

**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 ：大数据2201班**

**学 号 ：**

**姓 名 ：**

**指导教师 ：纪俊文**

**2024年 6 月 5 日**

1. **需求分析**
2. **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。**

1. **需求分析**

**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

（1）对给定的目录，逐个读取每一个文本文件，建立存储每个文档信息的结构(document)用于存储每个文档的以下信息：文档编号、文档路径、文档内容；

（2）将每个文本文件的内容切分成一个个的单词，针对每一个单词构建包含此单词信息的三元组(TermTuple)：单词文本、出现的位置、出现的次数；

（3）对给定目录的所有文本文件完成三元组的构建和去重之后建立索引：索引包含两个结构，由文档编号和文档路径构成，用于查询文档；由词语和对应的postinglist构成，用于查找具体词条；

（4）索引的第一个结构使用map结构建立即可；第二个结构需要遍历每一个文档的TermTuple来构建每一个词语的PositionList；

（5）索引对象的序列化与反序列化过程中要注意，对应元素序列化的顺序和反序列化的顺序要一致，否则可能导致反序列化失败；

（6）可以在控制台输出索引的内容，完成相应的toString方法，可能的话需要调用子成员的toString方法。

**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

1. 命中的结果集合需要使用一定的数据结构存储，此处通过Map结果保存，命中的单词term和对应的Posting键值对；

（2）计算每个命中的文档的得分时简单地根据在文档中的出现次数来进行计分；

（3）注意对构建好的索引检索时可能为空的情况，针对这种情况要有相应的处理。

**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。

（1）支持二个关键词的与或查询。先对分别两个关键词进行单词查询，针对查询结果，对于不同的要求（与/或）进行查询结果的筛选或合并，同样要注意无结果的处理；

（2）可以计算每个命中的文档的得分进行排序，在控制台显示命中的文档的详细信息，这两个功能的实现与功能2中类似。

1. **系统设计**
2. **概要设计**

对于指定目录下的文件，读取文件内容后，需要将其拆分为一个个合法的单词，并建立起关于此文档信息的一个Document数据结构，并且建立其相关索引，最后完成词条搜索的过程，流程图如图 1 倒排索引流程图所示。

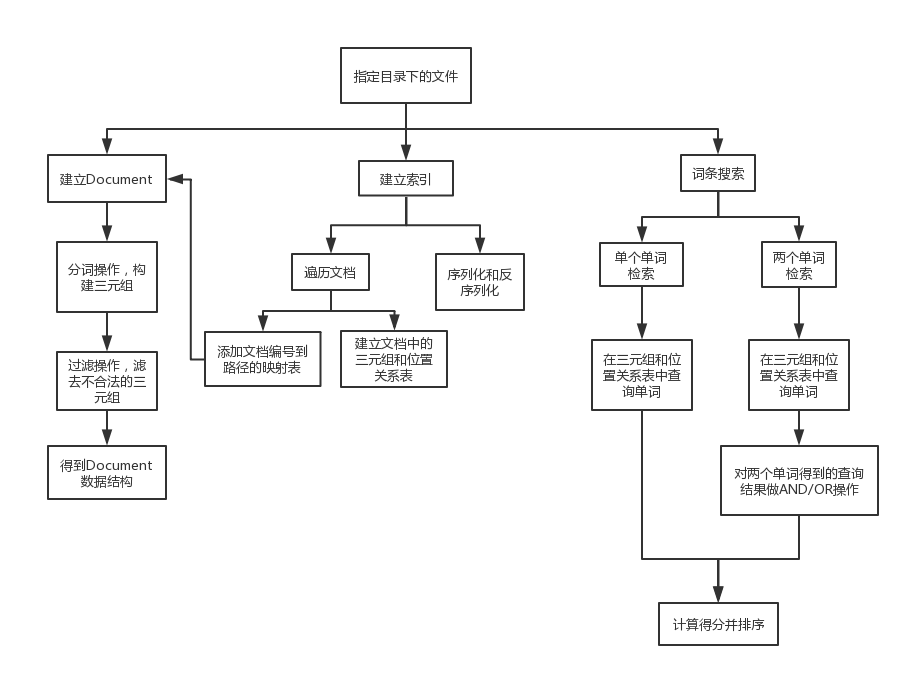


图 1 倒排索引流程图

主体流程分为三个部分，跟工程所分的三个package一样(index, parse, query)，分别为以下内容

1. 对给定的目录下的文件进行扫描，将每一个文本文件通过StringSplitter拆分为单词，同时使用Filter对不满足要求的单词进行筛选，将单词存储为TermTuple格式，并以此来构建Document对象。
2. 对每一个Document建立索引，分为两个部分，对于文档自身的索引以及对于词条的索引，对于后者，我们通过不断地读取Document中的TermTuple来构建针对每一个单词的Postinglist。接着完成序列化以及反序列的功能。
3. 词条查询则直接在对应的term到postinglist的map中寻找对应的词语，并将结果包装为Hit即可。
4. **详细设计**

