

DSAA Report

一、人员分工

吕炳杰 11712609: 负责 GUI 界面, 与算法的交互逻辑

刘拯涛 11711020: 负责实现诗词的录入、搜索算法

王昊文 11711018: 负责模拟学习及复习算法

工作量均为三分之一, 合作过程很愉快。

二、设计模式

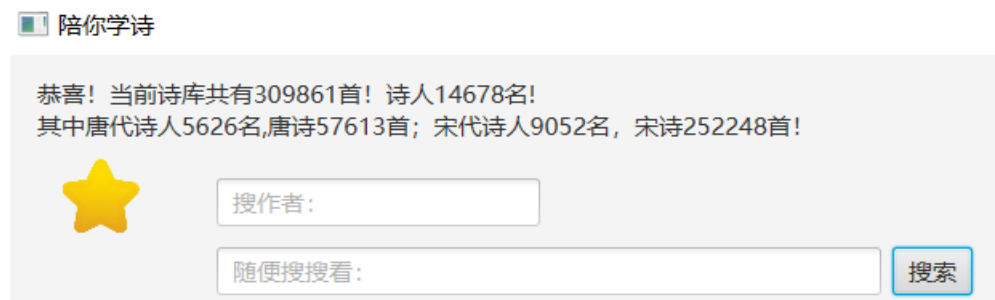
项目采用, MVC 设计模式: 模块、视图、控制器, 以实现简洁的用户界面和友好的操作步骤, 力求有良好的使用体验。

三、步骤演示

搜索界面




点击搜索按钮, 即会在控制台输出当前诗库的所有诗以及统计结果。



搜作者: 实现按作者搜索。

随便搜搜看: 该搜索栏集成了作者搜索, 题目搜索, 标签搜索, 内容搜索, 同时我们实现了任意多关键字搜索, 中间用 and 隔开。实现了 and 搜索之后, 我们不满足于此, 也实现了 or 搜索。我们搜索一个 txt 文件, 如 poem75,

就会打印出来这个文件的内容，再加几个汉字比如我太难了，然后搜索，就实现了为文件批量一键设置 tag 功能。



搜作者:

王之道and和吕

搜索

<王之道, 和吕叔恭題和州香林湯, 新詩等絳繡, 文采照座右。null>



搜作者:

poem75

搜索

1 江南 (江南可采莲) 汉乐府

2 长歌行 (青青园中葵) 汉乐府

3 敕勒歌 (敕勒川) 北朝民歌

4 咏鹅 (鹅鹅鹅) 骆宾王

5 风 (解落三秋叶) 李峤

6 咏柳 (碧玉妆成一树高) 贺知章

7 回乡偶书 (少小离家老大回) 贺知章

8 凉州词 (黄河远上白云间) 王之涣

9 登鹳雀楼 (白日依山尽) 王之涣


10 春晓 (春眠不觉晓) 孟浩然

11 凉州词 (葡萄美酒夜光杯) 王翰

12 出塞 (秦时明月汉时关) 王昌龄

13 芙蓉楼送辛渐 (寒雨连江夜入吴) 王昌龄

恭喜! 当前诗库共有309861首! 诗人14678名!
其中唐代诗人5626名,唐诗57613首; 宋代诗人9052名, 宋诗252248首!



搜作者:

