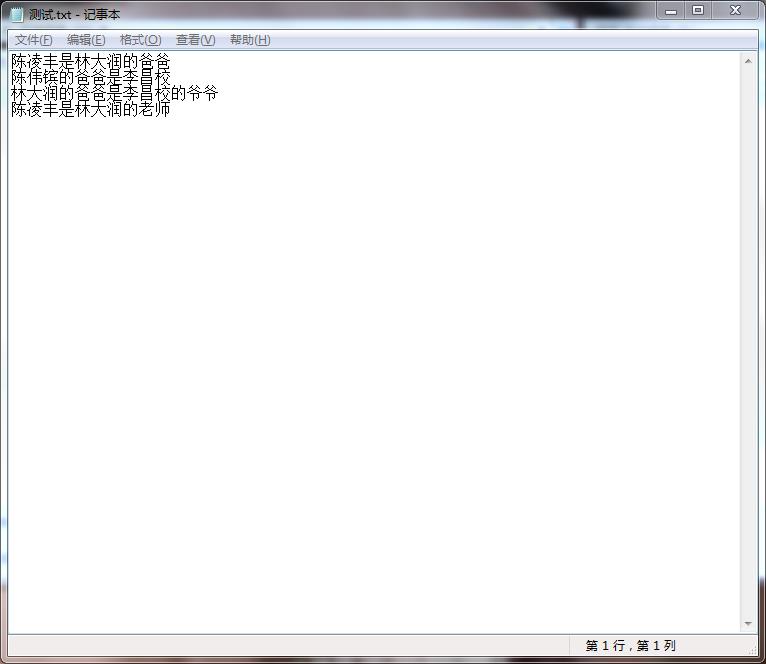
controlInquiry -控制查询业务

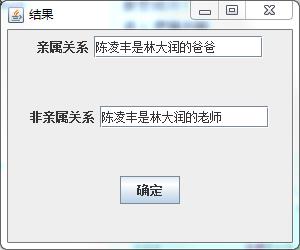
getresult – 将结果判断出关系并显示在界面

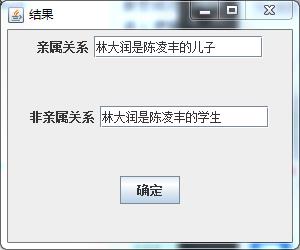
运行结果截图：

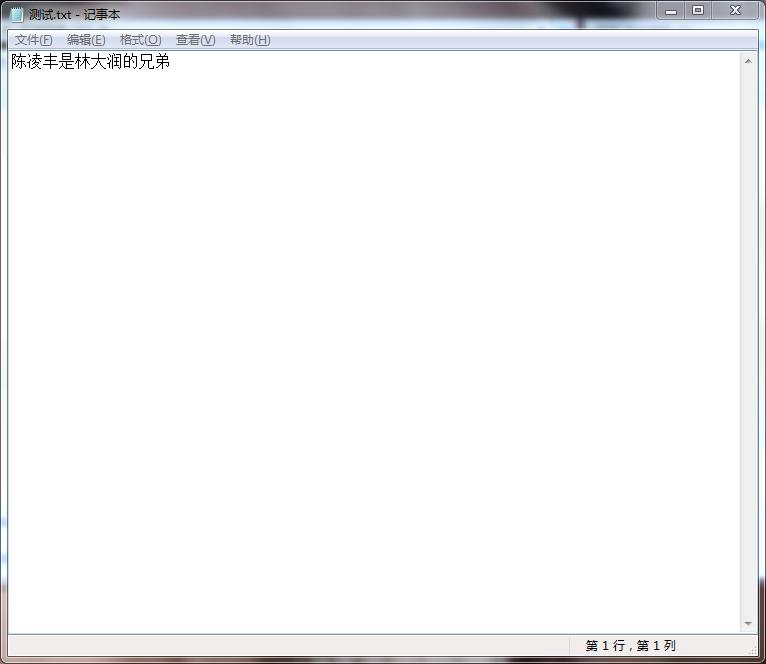


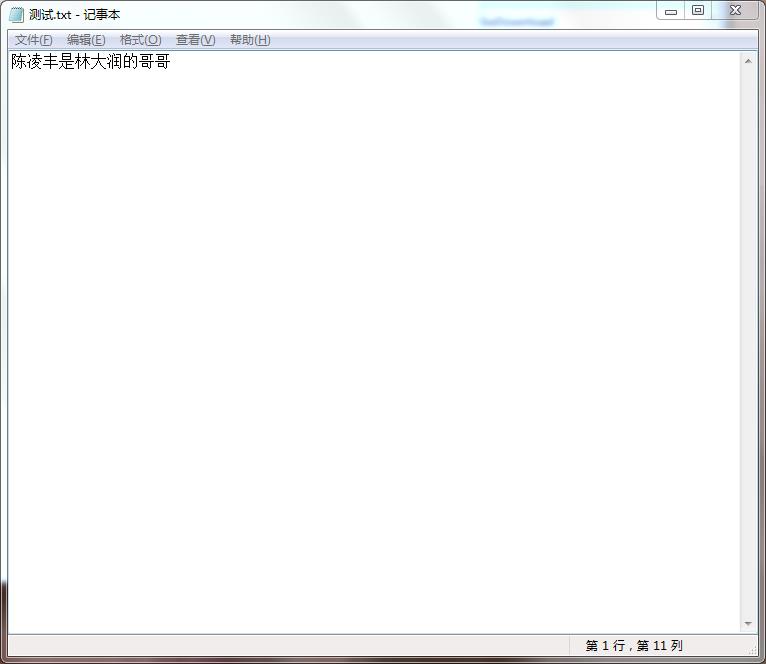


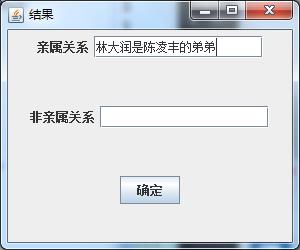










难点：树的合并

原因：因为存在”未知者”的存在 所以存在两棵树上相同位置有不同的人： 一个为名字/一个为“未知者” 所以要合并 但是每次判断合并行不通 因为合并“儿子”要转移 而合并儿子又要判断是否是“未知者”…

策略：找到两棵树的最远祖先-判断族谱关系（辈分）然后直接合并（不管是否为未知者 有未知者则往右排）

然后更新：先序遍历每一个点 判断是都为未知者 若是进行兄弟合并（以第一个出现的人物代替未知者位置）

心得：做前最主要的分包/主要用的数据结构确定好开始动手做。卡壳后多尝试别的思路解决（例如合并树 每个点分别合并不可行 就先合并再更新）

参考资料：

二叉树：https://blog.csdn.net/fengrunche/article/details/52305748