### 相关存储结构

构建索引用到的相关类的UML图如图 2 倒排索引各结构UML图所示

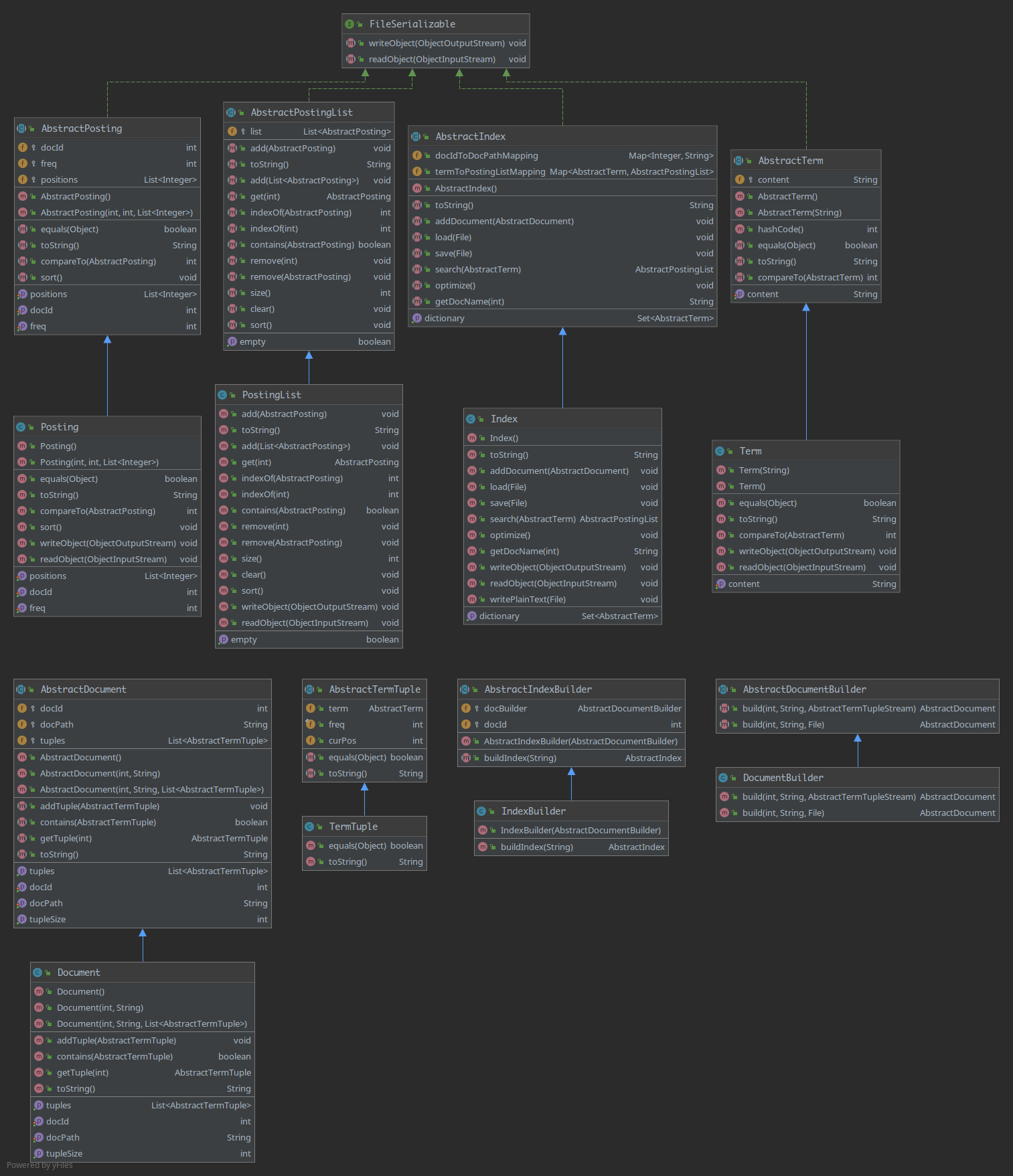


图 2 倒排索引各结构UML图

* Document：AbstractDocument的实现类

Document对象是解析一个文本文件得到结果，文档对象里面包含：文档id；文档的绝对路径；文档包含的三元组对象列表，一个三元组对象是抽象类AbstractTermTuple的子类实例。

字段：

|  |  |
| --- | --- |
| 限定符和类型 | 字段和说明 |
| protected int | docId  文档id |
| protected java.lang.String | docPath  文档绝对路径 |
| protected java.util.List<AbstractTermTuple> | tuples  文档包含的三元组列表 |

构造器：

|  |
| --- |
| 构造器和说明 |
| Document()  缺省构造函数 |
| Document(int docId, java.lang.String docPath)  构造函数 |
| Document(int docId, java.lang.String docPath, java.util.List<AbstractTermTuple> tuples)  构造函数 |

方法：

|  |  |  |  |
| --- | --- | --- | --- |
| 限定符和方法名 | 接收参数 | 返回类型 | 功能说明 |
| public getDocId | 无 | int | 获取文档id |
| public setDocId | int | void | 设置文档id |
| public getDocPath | 无 | String | 获取文档绝对路径 |
| public setDocPath | String | void | 设置文档绝对路径 |
| public getTuples | 无 | List<AbstractTermTuple> | 获取文档包含的三元组列表 |
| public addTuple | AbstractTermTuple | void | 向文档对象里添加三元组, 要求不能有内容重复的三元组 |
| public contains | AbstractTermTuple | Boolean | 判断是否包含指定的三元组 |
| public getTuple | int | AbstractTermTuple | 获取指定下标的三元组 |
| public getTupleSize | 无 | int | 获取文档对象包含的三元组个数 |
| public toString | 无 | String | 获得Document的字符串表示 |

重点方法实现细节：

1. addTuples方法：将传入的Tuple添加到三元组中并保证不重复。
2. contains方法：判断三元组中是否存在传入的Tuple。
3. getTuple方法：根据传入的下标获取Tuple。

* DocumentBuilder：AbstractDocumentBuilder的实现类

Document构造器的功能应该是由解析文本文档得到的TermTupleStream，产生Document对象。

字段：

|  |  |
| --- | --- |
| 限定符和类型 | 字段和说明 |
| 无 | 无 |

构造器：

|  |
| --- |
| 构造器和说明 |
| DocumentBuilder()  缺省构造函数 |

方法：

|  |  |  |  |
| --- | --- | --- | --- |
| 限定符和方法名 | 接收参数 | 返回类型 | 功能说明 |
| public build | int，文档id  String，文档绝对路径  File，文档对应File对象 | AbstractDocument | 由给定的File，构造Document对象 |
| public build | int，文档id  String，文档绝对路径  AbstractTermTupleStream，文档对应的TupleStream | AbstractDocument | 由解析文本得到的TermTupleStream构造Document对象 |

重点方法实现细节：

build方法：既可以通过传入文档id、文档绝对路径和文档对应的TermTupleStream来构建，也可以通过传入文档id、文档绝对路径和文档对应的文档对应File对象来构建，实际使用的时候，会通过调用第二个build函数进而调用第一个build函数，从而完成整个build过程。在第二个build函数中需要调用三种过滤器对得到的三元组进行过滤。

* Index：AbstractIndex的实现类

Index对象是包含了文档集合的倒排索引。内存中的倒排索引结构为HashMap， key为Term对象，value为对应的PostingList对象。另外在AbstractIndex里还定义了从docId和docPath之间的映射关系.

实现下面接口:

FileSerializable：可序列化到文件或从文件反序列化.