poem75我太难了

搜索

1 江南 (江南可采莲) 汉乐府 我太难了

2 长歌行 (青青园中葵) 汉乐府 我太难了

3 敕勒歌 (敕勒川) 北朝民歌 我太难了

4 咏鹅 (鹅鹅鹅) 骆宾王 我太难了

5 风 (解落三秋叶) 李峤 我太难了

6 咏柳 (碧玉妆成一树高) 贺知章 我太难了

7 回乡偶书 (少小离家老大回) 贺知章 我太难了

8 凉州词 (黄河远上白云间) 王之涣 我太难了

9 登鹳雀楼 (白日依山尽) 王之涣 我太难了

10 春晓 (春眠不觉晓) 孟浩然 我太难了

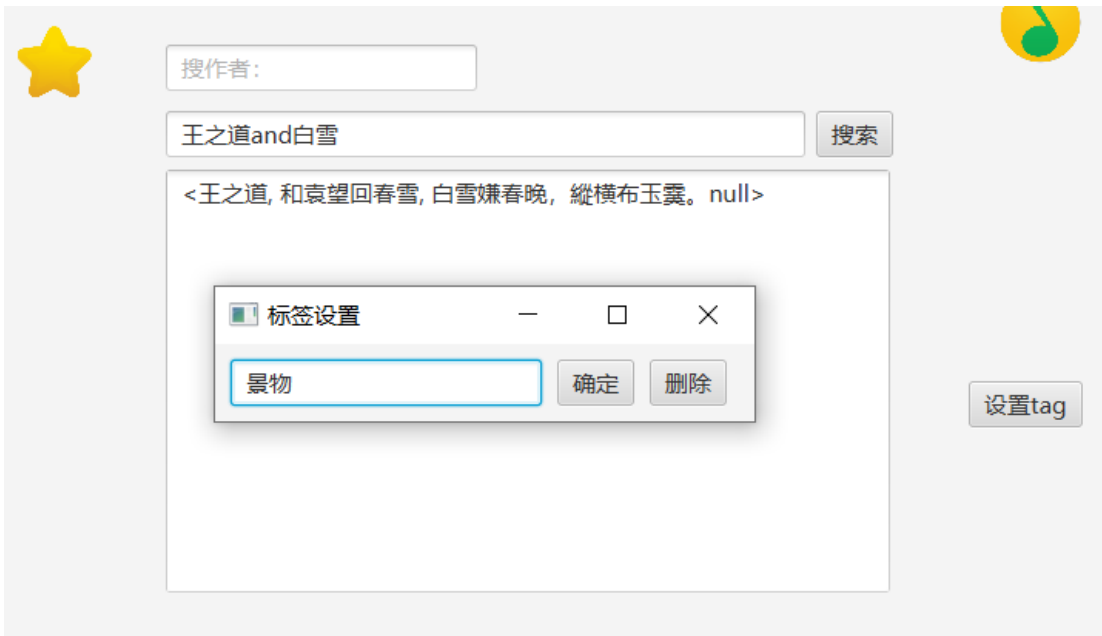
11 凉州词 (葡萄美酒夜光杯) 王翰 我太难了

12 出塞 (秦时明月汉时关) 王昌龄 我太难了

13 芙蓉楼送辛渐 (寒雨连江夜入吴) 王昌龄 我太难了

设置tag

设置 tag: 可以为单首诗增删标签。



学习界面



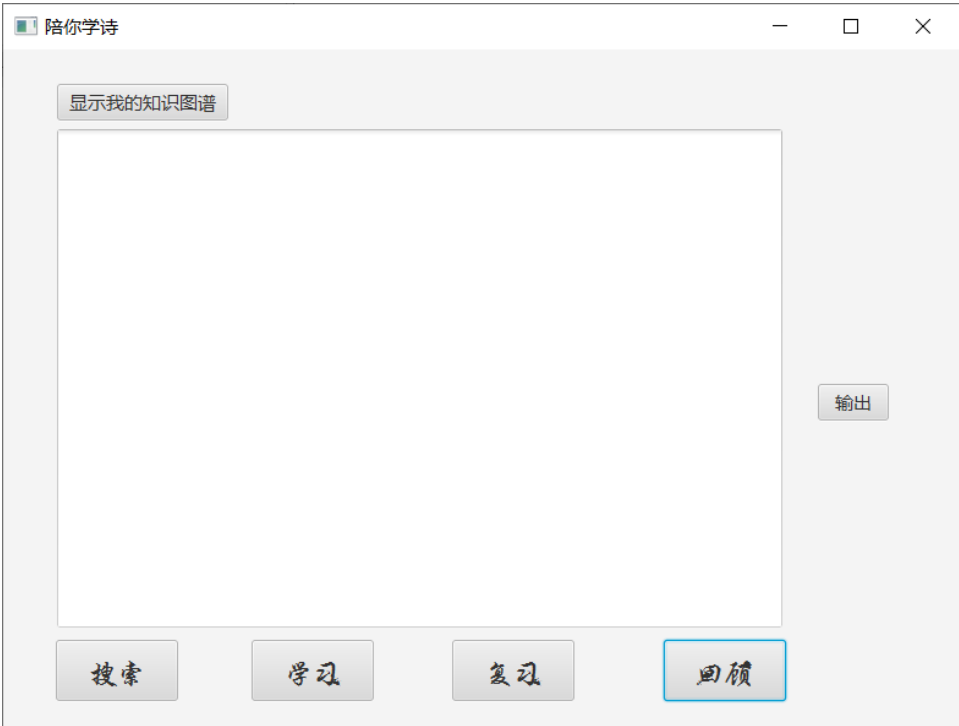
可以自行选择想要学习多少首诗，点击充实自己开始学习，通过点击“完全忘记”“丢三落四”“渐入佳境”“滚瓜烂熟”来为每首诗设置熟练度，点击“反馈当天”可以显示当天的学习情况，点击“反馈所有”可以显示从第一天到现在的学习情况，左上角的方框可以实现当天的学习天数。

