

**课 程 实 验 报 告**

**课程名称：Java语言程序设计**

**实验名称：基于内存的搜索引擎设计和实现**

**院 系 ：计算机科学与技术**

**专业班级 ： 校交1902**

**学 号 ： U201912633**

**姓 名 ： 张睿**

**指导教师 ： 辜希武**

**2021 年 5 月 13 日**

目录

[一、需求分析 - 2 -](#_Toc71903898)

[1.1 题目要求 - 2 -](#_Toc71903899)

[二、系统设计 - 4 -](#_Toc71903900)

[2.1 概要设计 - 4 -](#_Toc71903901)

[2.2 详细设计 - 5 -](#_Toc71903902)

[2.2.1 parse模块 - 5 -](#_Toc71903903)

[2.2.2 index模块 - 7 -](#_Toc71903904)

[2.2.3 query模块 - 12 -](#_Toc71903905)

[2.2.4 utils模块和run模块 - 17 -](#_Toc71903906)

[三、软件开发 - 18 -](#_Toc71903907)

[四、软件测试 - 19 -](#_Toc71903908)

[2.1 测试包测试结果 - 19 -](#_Toc71903909)

[2.2 额外功能测试 - 19 -](#_Toc71903910)

[五、特点与不足 - 24 -](#_Toc71903911)

[5.1技术特点 - 24 -](#_Toc71903912)

[5.2不足和改进的建议 - 24 -](#_Toc71903913)

[六、过程和体会 - 24 -](#_Toc71903914)

[1.遇到的主要问题和解决方法 - 24 -](#_Toc71903915)

[2.课程设计的体会 - 25 -](#_Toc71903916)

[七、说明 - 25 -](#_Toc71903917)

# 一、需求分析

## 1.1 题目要求

实现一个基于内存的英文全文检索搜索引擎，需要完成以下功能：

**功能1：**将指定目录下的一批.txt格式的文本文件扫描并在内存里建立倒排索引，这里面包含必须的子功能包括：

（1）读取文本文件的内容；

（2）将内容切分成一个个的单词；

（3）过滤掉其中一些不需要的单词,例如数字、停用词（the, is and这样的单词）、过短或过长的单词（例如长度小于3或长度大于20的单词）；

（4）利用Java的集合类在内存里建立过滤后剩下单词的倒排索引；

（5）内存里建立好的索引对象可以序列化到文件，同时可以从文件里反序列化成内存里的索引对象；

（6）可以在控制台输出索引的内容。

**功能2：**基于构建好的索引，实现单个搜索关键词的全文检索，包含的子功能包括：

（1）根据搜索关键词得到命中的结果集合；

（2）可以计算每个命中的文档的得分，并根据文档得分对结果集排序；

（3）在控制台显示命中的文档的详细信息，如文档的路径、文档内容、命中的关键词信息（如在文档里出现次数）、文档得分；

**功能3：**基于构建好的索引，实现二个搜索关键词的全文检索。包含的子功能包括：

（1）支持这二个关键词的与或查询。与关系必须返回同时包含这二个单词的文档集合，或关系返回包含这二个单词中的任何一个的文档集合；

（2）可以计算每个命中的文档的得分，并根据文档得分对结果集排序；

（3）在控制台显示命中的文档的详细信息，如文档的路径、文档内容、命中的关键词信息（如在文档里出现次数）、文档得分；

**功能4：**基于构建好的索引，实现包含二个单词的短语检索，即这二个单词必须在作为短语文档里出现，它们的位置必须是相邻的。**这个功能为进阶功能**。

除了以上功能上的要求外，其他要求包括：

（1）针对搜索引擎的倒排索引结构，已经定义好了创建索引和全文检索所需要的抽象类和接口。**学生必须继承这些预定义的抽象类和和实现预定义接口来完成实验的功能**，**不能修改抽象类和接口里规定好的数据成员、抽象方法；也不能在预定义抽象类和接口里添加自己新的数据成员和方法**。但是实现自己的子类和接口实现类则不作任何限定。

（2）自己实现的抽象类子类和接口实现类里的关键代码必须加上注释，其中每个类、每个类里的公有方法要加上Javadoc注释，并自动生成Java API文档作为实验报告附件提交。

（3）使用统一的测试文档集合、统一的搜索测试案例对代码进行功能测试，构建好的索引和基于统一的搜索测试案例的检索结果最后输出到文本文件里作为实验报告附件提交。

（4）本实验只需要基于控制台实现，实验报告里需要提供运行时控制台输出截屏。

**关于搜索引擎的倒排索引结构、相关的抽象类、接口定义、还有相关已经实现好的工具类会在单独的PPT文档里详细说明。同时也为学生提供了预定义抽象类和接口的Java API文档和UML模型图。**

# 二、系统设计

## 2.1 概要设计

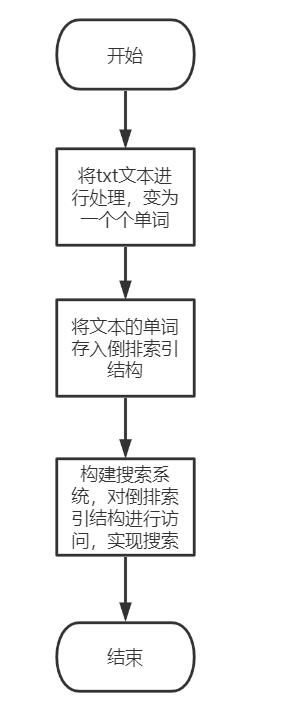
分析需求后我们可以发现，要解决“搜索”的问题，我们可以从三个方面考虑：

1. 如何获得搜索的文本资料

2. 获得文本资料后如何安排构建存储结构

3. 构建好文本资料的存储结构后如何对其进行访问实现搜索

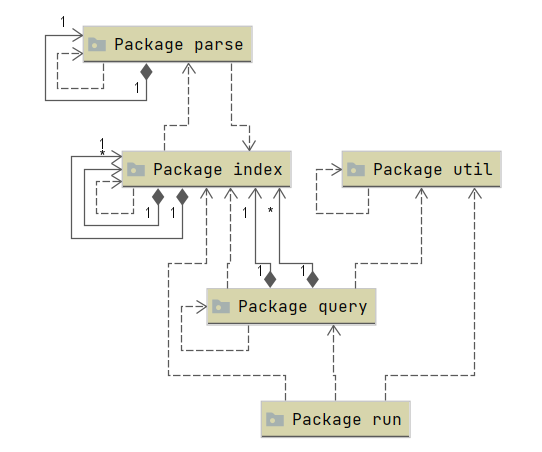
从这三个方面考虑，我们可以做出搜索程序的大致流程图，流程图如下：



分析需求后，根据功能可大致分解出所需模块，模块如下表所示：

|  |  |
| --- | --- |
| 模块名称 | 模块功能 |
| Parse | 处理文本 |
| Index | 构建倒排索引 |
| Query | 构建搜索系统 |
| Util | 功能包 |
| Run | 简单测试模块 |