字段：

|  |  |
| --- | --- |
| 限定符和类型 | 字段和说明 |
| protected java.util.Map<java.lang.Integer,java.lang.String> | docIdToDocPathMapping  内存中的docId和docPath的映射关系, key为docId，value为对应的docPath. |
| protected java.util.Map<AbstractTerm,AbstractPostingList> | termToPostingListMapping  内存中的倒排索引结构为HashMap，key为Term对象，value为对应的PostingList对象. |

构造器：

|  |
| --- |
| 构造器和说明 |
| Index()  缺省构造函数 |

方法：

|  |  |  |  |
| --- | --- | --- | --- |
| 限定符和方法名 | 接收参数 | 返回类型 | 功能说明 |
| public toString | 无 | String | 获得索引的字符串表示 |
| public addDocument | AbstractDocument | void | 添加文档到索引 |
| public load | File | void | 从索引文件里反序列化出索引 |
| public save | File | void | 将内存中索引序列化储存到文件 |
| pubilc search | AbstractTerm | AbstractPostingList | 获取指定单词的PostingList |
| public getDictionary | 无 | Set<AbstractTerm> | 获取索引中的所有单词 |
| public optimize | 无 | void | 对索引进行优化：  PostingList按照文档id排序  Posting中的positions排序 |
| public getdocName | int | String | 根据文档id获取文档绝对路径 |
| public writeObject | ObjectOutputStream | void | 将内容写入二进制文件 |
| public readObject | ObjectInputStream | void | 从二进制文件读取内容 |

重点方法实现细节：

1. addDocument方法：对传入的Document，将docId和docPath加入docIdToDocPathMapping索引，同时遍历该文档的所有三元组，更新termToPostingListMapping成员。
2. load方法：从索引文件里加载已经构建好的索引，内部调用自身的readObject方法。
3. save方法：在内存里构建好的索引写入到文件，内部调用自身的writeObject方法。
4. search方法：传入Term返回对应的PostingList（如果不存在返回null）。
5. optimize方法：对索引进行优化，即将每一个单词的PostingList按照docId进行排序，同时对每一个Posting里面的position排序。
6. getDocName方法：根据传入的docId获取完全路径名。
7. writeObject方法：写到二进制文件。
8. readObject方法：从二进制文件中读。

* IndexBuilder：AbstractIndexBuilder的实现类

功能：完成索引构造的工作

字段：

|  |  |
| --- | --- |
| 限定符和类型 | 字段和说明 |
| protected AbstractDocumentBuilder | docBuilder  构建索引必须解析文档构建Document对象，因此包含AbstractDocumentBuilder的子类对象 |
| protected int | docId  docId计数器，每当解析一个文档并写入索引，计数器应该+1 |

构造器：

|  |
| --- |
| 构造器和说明 |
| IndexBuilder(AbstractDocumentBuilder docBuilder) |

方法：

|  |  |  |  |
| --- | --- | --- | --- |
| 限定符和方法名 | 接收参数 | 返回类型 | 功能说明 |
| public buildIndex | String | AbstractIndex | 构建指定目录下的所有文本文件的倒排索引 |

重点方法实现细节：

buildIndex方法：接收一个表示根目录的字符串,通过遍历该目录下的文本文件，生成Document并逐步加入Index中，并且存入index.dat文件之中，最终进行返回，需要注意目录为空的情况。

* Posting：AbstractPosting的实现类

Posting对象代表倒排索引里每一项， 一个Posting对象包括:包含单词的文档id，单词在文档里出现的次数，单词在文档里出现的位置列表（位置下标不是以字符为编号，而是以单词为单位进行编号。实现下面二个接口:

Comparable：可比较大小（按照docId大小排序）

FileSerializable：可序列化到文件或从文件反序列化

字段：

|  |  |
| --- | --- |
| 限定符和类型 | 字段和说明 |
| protected int | docId  包含单词的文档id |
| protected int | freq  单词在文档里出现的次数 |
| protected java.util.List<java.lang.Integer> | positions  单词在文档里出现的位置列表（以单词为单位进行编号，如第1个单词，第2个单词，...), 单词可能在文档里出现多次，因此需要一个List来保存 |

构造器：

|  |
| --- |
| 构造器和说明 |
| Posting()  缺省构造函数 |
| Posting(int docId, int freq, java.util.List<java.lang.Integer> positions)  构造函数 |

方法：

|  |  |  |  |
| --- | --- | --- | --- |
| 限定符和方法名 | 接收参数 | 返回类型 | 功能说明 |
| public equals | Object | boolean | 判断两个Posting是否相同 |
| public getDocId | 无 | int | 获取文档id |
| public setDocId | int | void | 设置文档id |
| public getFreq | 无 | int | 获取单词的出现次数 |
| pubilc setFreq | int | void | 设置单词的出现次数 |
| public getPositions | 无 | List<Integer> | 获取单词的出现位置列表 |
| public setPositions | List<Integer> | void | 设置单词的出现位置列表 |
| public compareTo | AbstractPosting | int | 根据文档id比较两个Posting |
| public sort | 无 | void | 对内部positions进行排序 |
| public readObject | ObjectInputStream | void | 从二进制文件读取内容 |
| public writeObject | ObjectOutputStream | void | 写入二进制文件 |

重点方法实现细节：

sort方法：用于将positions进行排序。

* PostingList：AbstractPostingList的实现类

PostingList对象包含了一个单词的Posting列表，必须实现下面接口:

FileSerializable：可序列化到文件或从文件反序列化。

字段：

|  |  |
| --- | --- |
| 限定符和类型 | 字段和说明 |
| protected java.util.List<AbstractPosting> | list  Posting列表，Posting必须是AbstractPosting子类型对象 |

构造器：

|  |
| --- |
| 构造器和说明 |
| PostingList()  缺省构造函数 |

方法：

|  |  |  |  |
| --- | --- | --- | --- |
| 限定符和方法名 | 接收参数 | 返回类型 | 功能说明 |
| public toString | 无 | String | 返回PostingList的字符串表示 |
| public add | AbstractPosting | void | 添加Posting，不能重复 |
| public add | List< AbstractPosting > | void | 添加Posting列表，不能重复 |
| public get | int | AbstractPosting | 获取指定下标的Posting |
| pubilc indexOf | AbstractPosting | int | 返回某个Posting的下标 |
| public indexOf | int | int | 获取指定文档id的Posting下标 |
| public contains | AbstractPosting | boolean | 判断是否包含指定Posting对象 |
| public remove | AbstractPosting | void | 移除指定Posting |
| public remove | int | void | 移除指定下标的Posting |
| public size | void | int | 获取PostingList的大小 |
| public clear | void | void | 清空PostingList |
| public isEmpty | void | boolean | 判断PostingList是否为空 |
| public sort | void | void | 根据文档id对PostingList排序 |
| public writeObject | ObjectOutputStream | void | 将内容序列化写入二进制文件 |
| public readObject | ObjectInputStream | void | 将二进制文件反序列化写入内存 |