复习界面



可以自行选择每天想要复习多少首诗以及重新设置熟练度，点击“给我反馈”即可显示最新的学习情况。

回顾界面



点击“显示我的知识图谱”即可展示当前的所学情况，点击“输出”即可将自己的学习情况变成 txt 文件进行输出。

三、算法实现

1. json 录入和解析：

```
import com.google.gson.stream.JsonReader;

public class jsonjiexi {

    //输入文件的路径，朝代和一个空列表，最后输出一个读取完信息的列表
    public static ArrayList<yuanSu> newjiexi(String filepath, ArrayList<yuanSu> list,String dynasty) {
        InputStream is = Thread.currentThread().getContextClassLoader().getResourceAsStream(filepath);
        InputStreamReader in = new InputStreamReader(is);
        JsonReader a = new JsonReader(in); //将json导入存储器中
        list = readMessageArray(a, list, dynasty); //调用readMessageArray读取每首诗的节点
        return list;
    }

    //输入存储器，和一个空表，朝代。
    private static ArrayList<yuanSu> readMessageArray(JsonReader a, ArrayList<yuanSu> list,String dynasty) {
        try {
            a.beginArray();
            while (a.hasNext()) { //读取每一个段落里面的key, Name, Paragraph, Title, Author.
                list.add(readMessage(a, dynasty)); //获取每一个key所对应的value，并存储进列表里面
            }
            a.endArray();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return list;
    }

    //读取jsonReader里的key，并根据key获取所对应的value，最后把一首诗的所有value存储进一个节点内
    private static yuanSu readMessage(JsonReader a,String dynasty) {
        yuanSu poem = new yuanSu();
        try {
            a.beginObject();
            while (a.hasNext()) {
                String atyuanSu = a.nextName();
                //调用json包里的nextName方法，获取下一个key的名字，然后匹配名字并存入这个节点的相关变量中
                if ("author".equals(atyuanSu)) {
                    poem.setAuthor(a.nextString());
                } else if ("paragraphs".equals(atyuanSu) && a.peek() != JsonToken.NULL) {
                    poem.setParagraphs(readparagraph(a)); //因为paragraph所对应的value是一个数组，所以构建读取paragraph的方法，再导入节点中。
                } else if ("tags".equals(atyuanSu)) {
                    poem.setTags(readparagraph(a));
                } else if ("title".equals(atyuanSu)) {
                    poem.setTitle(a.nextString());
                } else if ("id".equals(atyuanSu)) {
                    poem.setId(a.nextString());
                } else {
                    a.skipValue();
                }
                poem.setDynasty(dynasty);
            }
            a.endObject();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return poem;
    }

    //如果key对应的value是一个数组，那么就先将value读取入数组内，最后再将数组导入元素中
    private static ArrayList<String> readparagraph(JsonReader a) {
        // TODO Auto-generated method stub
        ArrayList<String> paragraph = new ArrayList<String>(); //构建一个空的array用于存储value
        try {
            a.beginArray();
            while (a.hasNext()) {

                paragraph.add(a.nextString());

            }
            a.endArray();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return paragraph;
    }
}
```

基本思路：将 json 文件存储进 jsonReader 中，然后使用 json 的 nextName 方法读取 json 的 key，并根据读取的 key 读取相对应的 value，最后把一首诗的所有信息存储进一个节点内，再把节点添加进列表内