各个模块之间的依赖关系如下图所示



## 2.2 详细设计

接下来将从每个模块的功能、流程（图）和局部数据结构等方面叙述各个模块的具体实现。

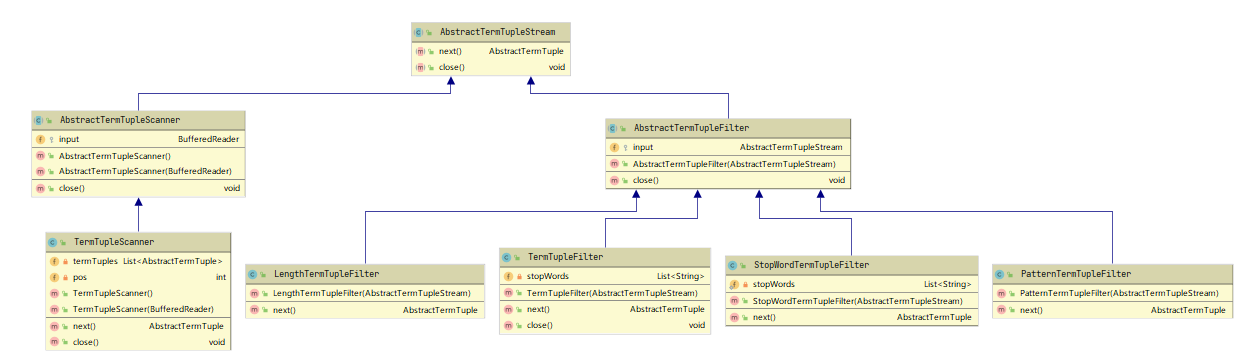
2.2.1 parse模块

1. 模块总览

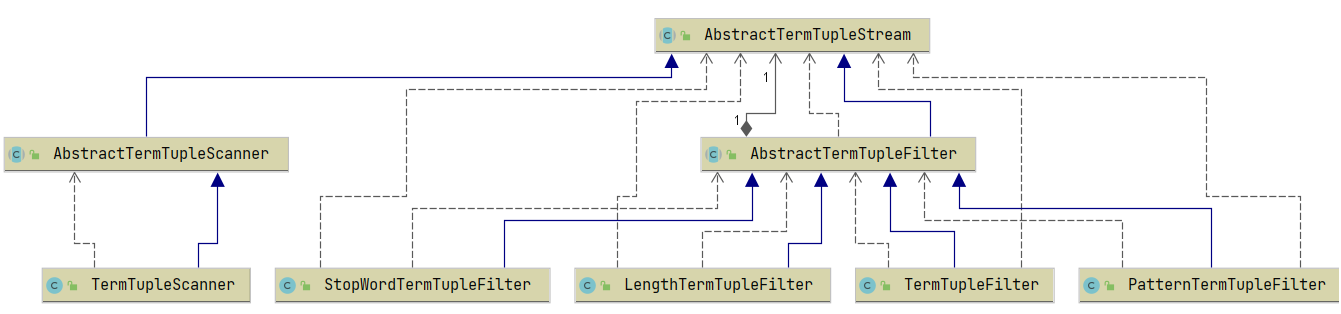
parse模块需要达成以下目的：

* 1. 对每一篇文章完成分词操作
  2. 对分好的单词进行过滤

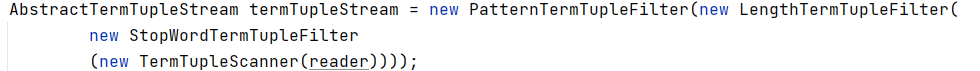
为达成目的设计出以下类，并构建其继承关系，如下图所示。



各个类的依赖关系如下图所示：



这个模块需要完成装饰者模式的构建，所以所有具体实现类都继承AbstractTermTupleStream，里面定义了next和close两个方法，方便我们在获取单词时直接进行过滤，得到三元组。这里的三元组为引进的数据结构，用来方便文档的内部存储，具体介绍参看index模块。所以，基于流的方法，我们可以很方便地通过以下语句