重点方法实现细节：

1. add方法：可以一次加入一个Posting也可以一次加入多个Posting（传入由Posting组成的list时），并且确保不重复。
2. get方法：通过下标获取对应的Posting。
3. indexOf方法：由Posting或由docId获取。
4. remove方法：由Posting或由docId删除。
5. size方法：获得PostingList的大小，即含有Posting的个数。
6. contains方法：判断是否包含指定Posting对象。
7. clear方法：清除PostingList。
8. sort方法：根据docId大小对posting进行排序。

* Term：AbstractTerm实现类

Term对象表示文本文档里的一个单词，必须实现下面二个接口:

Comparable：可比较大小（字典序）,为了加速检索过程，字典需要将单词进行排序。

FileSerializable：可序列化到文件或从文件反序列化。

|  |  |
| --- | --- |
| 限定符和类型 | 字段和说明 |
| protected java.lang.String | content  Term内容 |

构造器：

|  |
| --- |
| 构造器和说明 |
| Term()  缺省构造函数 |
| Term(java.lang.String content)  构造函数 |

方法：

|  |  |  |  |
| --- | --- | --- | --- |
| 限定符和方法名 | 接收参数 | 返回类型 | 功能说明 |
| public toString | 无 | String | 返回Term的字符串表示 |
| public equals | Object | boolean | 判断两个Term是否相同 |
| public getContent | void | String | 获取content内容 |
| public setContent | String | void | 设置content内容 |
| pubilc compareTo | AbstractTerm | int | 比较两个Term大小（按字典序） |
| public writeObject | ObjectOutputStream | void | 将内容序列化写入二进制文件 |
| public readObject | ObjectInputStream | void | 将二进制文件反序列化写入内存 |

* TermTuple：AbstractTermTuple实现类

一个TermTuple对象为三元组(单词，出现频率，出现的当前位置).

当解析一个文档时，每解析到一个单词，应该产生一个三元组，其中freq始终为1(因为单词出现了一次).

|  |  |
| --- | --- |
| 限定符和类型 | 字段和说明 |
| int | curPos  单词出现的当前位置 |
| int | freq  出现次数，始终为1 |
| AbstractTerm | term  单词 |

构造器：

|  |
| --- |
| 构造器和说明 |
| TermTuple()  缺省构造函数 |
| TermTuple(Term term, int pos)  构造函数 |

方法：

|  |  |  |  |
| --- | --- | --- | --- |
| 限定符和方法名 | 接收参数 | 返回类型 | 功能说明 |
| public toString | 无 | String | 返回TermTuple的字符串表示 |
| public equals | Object | boolean | 判断两个TermTuple是否相同 |

### 过滤器

当对文档进行单词划分得到初始三元组之后，要利用过滤器将不符合条件的三元组过滤，过滤器的UML图如图 3 过滤器UML图所示。

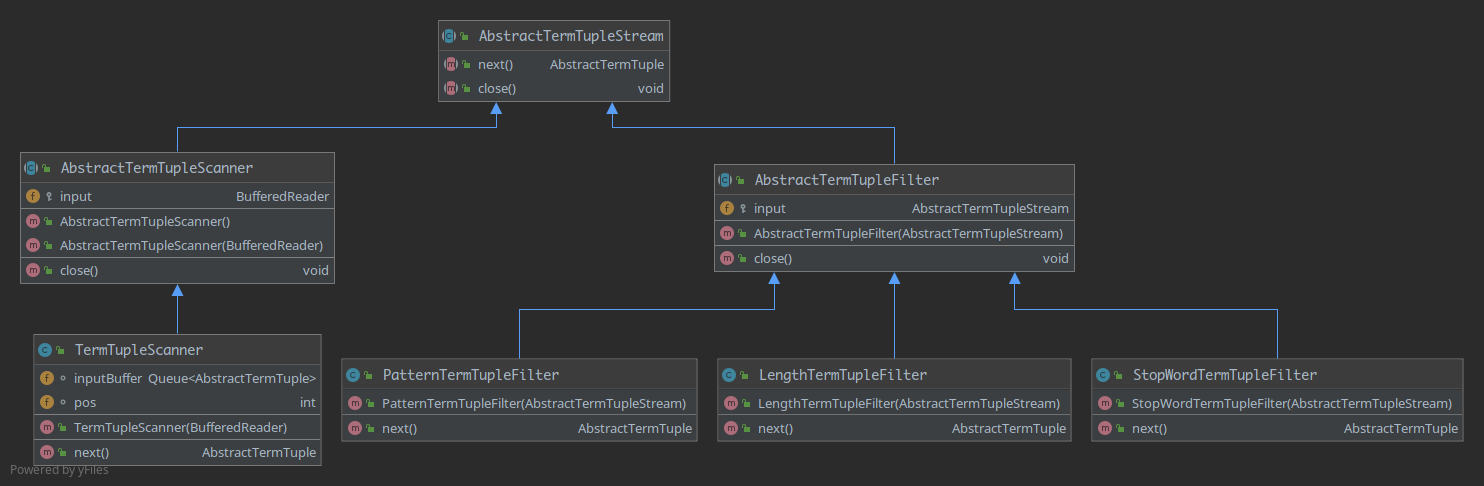


图 3 过滤器UML图

* TermTupleScanner：Abstract TermTupleScanner实现类

AbstractTermTupleScanner是AbstractTermTupleStream的抽象子类，即一个具体的TermTupleScanner对象就是一个AbstractTermTupleStream流对象，它利用java.io.BufferedReader去读取文本文件得到一个个三元组TermTuple。

|  |  |
| --- | --- |
| 限定符和类型 | 字段和说明 |
| protected java.io.BufferedReader | input  input作为输入流对象，读取文本文件得到一个个三元组TermTuple |
| int | curPos  单词出现位置，初始为0 |
| private List<TermTuple> | tuples  解析文本文件获得的三元组 |

构造器：

|  |
| --- |
| 构造器和说明 |
| TermTupleScanner(BufferedReader input)  根据input构造出tuples |

方法：

|  |  |  |  |
| --- | --- | --- | --- |
| 限定符和方法名 | 接收参数 | 返回类型 | 功能说明 |
| public next | void | AbstractTermTuple | 获取下一个三元组 |
| public close | void | void | 关闭流 |

重点方法实现细节：

next方法：进行判断，如果inputBuffer不为空，则直接从inputBuffer中获取数据，如果inputBuffer为空则再次从input流中读取一行，其间需要注意更新pos字段。

* StopWordTermTupleFilter：AbstractTermTupleFilter实现类

StopWordTermTupleFilter用于过滤包含停顿词的三元组。

|  |  |
| --- | --- |
| 限定符和类型 | 字段和说明 |
| private List<String> | stopWord  所有停顿词的列表 |
| protected AbstractTermTupleStream | input  Filter的输入，类型为AbstractTermTupleStream |