2. 显示已读入 xx 首诗，xx 名作者，其中唐代诗人 xx 名，诗 xx 首，宋代诗人 xx 名，诗 xx 首等

```
for (int i = 2; i < filepath.length; i++) {
    if (filepath[i].indexOf("song") != -1) {
        list = a.newjixi(filepath[i], list, "song");//在解析json的同时，根据文件名，给节点添加朝代名字并且计数
    } else if (filepath[i].indexOf("tang") != -1) {
        list = a.newjixi(filepath[i], list, "tang");
    }
}
int count1 = 0;
mm = new LinkedList<yuanSu>();
for (yuanSu k : list) {
    k.setCount(count1);
    mm.add(k);
    count1++;
}
```

基本思路：

在解析 json 的时候，根据输入的文件名字，给节点添加朝代名字这个变量，并且计数，最后将 ArrayList 的所有节点存储进 linkedlist 中。

3. 搜索查询

基本搜索：

```
yuanSu yuanSuJ;
for (Iterator<yuanSu> kk = mm.iterator(); kk.hasNext(); ) {
    yuanSuJ = kk.next();
    if (yuanSuJ.getAuthor().indexOf(keyName) != -1 || yuanSuJ.getId().indexOf(keyName) != -1 || yuanSuJ.getTitle().indexOf(keyName) != -1) {
        result += "<" + yuanSuJ.getAuthor() + ", " + yuanSuJ.getTitle() + ", " + yuanSuJ.getParagraph() + yuanSuJ.getTags() + ">" +
            counter = yuanSuJ.getCount();
    } else if (yuanSuJ.getParagraphs().size() != 0 && yuanSuJ.getParagraph().indexOf(keyName) != -1) {
        result += "<" + yuanSuJ.getAuthor() + ", " + yuanSuJ.getTitle() + ", " + yuanSuJ.getParagraph() + yuanSuJ.getTags() + ">" +
            counter = yuanSuJ.getCount();
    } else if (yuanSuJ.getTag() != null && yuanSuJ.getTag().indexOf(keyName) != -1) {
        result += "<" + yuanSuJ.getAuthor() + ", " + yuanSuJ.getTitle() + ", " + yuanSuJ.getParagraph() + yuanSuJ.getTags() + ">" +
            counter = yuanSuJ.getCount();
    }
}
```

基本思路：

搜索查询是用迭代器遍历和将要搜索的 string 和每一个节点的变量内容相匹配，看是否包含。如果包含就输出。

多关键词搜索：

```
else {
    splitInput splitInput = new splitInput();
    if (keyName.indexOf("and") != -1) {
        String[] andGet = splitInput.SplitInputAnd(mm, keyName);//and搜索
        result = andGet[0];
        counter = Integer.parseInt(andGet[1]);

    } else if (keyName.indexOf("or") != -1) {
        result = splitInput.SplitInputOr(mm, keyName);//or搜索
        System.out.println(result);
    }
}
```

基本思路：

如果是 or 搜索，那么就先进入基本搜索得出含有该关键词的链表，最后对这些链表进行对比，最后输出一个无重复的含有所有节点的链表。

如果是 and 搜索，就是对第一个关键词进行基本搜索，得出含这个关键词的链表，再对这个链表进行第二个关键词的基本搜索。最后得出一个交集链表。

增加或删除 tag:

```
textField.setPromptText("请输入你想设置的标签:"); //得到tag
textField.setFocusTraversable(false);
Button delete = new Button("删除");
Button button = new Button("确定");
button.setOnAction(event -> {
    String tag = textField.getText();
    ArrayList<String> tagLink = new ArrayList<String>();
    if (mm.get(counter).getTags() == null) { //counter是每一次搜索后最后一首诗的位置
        tagLink.add(tag);
    } else {
        tagLink = mm.get(counter).getTags(); //往确定位置的节点处添加tag
        tagLink.add(tag);
        mm.get(counter).setTags(tagLink);
        String tagText = "[";
        for (int i = 0; i < mm.get(counter).getTags().size(); i++) {
            tagText += mm.get(counter).getTags().get(i) + " ";
        }
        tagText += "];";
        textArea.setText(tagText);
    });

delete.setOnAction(event -> {
    String tag = textField.getText();
    ArrayList<String> tagLink = new ArrayList<String>();
    if (mm.get(counter).getTags() == null) {
        tagLink.remove(tag);
    } else {
        tagLink = mm.get(counter).getTags();
        tagLink.remove(tag);
    }
    mm.get(counter).setTags(tagLink);
    String tagText = "[";
    for (int i = 0; i < mm.get(counter).getTags().size(); i++) {
        tagText += mm.get(counter).getTags().get(i) + " ";
    }
    tagText += "];";
});
```