获取流对象，再利用next方法获取流数据。

2. 具体类的实现

在这几个类的实现方法中，大部分非常简单，这里只阐述scanner的实现，filter的实现即在scanner的实现上进行判断即可，因此不在此赘述。

Scanner需达成以下几个目的（也即处理流程）：

* 1. 读取文章
  2. 对文章进行分词
  3. 将分词后的单词构建成三元组
  4. 返回三元组

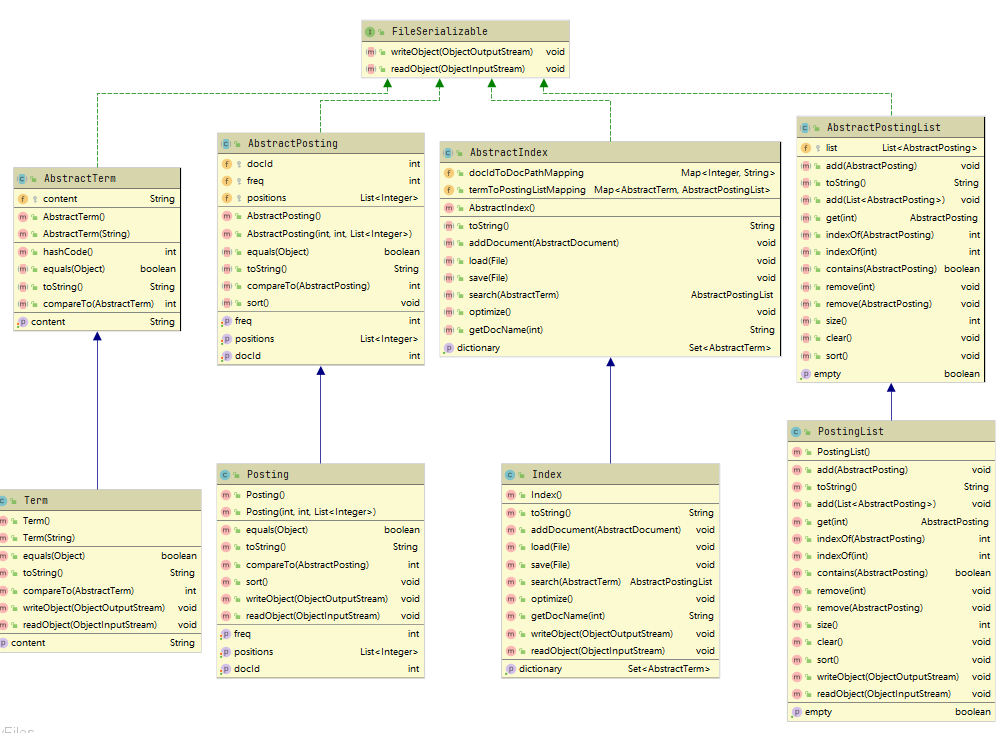
2.2.2 index模块

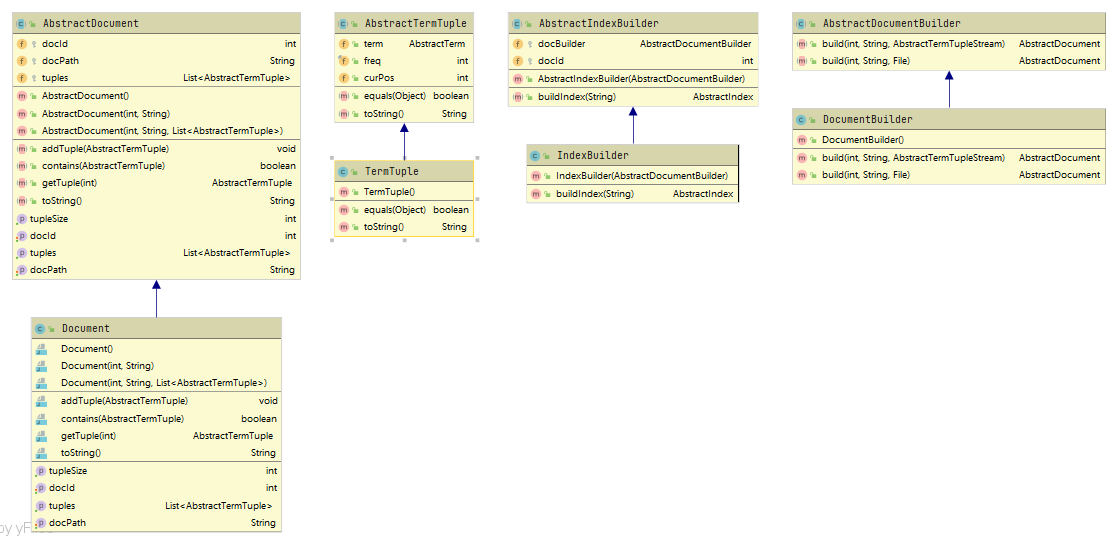
1. 模块总览

Index模块需要达到的目的有：

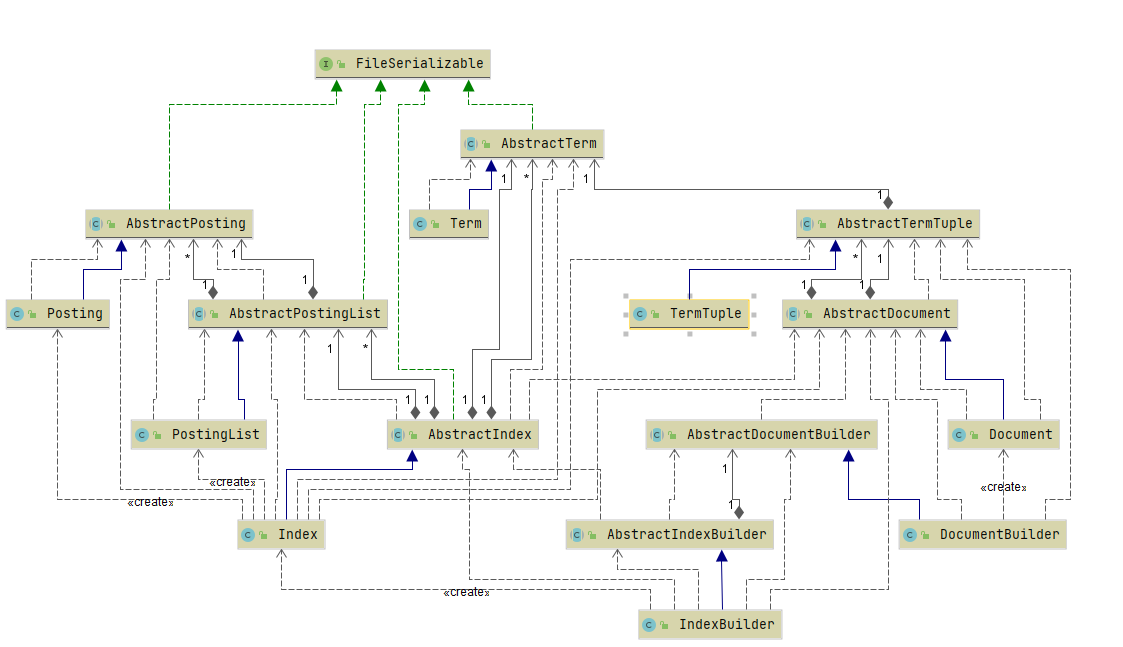
1. 构建文章对象
2. 通过文章对象构建倒排索引

为了达成目的，设计出以下类及其继承关系





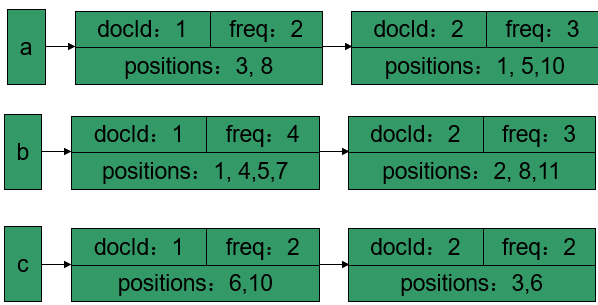
各个类的依赖关系如下：



这样设计的原因也是很明了的：

1. 首先定义数据结构

因为我们的原始数据为文本文件，任务以单词构建索引，为方便搜索，所以采用倒排索引的方法，而倒排索引的结构如下图所示：



每个单词对应一个postingList，每个postingList由若干个posting组成。因此Term、Posting、PostingList、Index这四个类就规定了倒排索引所需要的数据结构。

1. 其次定义辅助数据结构帮助核心数据结构的构建。

可以发现，每个posting都有几个组成要素：单词，单词对应的频率，单词出现的位置，文本id，为此我们可以设计出termTuple类辅助建立postingList，其中的数据成员包括单词、单词频率、单词出现的位置。

其次，由于有多个文档，而处理方式是逐个文档进行解析，而且由于搜索到后需要文章的地址，需要将其保存，所以我们再设计一个中间数据结构document来帮助倒排索引的构建和搜索的实现.

1. 最后实现数据的装入。

最后根据所建立的数据结构，将数据装入倒排索引结构中，为此设计出DocumentBuilder和IndexBuilder。

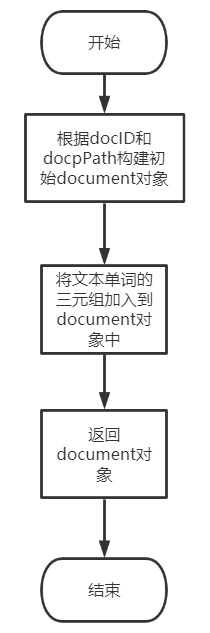
2.具体类的实现

|  |  |  |
| --- | --- | --- |
| 类名 | 功能描述 | 数据成员 |
| Term | 实例为一个单词term | Term |
| Posting | 实例为倒排索引里的一个Posting | docId、freq、positions，分别代表单词出现的文档Id、出现频率、出现的位置列表，其中位置列表采用Java的集合类型List<Integer>存贮单词出现的多个位置 |
| PostingList | 实例为倒排索引里一个单词对应的PostingList | 一个List<AbstractPosting>类型的数据成员存放这个PostingList包含的多个Posting |
| Index | 实例为内存中的整个倒排索引结构 | docIdToDocPathMapping：类型为Map<Integer, String> ，保存了文档Id和文档绝对路径之间的映射关系  termToPostingListMapping：类型为Map<AbstractTerm, AbstractPostingList> ，保存了每个单词与其对应的PostingList的映射关系。 |
| TermTuple | 实例为和单词term相关的三元组 | AbstractTerm term：代表当前解析得到的一个term  final int freq = 1：因为解析得到了一个term，该term出现了一次，因此其频率为1  int curPos：该term的位置（注意位置序号是以term为单位不是以字符为单位） |
| Document | 实例为：解析完一个文档后文档在内存中的表示。因为当解析完文档后，文档需要一种中间类型的数据结构表示，以方便后面倒排索引的建立。 | int docId：文档Id  String docPath：文档绝对路径  List<AbstractTermTuple> tuples：文档解析完后得到的所有term的三元组 |
| DocumentBuilder | 构建document | 无 |
| IndexBuilder | 构建倒排索引 | 无 |