构造器：

|  |
| --- |
| 构造器和说明 |
| StopWordTermTupleFilter(AbstractTermTupleStream input)  构造函数，将预置的停顿词列表更新至stopWord |

方法：

|  |  |  |  |
| --- | --- | --- | --- |
| 限定符和方法名 | 接收参数 | 返回类型 | 功能说明 |
| public next | void | AbstractTermTuple | 获取下一个三元组，过滤掉停顿词 |

* PatternTermTupleFilter：AbstractTermTupleFilter实现类

PatternTermTupleFilter用于过滤Term内容包含除字母之外的三元组。

|  |  |
| --- | --- |
| 限定符和类型 | 字段和说明 |
| protected AbstractTermTupleStream | input  Filter的输入，类型为AbstractTermTupleStream |

构造器：

|  |
| --- |
| 构造器和说明 |
| PatternTermTupleFilter(AbstractTermTupleStream input)  构造函数 |

方法：

|  |  |  |  |
| --- | --- | --- | --- |
| 限定符和方法名 | 接收参数 | 返回类型 | 功能说明 |
| public next | void | AbstractTermTuple | 获取下一个三元组，过滤掉Term内容包含除字母之外的三元组 |

* LengthTermTupleFilter：AbstractTermTupleFilter实现类

LengthTermTupleFilter用于过滤Term内容过长或过短的三元组。

|  |  |
| --- | --- |
| 限定符和类型 | 字段和说明 |
| protected AbstractTermTupleStream | input  Filter的输入，类型为AbstractTermTupleStream |

构造器：

|  |
| --- |
| 构造器和说明 |
| LengthTermTupleFilter(AbstractTermTupleStream input)  构造函数 |

方法：

|  |  |  |  |
| --- | --- | --- | --- |
| 限定符和方法名 | 接收参数 | 返回类型 | 功能说明 |
| public next | void | AbstractTermTuple | 获取下一个三元组，过滤掉Term内容过长或过短的三元组 |

### 查询

查询过程有三个步骤：查询、保存查询结果、对查询结果进行排序。查询相关类的UML图如图 4 查询UML图所示。

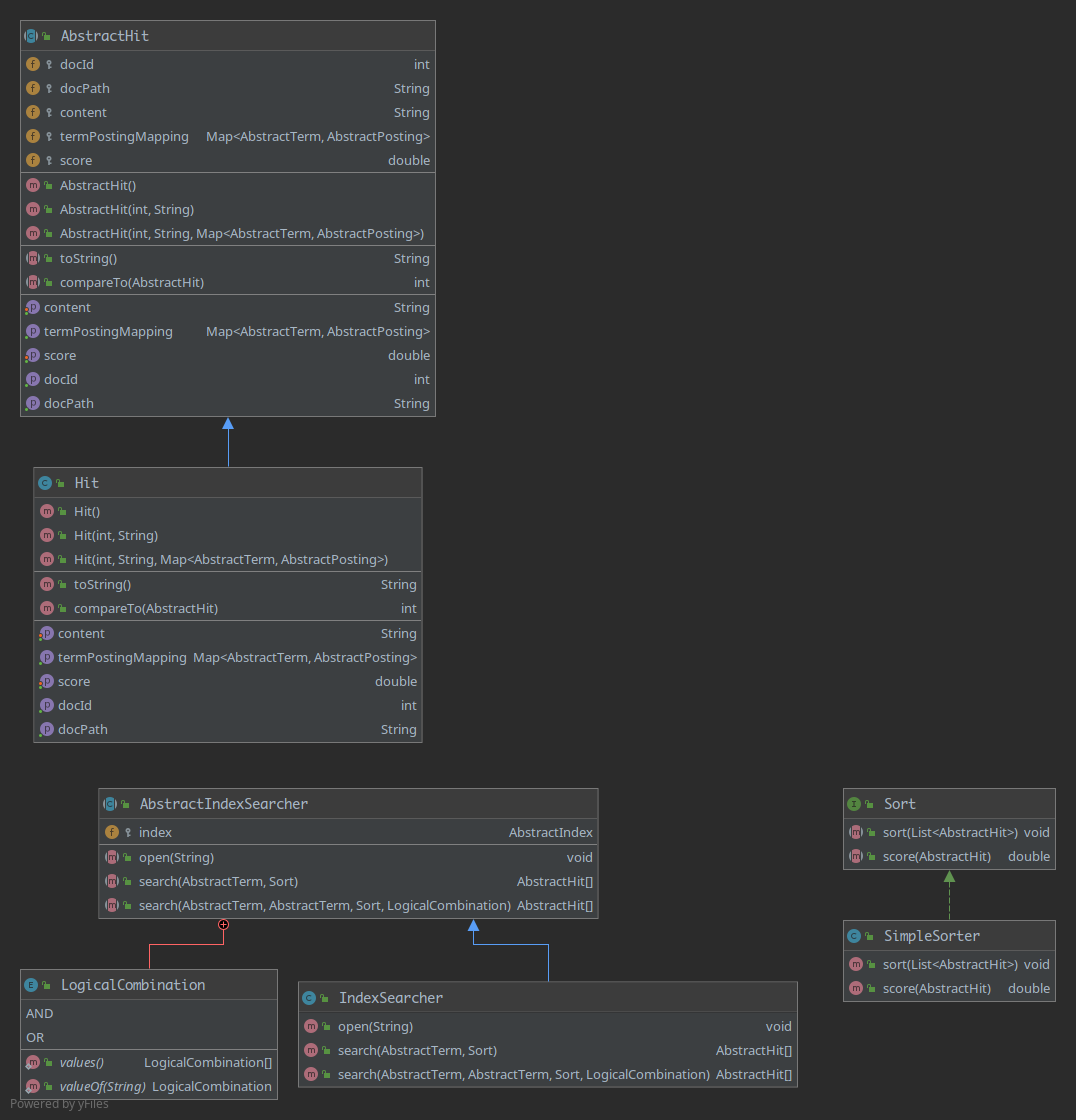


图 4 查询UML图

* Hit：AbstractHit的实现类

AbstractHit是一个搜索命中结果的抽象类。该类子类要实现Comparable接口。实现该接口是因为需要必须比较大小，用于命中结果的排序。

字段：

|  |  |
| --- | --- |
| 限定符和类型 | 字段和说明 |
| protected java.lang.String | content  文档原文内容，显示搜索结果时有用 |
| protected int | docId  文档id |
| protected java.lang.String | docPath  文档绝对路径 |
| protected double | score  该命中文档的得分，文档的得分通过Sort接口计算.每个文档得分默认值为1.0 |
| protected java.util.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term对应的是Posting而不是PostingList |