基本思路: 我们在录入节点的同时输入节点的位置。在 gui 上返回所显示的节点的位置和用户想增加或删除的 tag, 最后向所对应节点中输入 tag。

4. 显示学习模拟全部诗列表 (标题和作者), 不少于 100 首

通过确定读取包含全部诗的链表确定长度, 每次从中随机生成一个序号, 在推荐学习链表 studyList 中加入诗词库中该序号对应的古诗, 最后依次打印出来。

5. 学习，推荐选择新学习诗

```
7 public class simulationStudy {
8     //poem中存所有古诗
9     private static int getRandomWord(LinkedList<Node> poem) {
10         int indexBound;
11         indexBound = poem.size(); //确定长度
12         Random random = new Random();
13         int index = random.nextInt(indexBound); //随机生成序号
14         return index;
15     }
16     //显示学习模拟全部诗列表
17     public static void simulationStudy(LinkedList<Node> temp, LinkedList<Node> poem) {
18         LinkedList<yuanSu> library = new LinkedList<yuanSu>(); //设定存储链表
19         int i = 0;
20         while (temp.size() != 0 && i < 200) {
21             int x = getRandomWord(temp); //从诗词库中随机生成一个序号
22             library.add(temp.get(x)); //在library链表中添加序号对应的古诗
23             temp.remove(temp.get(x)); //从temp中删除对应的古诗
24             i++;
25         }
26         int j = 1;
27         //将要学习模拟的诗依次打印出来
28         for (yuanSu a : library) {
29             System.out.println(j + "." + a.getTitle() + "\t" + a.getAuthor());
30             j++;
31         }
32         //返回一个链表值
33         return library;
34     }
35 }

//推薦新學習詩詞
36 public static LinkedList<yuanSu> recommend2(LinkedList<yuanSu> library, LinkedList<yuanSu> newStudy, int num) {
37     //从模拟学习的库中推荐要新学习的古诗，并将其放入newStudy链表中
38     for (int i = 0; i < num; i++) {
39         if (i <= library.size()) {
40             int x = getRandomWord(library);
41             newStudy.add(library.get(x));
42             library.remove(x);
43         } else {
44             break;
45         }
46     }
47     int j = 1;
48     //将要学习的诗词打印出来
49     for (yuanSu a : newStudy) {
50         System.out.println(j + "." + a.getTitle() + "\t" + a.getAuthor());
51         j++;
52     }
53     return newStudy;
54 }
55 //测试新学习诗
```

返回 newStudy 链表的值

6. 背诵新学习的诗

```
7 //背诵新詩詞
8 public static LinkedList<yuanSu> reciteNewPoem(LinkedList<yuanSu> newStudy, LinkedList<yuanSu> studyList) {
9     Scanner input = new Scanner(System.in);
10     int length = newStudy.size();
11     //遍历要新学习的诗词
12     for (int i = 0; i < length; i++) {
13         yuanSu a = newStudy.getFirst();
14         //打印出当前要背的诗
15         System.out.println("<" + a.getAuthor() + ", " + a.getTitle() + ", " + a.getParagraph()
16             + a.getTags() + ">");
17         //设定背完诗后自己感觉的熟练度
18         String change = input.next();
19         a.setDegree(change);
20         //将改变完熟练度的古诗放入studyList链表中
21         studyList.add(a);
22         //从newStudy中移除该首诗。
23         newStudy.remove(a);
24     }
25     return studyList;
26 }
27 }
```