接下来将挑选叙述关键类的相关内容。

1. documentBuilder

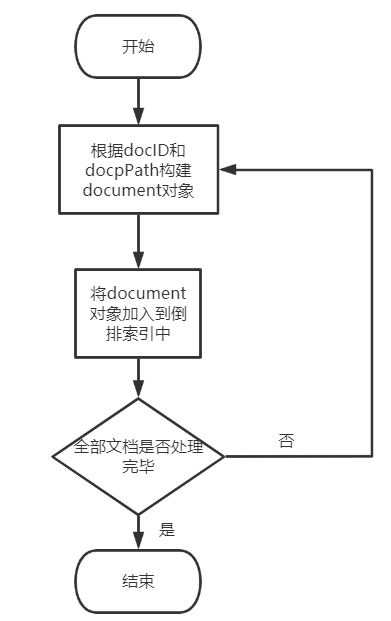
该类的处理流程为：



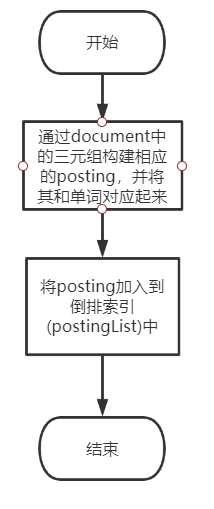
由此可见，为了实现目标，我们需要用到parse模块得到过滤好的三元组流，这也是两模块的联系点。

2. indexBuilder

该类的处理流程为：



仔细分析，会发现加入到倒排索引这一步很重要，因为document对象还需要转化为postingList，接下来叙述倒排索引对象的addDocument方法。



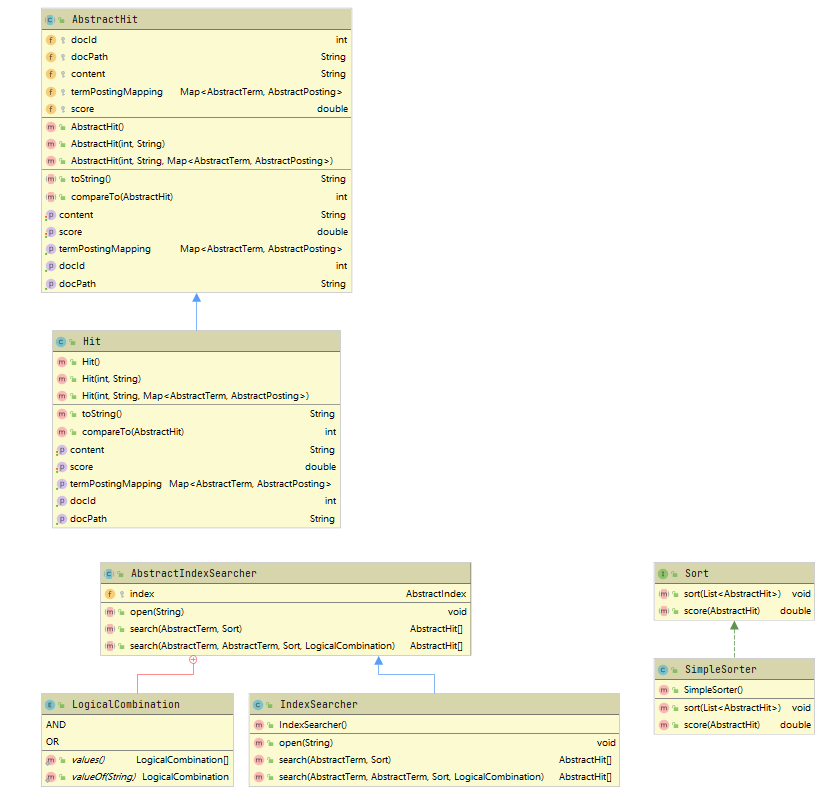
2.2.3 query模块

1. 模块总览

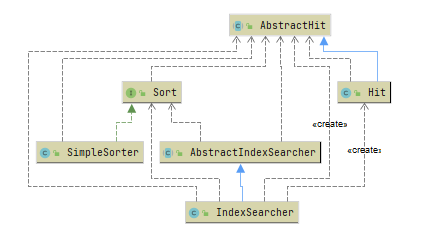
Query模块需要达到的目的有：

1. 构建搜索时的命中类型结构
2. 构建搜索器实现搜索操作

为达成目的，设计出以下类及其继承关系。



各个类之间的依赖关系如下图所示



该模块设计思路如下：

1. 首先对于搜索，我们需要显示搜索结果，因此设计出hit数据结构代表一条搜索结果
2. 多个搜索结果之间具有一定关系，所以设计出simpleSorter类对搜索结果进行排序，注意因为排序是可以以不同方法排的，所以将排序单独抽成一个类，方便排序方法的改变。
3. 最后构建搜索器

2. 具体类的实现

|  |  |  |
| --- | --- | --- |
| 类名 | 功能描述 | 数据成员 |
| Hit | 搜索命中结果数据结构 | int docId：文档Id  String docPath：文档绝对路径  String content: 文档内容  double score = 1.0 : 命中结果得分，默认值为1.0，得分通过Sort接口计算  Map<AbstractTerm, AbstractPosting> termPostingMapping：命中的单词和对应的Posting键值对，对计算文档得分有用，对于一个查询命中结果（一个文档），一个term对应的是Posting而不是PostingList。 |
| IndexSearcher | 搜索器 | AbstractIndex index  倒排索引字典 |
| simpleSorter | 计算得分并排序 | 无 |