构造器：

|  |
| --- |
| 构造器和说明 |
| Hit()  默认构造函数 |
| Hit(int docId, java.lang.String docPath)  构造函数 |
| Hit(int docId, java.lang.String docPath, java.util.Map<AbstractTerm,AbstractPosting> termPostingMapping)  构造函数 |

方法：

|  |  |  |  |
| --- | --- | --- | --- |
| 限定符和方法名 | 接收参数 | 返回类型 | 功能说明 |
| public getDocId | 无 | int | 获取文档id |
| public getContent | 无 | String | 获取文章内容 |
| public getDocPath | 无 | String | 获取文档绝对路径 |
| public setContent | String | void | 设置文档内容 |
| public getScore | 无 | double | 获取文章得分 |
| public setScore | double | void | 设置文章得分 |
| public getTermPositingMapping | void | Map<AbstractTerm, AbstractPosting> | 获取命中的单词对应的Posting键值对 |
| public compareTo | AbstractHit | int | 根据得分比较两个Hit，得分相同根据文章id比较 |
| public toString | 无 | String | 获得Hit的字符串表示 |

* IndexSearcher：AbstractIndexSearcher具体实现类

用于对内存中的倒排索引进行搜索。该类有一个嵌套类AbstractIndexSearcher.LogicalCombination用于表示对两个搜索词的与或关系。

字段：

|  |  |
| --- | --- |
| 限定符和类型 | 字段和说明 |
| protected AbstractIndex | index  内存中的索引，子类对象被初始化时为空 |

构造器：

|  |
| --- |
| 构造器和说明 |
| IndexSearcher()  默认构造函数 |

方法：

|  |  |  |  |
| --- | --- | --- | --- |
| 限定符和方法名 | 接收参数 | 返回类型 | 功能说明 |
| public open | String | void | 从指定索引文件打开索引并加载到index对象中 |
| public search | AbstractTerm，Sort | AbstractHit[] | 根据单个检索词进行搜索 |
| public search | AbstractTerm，  AbstractTerm，  Sort，  LogicalCombination | AbstractHit[] | 根据两个检索词以及两个检索词的逻辑关系进行搜索 |

search方法有三个，分别实现对一个词，对两个词和对两个连续单词的检索。对于一个词的检索，直接使用index的search方法，利用返回的PostingList构建Hit数组，对Hit数组排序后返回最终结果；对于两个词的搜索，对两个词分别进行搜索后，针对结合规则，对搜索结果进行合并或者筛选出重合的，利用SimpleSorter排序后返回Hit数组；对于两个连续词的检索，对两个词分别进行搜索后，逐个检验位置中是否存在仅差1的清空，对结果利用SimpleSorter排序后返回Hit数组。

* SimpleSorter：实现了Sorter接口的类

用于对保存后的三元组进行分数计算以及排序

字段：

|  |  |
| --- | --- |
| 限定符和类型 | 字段和说明 |
| 无 | 无 |

构造器：

|  |
| --- |
| 构造器和说明 |
| SimpleSorter()  默认构造函数 |

方法：

|  |  |  |  |
| --- | --- | --- | --- |
| 限定符和方法名 | 接收参数 | 返回类型 | 功能说明 |
| public sort | List<AbstractHit> | void | 对命中的集合根据分数进行排序 |
| public score | AbstractHit | double | 给命中的文章根据检索词出现频率打分 |

1. **软件开发**

使用IntelliJ IDEA Community Edition 2023.3.3 x64作为开发环境。通过已经给出的代码框架进行总体功能的实现。

所用的JDK为openjdk-17版本。

1. **软件测试**

自动测试截图如图 5 自动测试截图和图 6 测试输出结果所示：测试用例106个，完成106个，跳过0个。

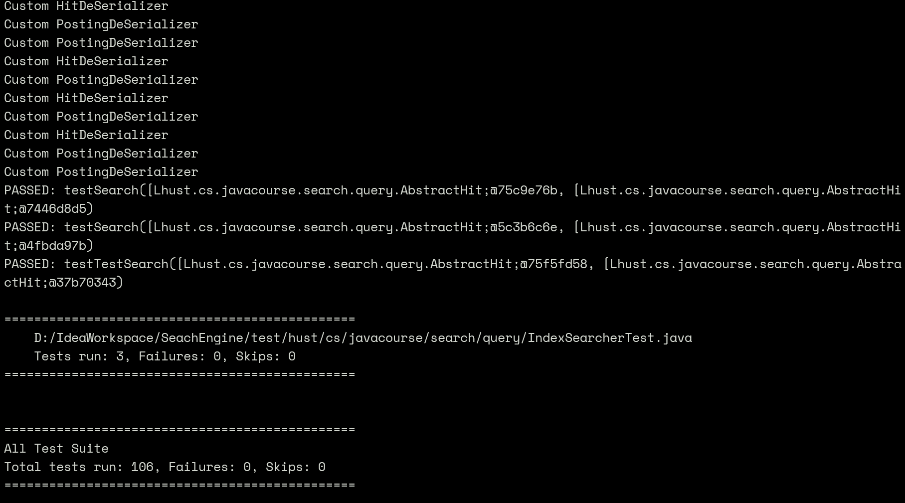


图 5 自动测试截图

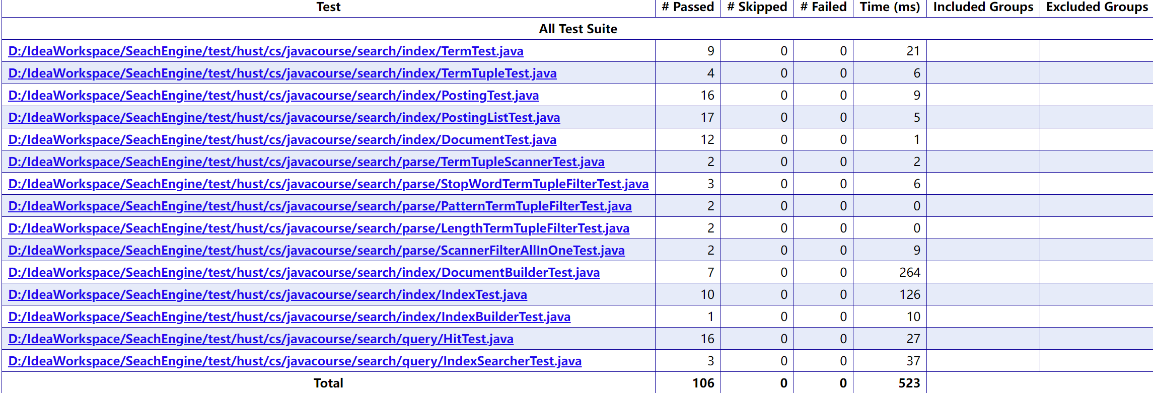


图 6 测试输出结果

可以看出所有的测试用例均通过，即满足了倒排索引的功能完整性。

在程序中编写TestBuildIndex测试程序，并在控制台进行输出，通过构建1.txt~15.txt文件的索引，得到的输出如图 7 TestBuildIndex测试结果所示：