返回链表 studyList 的值

7.显示要复习的诗以及熟练度

//顯示要複習的詩以及熟練程度

```
public static LinkedList<yuanSu> reviewRecommend(LinkedList<yuanSu> studyList, int num) {
    //设置四个新的链表用来存储四种熟练度对应的诗词
    LinkedList<yuanSu> list1 = new LinkedList<yuanSu>();
    LinkedList<yuanSu> list2 = new LinkedList<yuanSu>();
    LinkedList<yuanSu> list3 = new LinkedList<yuanSu>();
    LinkedList<yuanSu> list4 = new LinkedList<yuanSu>();
    LinkedList<yuanSu> list = new LinkedList<yuanSu>();
    for (yuanSu a : studyList) {
        switch (a.getDegree()) {
            case "完全忘記":
                list1.add(a);
                break;
            case "丟三落四":
                list2.add(a);
                break;
            case "漸入佳境":
                list3.add(a);
                break;
            case "滾瓜爛熟":
                list4.add(a);
                break;
        }
    }
    //我们设定了一个背诵古诗的规则，每个熟练度有一定配额，并将要复习的诗都放入list链表中
    int count = 1;
    int second = num/3;
    int third = num /4;
    int first = num-second-third;
    int i = 0;
    //从每个熟练度打印额定数目的古诗
    for (yuanSu a : list1) {
        if (i < first) {
            System.out.println(count+" "+a.getTitle() + '\t' + a.getAuthor() + '\t' + a.getDegree());
            studyList.remove(a);
            i++;
            count++;
            list.add(a);
        }else {
            break;
        }
    }
    i = 0;
    for (yuanSu a : list2) {
        if (i < second) {
            System.out.println(count+" "+a.getTitle() + '\t' + a.getAuthor() + '\t' + a.getDegree());
            studyList.remove(a);
            i++;
            count++;
            list.add(a);
        }else {
            break;
        }
    }
    i = 0;
    for (yuanSu a : list3) {
        if (i < third) {
            System.out.println(count+" "+a.getTitle() + '\t' + a.getAuthor() + '\t' + a.getDegree());
            studyList.remove(a);
            i++;
            count++;
            list.add(a);
        }else {
            break;
        }
    }
    return list;
}
```

8. 背诵要复习的单词

```
//背诵要复习的单词
public static LinkedList<yuanSu> review(LinkedList<yuanSu> list,LinkedList<yuanSu> studyList) {
    Scanner input = new Scanner(System.in);
    //遍历要复习的list链表，并更改记忆后该首诗的熟练度
    for (yuanSu a : list) {
        System.out.println("<" + a.getAuthor() + "," + a.getTags() + "," + a.getParagraph()
            + a.getTitle() + "," + a.getId() + ">");
        String change = input.next();
        a.setDegree(change);
        studyList.add(a);
    }

    return studyList;
}
```

将复习背诵过的古诗再次放入 studyList 链表中。

9. 反馈学习后新的熟练度

```
//反馈学习后新的熟练度
public static void feedBack(LinkedList<yuanSu> studyList){
    LinkedList<yuanSu> list1 = new LinkedList<yuanSu>();
    LinkedList<yuanSu> list2 = new LinkedList<yuanSu>();
    LinkedList<yuanSu> list3 = new LinkedList<yuanSu>();
    LinkedList<yuanSu> list4 = new LinkedList<yuanSu>();

    LinkedList<yuanSu> list = new LinkedList<yuanSu>();
    for (yuanSu a : studyList) {
        switch (a.getDegree()) {
            case "完全忘记":
                list1.add(a);
                break;
            case "丢三落四":
                list2.add(a);
                break;
            case "渐入佳境":
                list3.add(a);
                break;
            case "滚瓜烂熟":
                list4.add(a);
                break;
        }
    }
    int count = 0;

    for (yuanSu a : list1) {
        count++;
        System.out.println(count+"."+a.getTitle() + '\t' + a.getAuthor() + '\t' + a.getDegree());
    }
    for (yuanSu a : list2) {
        count++;
        System.out.println(count+"."+a.getTitle() + '\t' + a.getAuthor() + '\t' + a.getDegree());
    }
    for (yuanSu a : list3) {
        count++;
        System.out.println(count+"."+a.getTitle() + '\t' + a.getAuthor() + '\t' + a.getDegree());
    }
    for (yuanSu a : list4) {
        count++;
        System.out.println(count+"."+a.getTitle() + '\t' + a.getAuthor() + '\t' + a.getDegree());
    }
}
```

从学习过的 studyList 链表中的古诗按照熟练度分类，并依次打印出来。

四、改进空间

亮点：秒搜、界面简洁操作友好，增加了要求外的收藏夹功能

改进：播放音乐功能、窗口自适应大小、加括号进来实现 and 和 or 并列搜索