接下来将挑选叙述关键类的相关内容。

1. indexSearcher

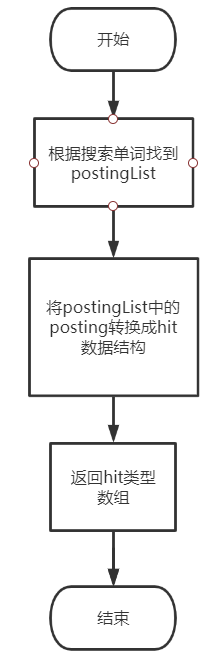
这个类需要达成三个目的：

1. 构建单个词的搜索器
2. 构建两个词的搜索器
3. 构建具有特殊位置关系的两个词的搜索器

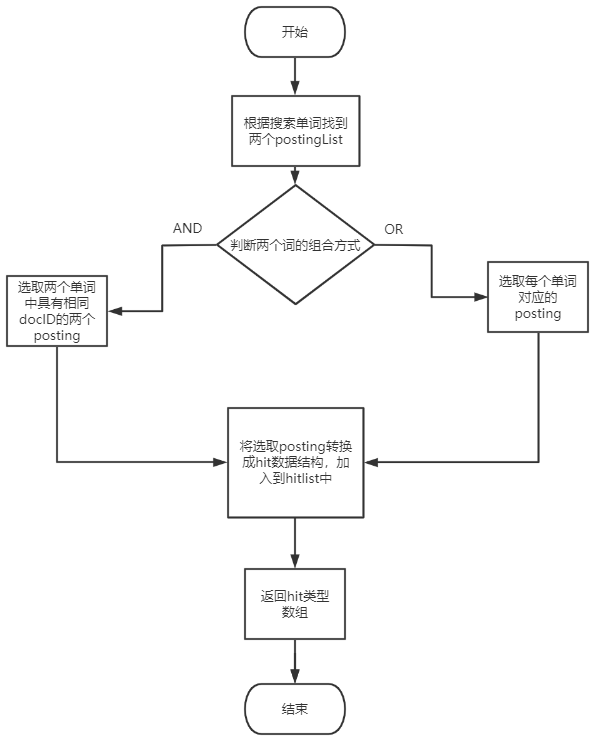
三个目的对应这个类的三个方法，接下来分别叙述

1. 单个词的搜索器

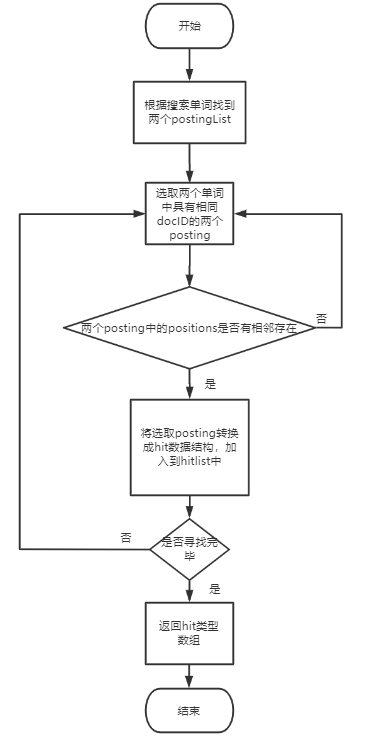
处理流程图如下：



1. 两个词的处理流程图如下：



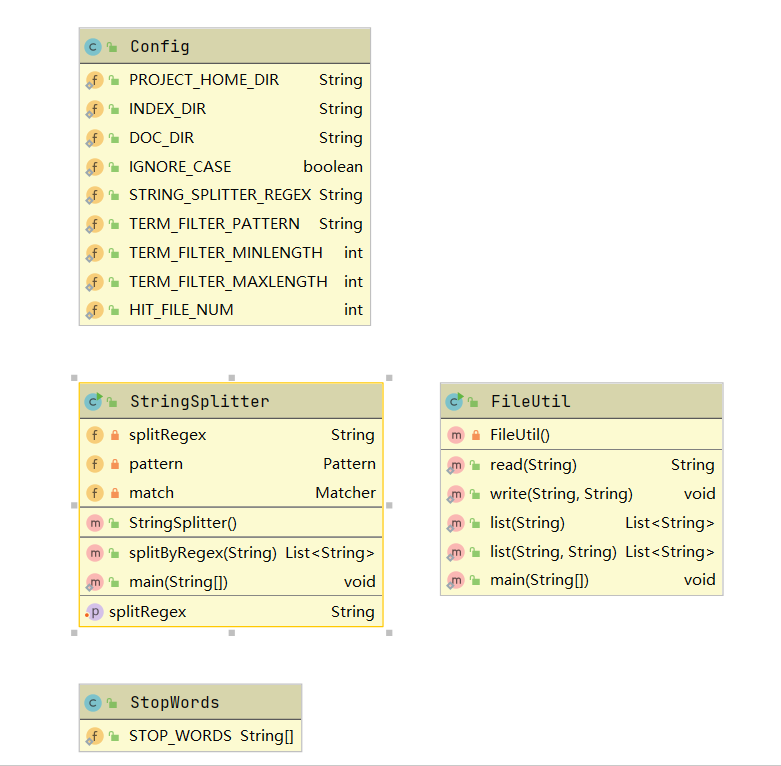
1. 两个具有特定位置关系的单词的搜索



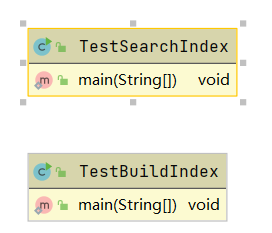
2.2.4 utils模块和run模块

由于这两个模块为工具模块和测试模块，在此不做过多阐述，只说明其中相关的类。

1. utils模块



2. run模块



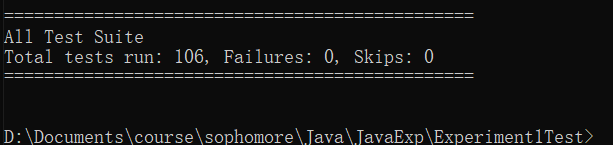
# 三、软件开发

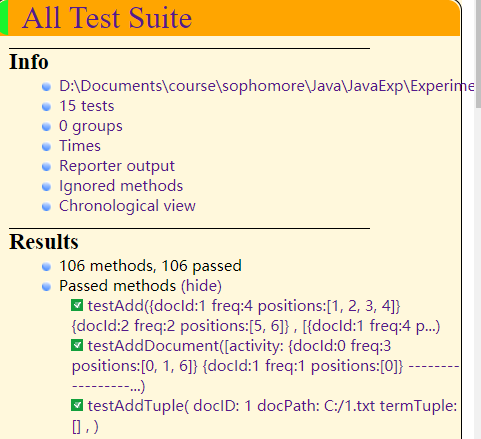
* 本次实验使用intellij IDEA 2020进行编写、编译、调试
* jdk版本为13
* 操作系统为windows10
* 测试利用控制台和testng框架（该测试框架和具体代码由老师编写）

# 四、软件测试

由于时间关系以及模块之间的耦合程度较大，所以未对每个模块单独设计测试，也没有专门制订测试计划，而是直接对整体功能进行测试和调试，该测试由两部分构成，第一部分为老师提供的测试包的测试，第二部分为自己构建的对额外功能的测试。