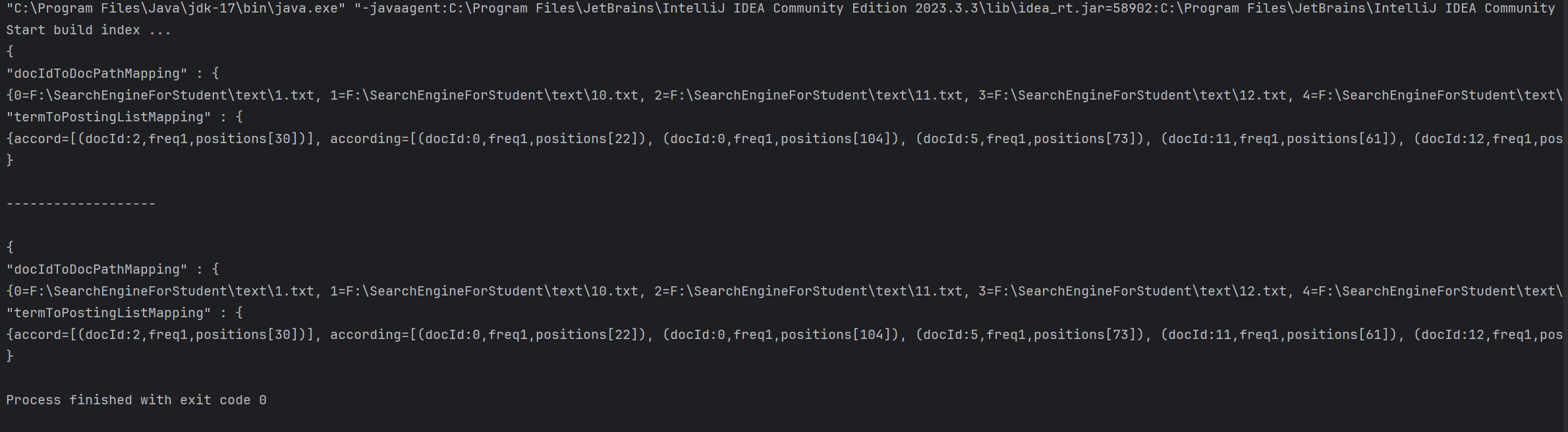


图 7 TestBuildIndex测试结果

可以看出通过对15个文本的索引建立，得到了相关的文件ID、文件路径、文件三元组的相关内容。

根据已经建立好的索引，编写TestSearchIndex测试程序，并在控制台进行个别单词的查找索引，得到输出结果如下所示：

1. 查询一个单词：



图 8 查询一个单词结果

1. 查询两个单词 AND：

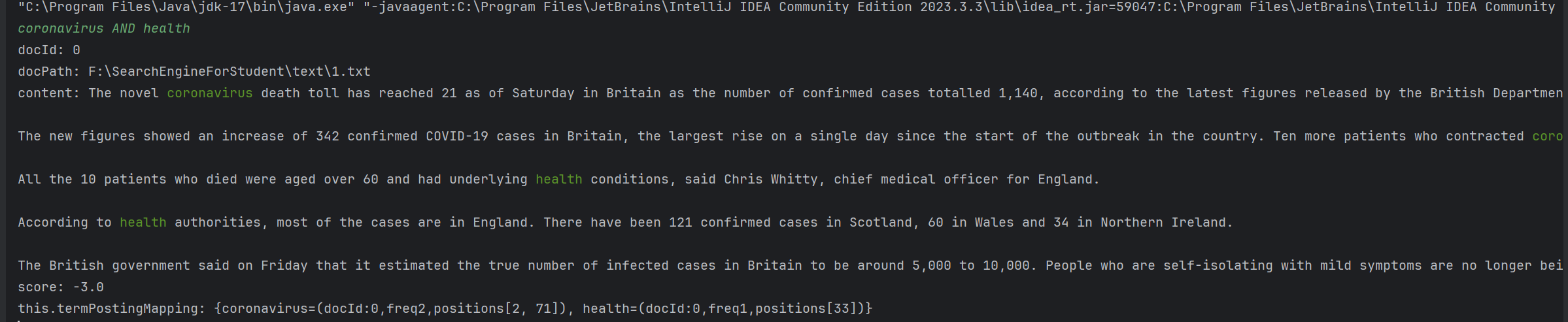


图 9 查询两个单词-同时存在

1. 查询两个单词OR：

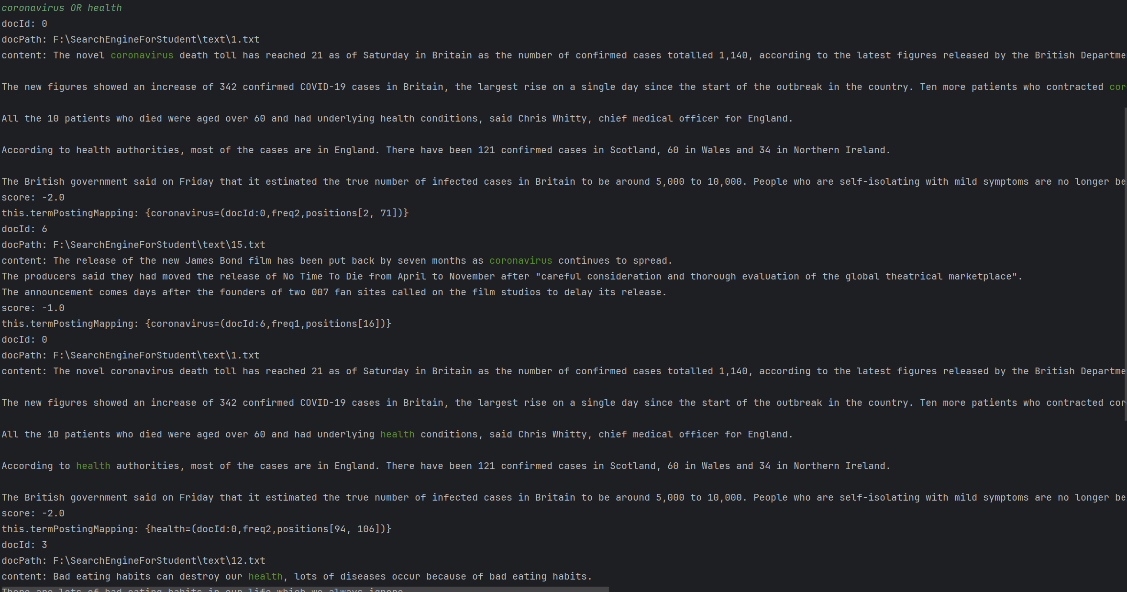


图 10 查询两个单词-存在一个

1. 查询短语：

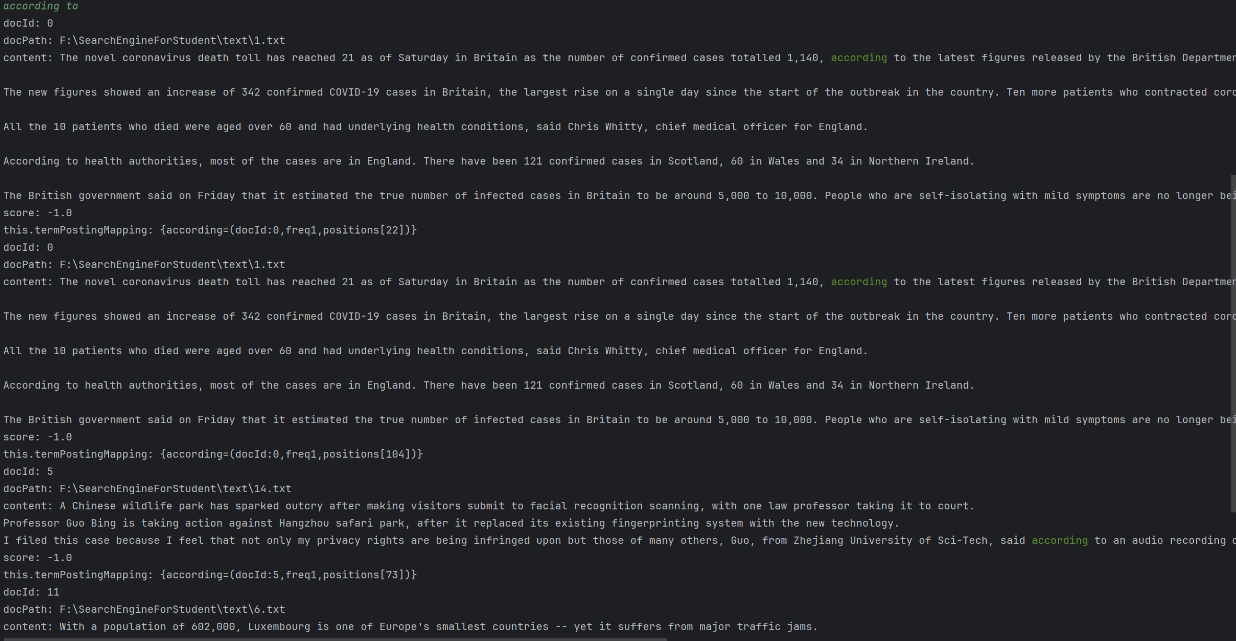


图 11 查询短语

可以看出对于单个单词和两个单词的检索，都能正确找到相应的文档，而且会根据文档中单词出现的频率进行排序输出。

1. **特点与不足**
2. **技术特点**

通过本次实验，我了解了真实的简单项目使如何进行开发的，同时也了解了一些常见的设计模式及思想，比如观察者模式等。同时也首次接触了Java的一些STL如Hashmap和，这有一些类似于C++中的unordered\_map，同样也可以如C++一样利用迭代器或是for循环对其进行遍历。同时在基础任务之上我还完成了对进阶功能的编写，即对词组进行搜索。

1. **不足和改进的建议**

在实验过程中有些算法的代码不够精简，没有考虑到算法的时间空间复杂度。而且在实现具体类的过程中，只是一味地跟着老师预先设置好的抽象类进行实现，没有很多自己的思考。同时，对于某些变量名称总是记混，代码中也缺乏注释，在实现某些功能时忘记了曾经曾定义过某些功能可以帮助此功能的实现，反而重新编写它，代码的复用率较低。