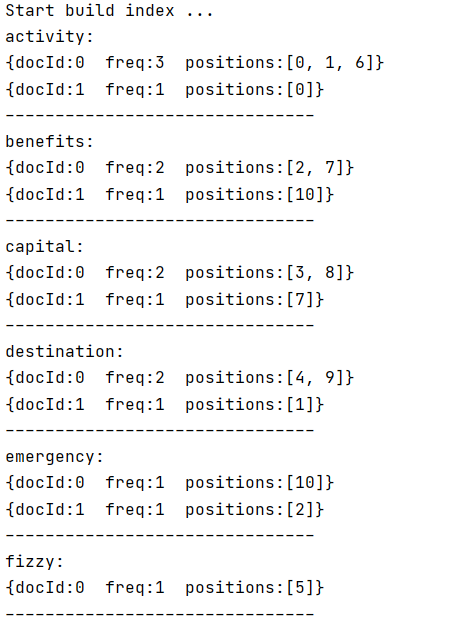
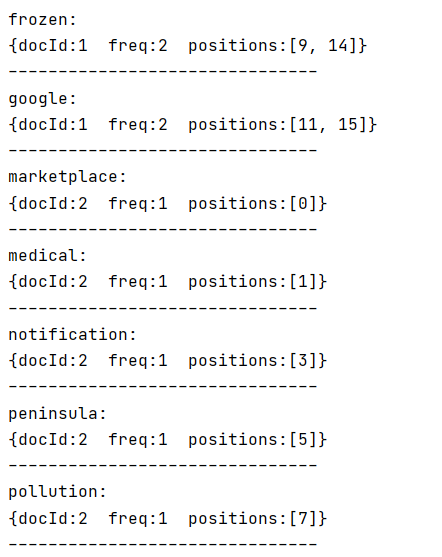
## 2.1 测试包测试结果