要加强编程思想的培养，加强设计策略的学习，同时要习惯规范的变量命名，即使有些名称很长，同时要加强注释代码以及复用代码的意识。

1. **过程和体会**
2. **遇到的主要问题和解决方法**
3. 自动测试时会产生“找不到文件”或关于NULL的空错误。尝试了很多解决方案，修改代码部分、检测自动测试的反编译文件、查看测试文件的正确性、修改序列化和反序列化等等，都无法解决。最后发现原来是中文路径的缘故（当知道是这么简单的错误时，心态有点复杂），然后将测试文件放到全英文无空格的路径下，再次运行就能通过了。
4. 查询测试有时会找不到对应的文档，即所输出的结果和预期的有些出入。在控制台进行单词查询时，会少输出几个应该出现的文件，原因是因为查询的单词中有大写字母，而查询操作没有实现大小写无区别查询，故后来修改成大小写无差别查询后，得到解决。
5. Filter函数的实现问题，起初的过滤器是全部放在一个文件里实现的，但是后来仔细阅读实验要求后，发现需要拆分成三个过滤器。因此后来对于这个大的过滤器进行了三部分的拆分。
6. **课程设计的体会**

此次实验让我深刻体验到了什么是所谓的面向对象编程，其主要思想就是把一个庞大的系统抽离成一个个具体的模块，每个模块中有自己的属性以及方法，最后再将这些模块整合到这个系统中去。这不仅考验了我们的代码能力更考验了我们的整体设计能力。虽然老师已经将整体的框架搭建起来，但其中的一些细节的处理方式还需要我们自己想办法去实现。这不仅锻炼了我们的代码实现能力，更在更高层面锻炼了我们的设计能力。

虽然离代码真正落地上线还有很长一段距离，但是通过这次试验让学生们了解了许多知识如常见的设计模式，序列化以及反序列化操作。学生通过逐步完善每一个类最终完成一个项目，是挺有成就感的一件事。