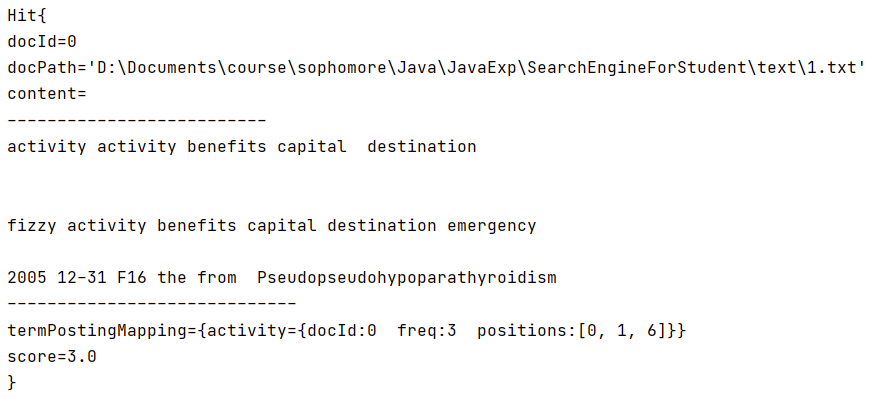


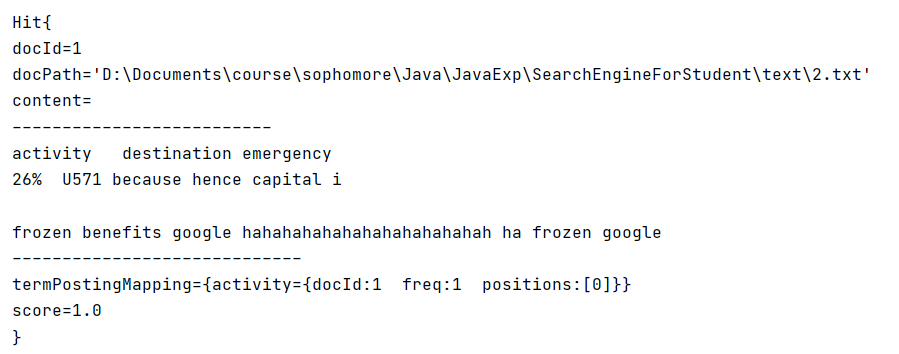
## 2.2 额外功能测试

1. 构建索引

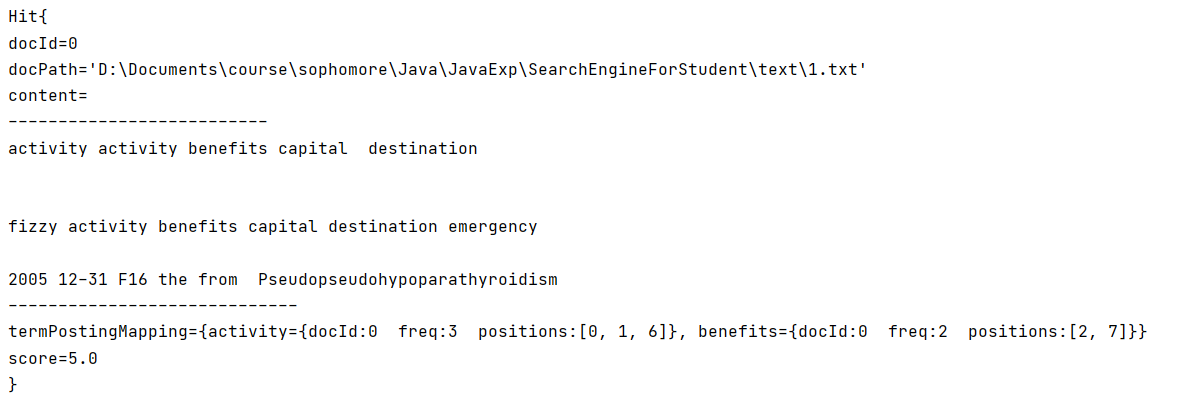
 

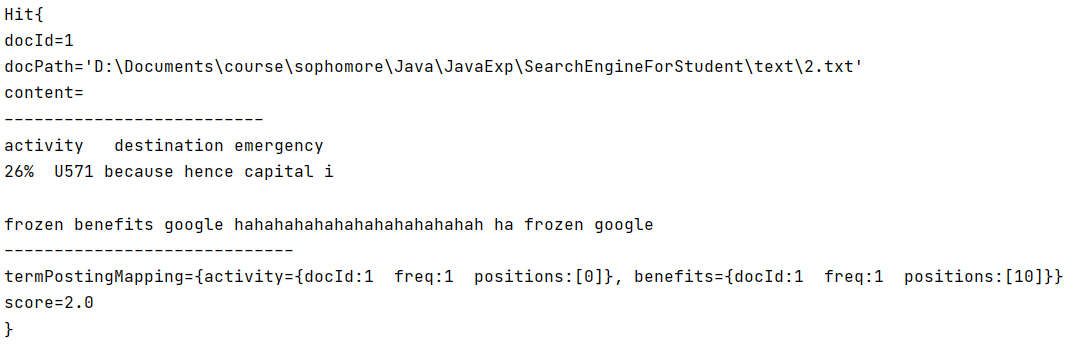
2. 搜索一个词



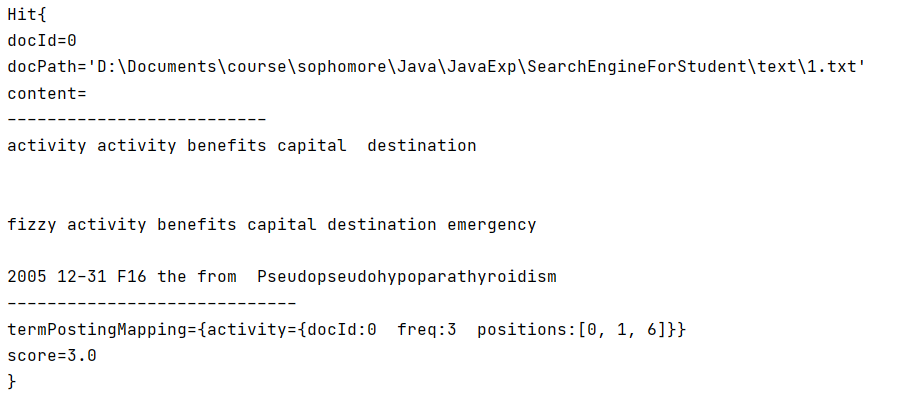


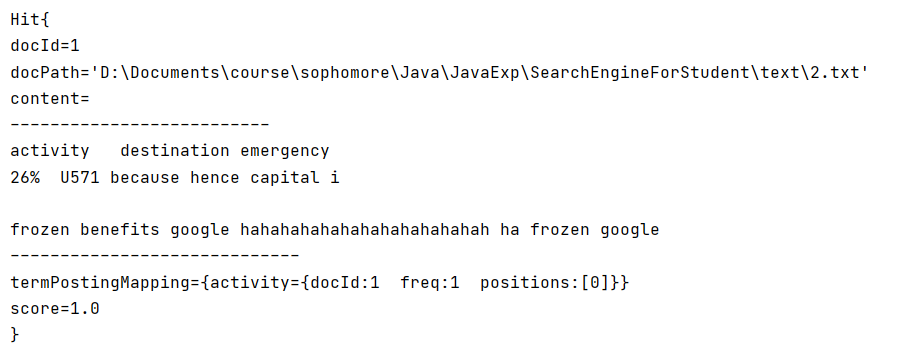
3. 搜索两个词（AND）（activity & benefits）

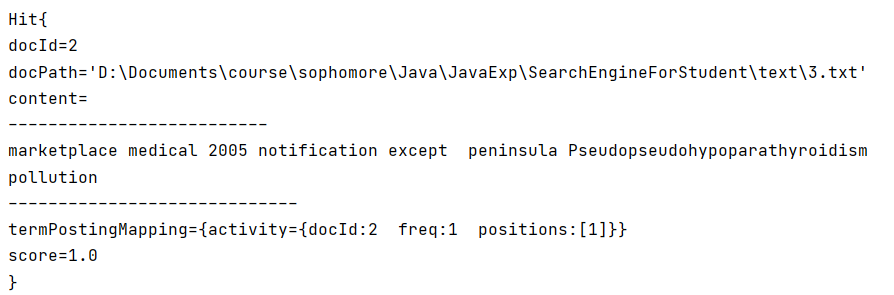




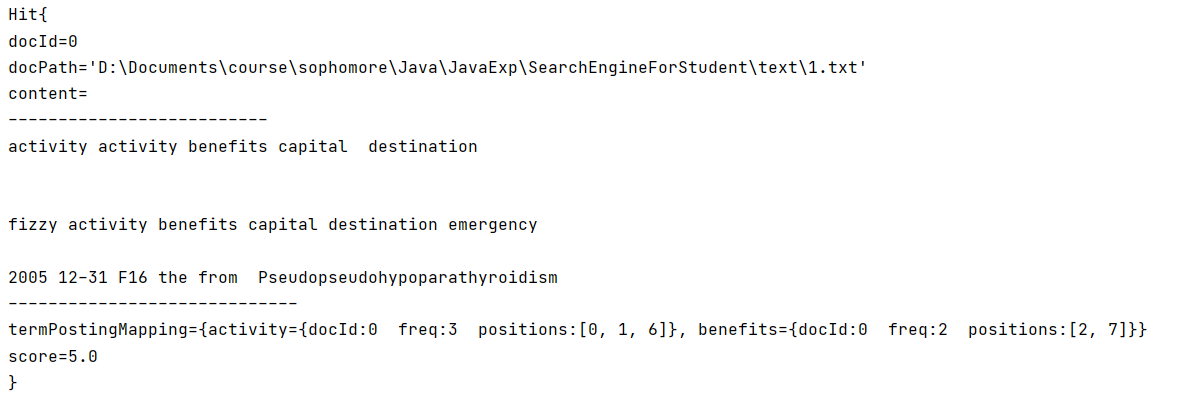
4. 搜索两个词（OR）（activity || medical）



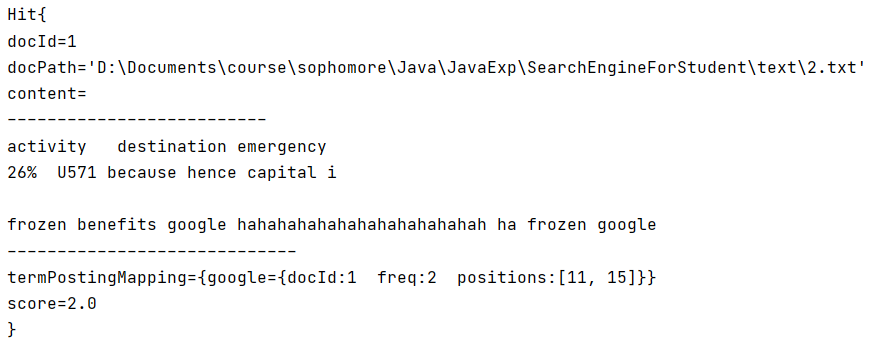


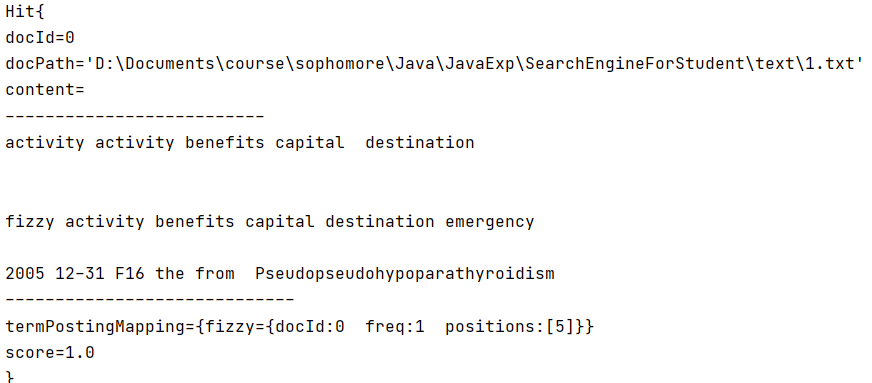


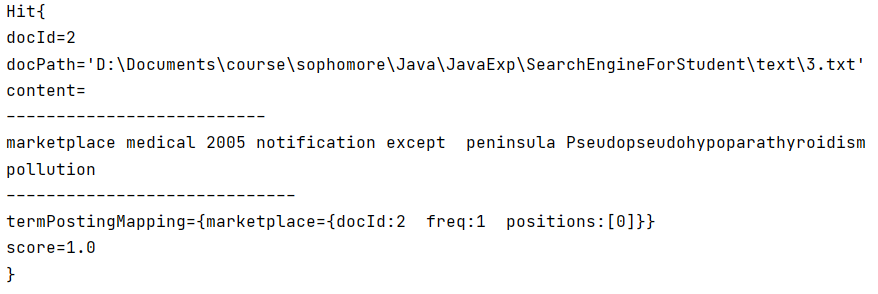
5. 搜索两个相邻的词（activity 、 benefits）（不考虑先后顺序）



6. 对任意数量的词进行搜索



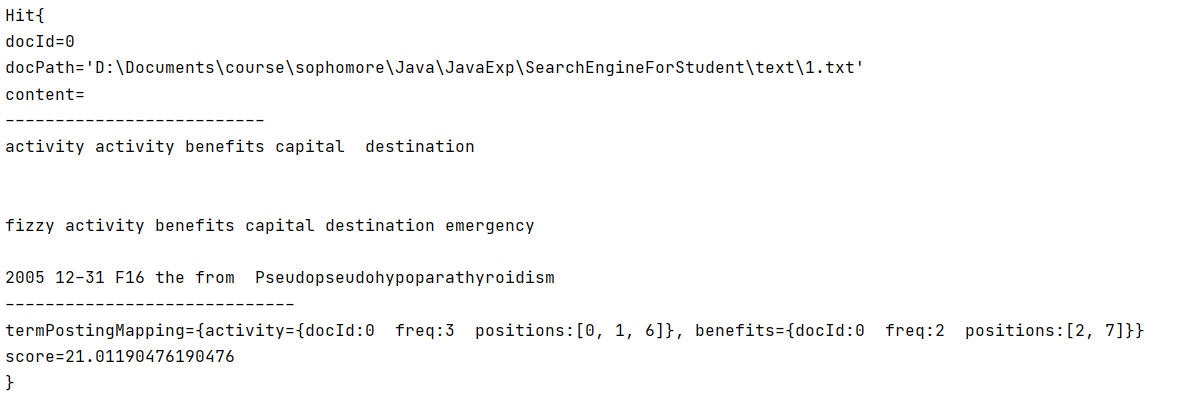


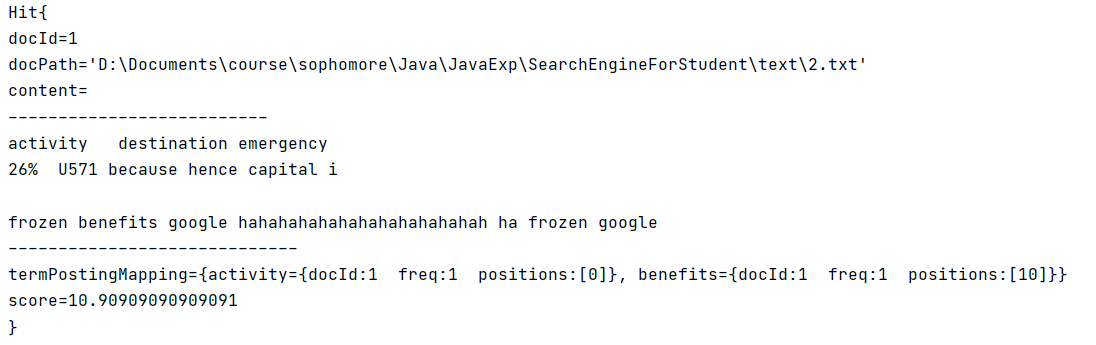


7. attentionSorter

由于一篇文章中，不同位置的词有不同的重要程度，所以考虑在文章不同位置给与单词不同权重来算得分。权重策略为，取位置的倒数，再将所有的倒数加起来作为得分。采取这个策略的思想是：对于大部分需要搜索的文章，大部分都为新闻，而新闻的写作特点是把重要的内容和中心放在文章前面，基于此设计出这个策略综合考虑单词位置和单词频率。







# 五、特点与不足

## 5.1技术特点

本次实验所额外做的功能有三个：

* + 1. 提供对两个具有相邻关系的词进行搜索
    2. 提供对任意数量的词进行搜索
    3. 改进了得分计算模型，综合考虑了单词位置和单词频率

三个额外功能均通过了测试和验证

## 5.2不足和改进的建议

但是本次实验也存在许多不足：

1. 没有实现单词搜索结果的高亮。

2. 没有考虑任意数量词之间的关系搜索，即默认任意数量的词为“或”关系。

3. 没有考虑任意数量词之间存在不同的关系，即可能两个词之间是或关系，但另外零个词之间是“与”关系。

3. 没有进一步优化得分模型。

这些都是可以进一步改进和完善的。

# 六、过程和体会

## 1.遇到的主要问题和解决方法

说实话，本次实验并没有遇到太大的问题，主要就是以下几个小问题：

1. 刚开始对java的语法不熟导致编写程序进度很慢

2. 对java的设计模式，如装饰者模式等模式不熟，导致不能以最优的方式编写代码。

3. 刚开始对java的map，list类型以及序列化等操作不熟，导致进度很慢。

但这些问题都不是很大，都可以通过查阅书籍、网站等资料自行解决。

## 2.课程设计的体会

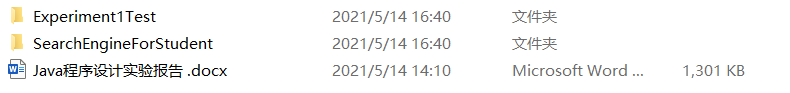
这次课程设计真的是受益匪浅，虽然只有一个实验，但是我觉得至少有几个方面对我有了很大的提升：

1. 首先就是java语法的熟悉，目前已经熟悉java的语法了，比如内部类，lambda表达式等的灵活运用和继承，实现接口等操作
2. 其次是对各种常用类型的熟悉，如map，list，array等
3. 还有就是对java的设计模式有了一定的了解，这个实验中主要是对装饰者模式和工厂模式有了了解。
4. 最重要的是通过这个理解这个实验，我理解到了一个工程的设计思想，拿到一个问题，应该以一种怎样的思维模式去将问题拆解，如何建立模块以至于使得模块之间耦合程度减小方便后期修改，在后面的编写中我也充分体会到了这样编程的好处---改bug特别容易！更改需求特别容易！之前写C面向过程只要稍稍需求有一点改动，需要对整个程序从头到尾改一遍，但是通过这次实验我真正体会到面向对象只需要改某一个类，或者增加某一个类就可以达到目的，真的是太方便了！
5. 当然也不止是工程的设计思想，还有与其相关的测试，验证等，通过观察老师写的测试，了解了testng的测试框架。

# 七、说明

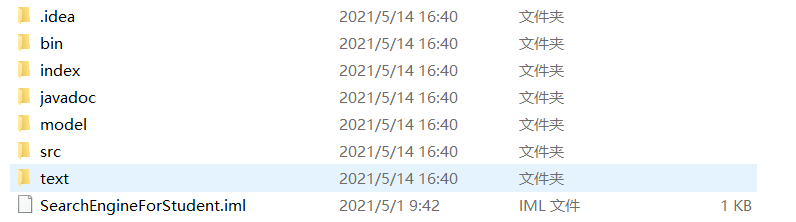
**文件清单及其功能说明**

文件目录结构如下图所示：



其中Experiment1Test为测试包，SearchEngineForStudent为项目文件夹

SearchEngineForStudent目录结构如下：



其中bin中为编译好的文件，index为构建好的倒排索引目录，model中有程序的uml模型，src中为源码，text中为数据集。