

**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1803班**

**学 号 ： U201815553**

**姓 名 ： 周大伟**

**指导教师 ： 马光志**

**2021年 4 月 23 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。

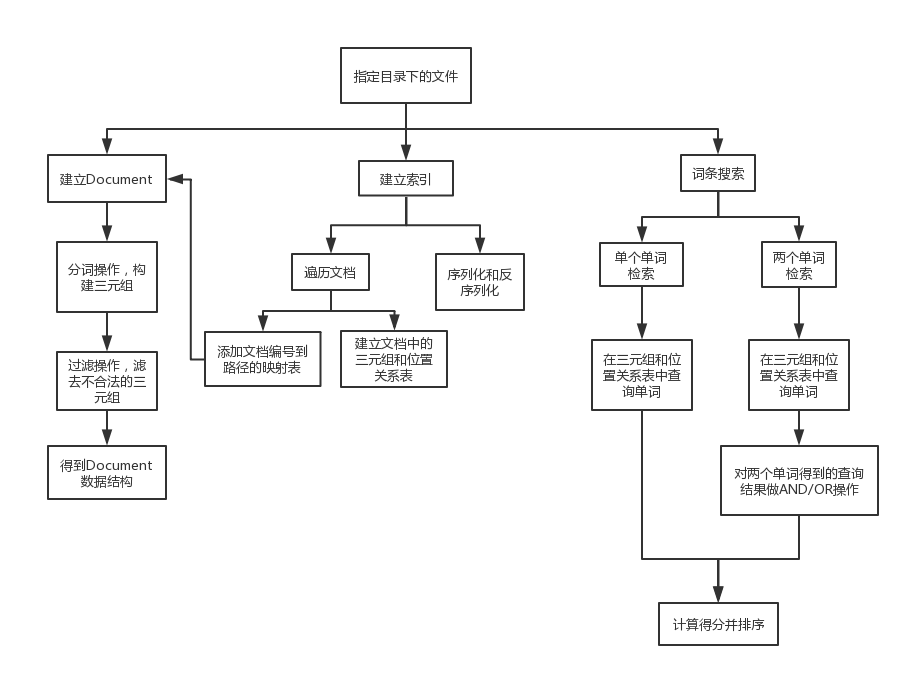


图 1 倒排索引流程图

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

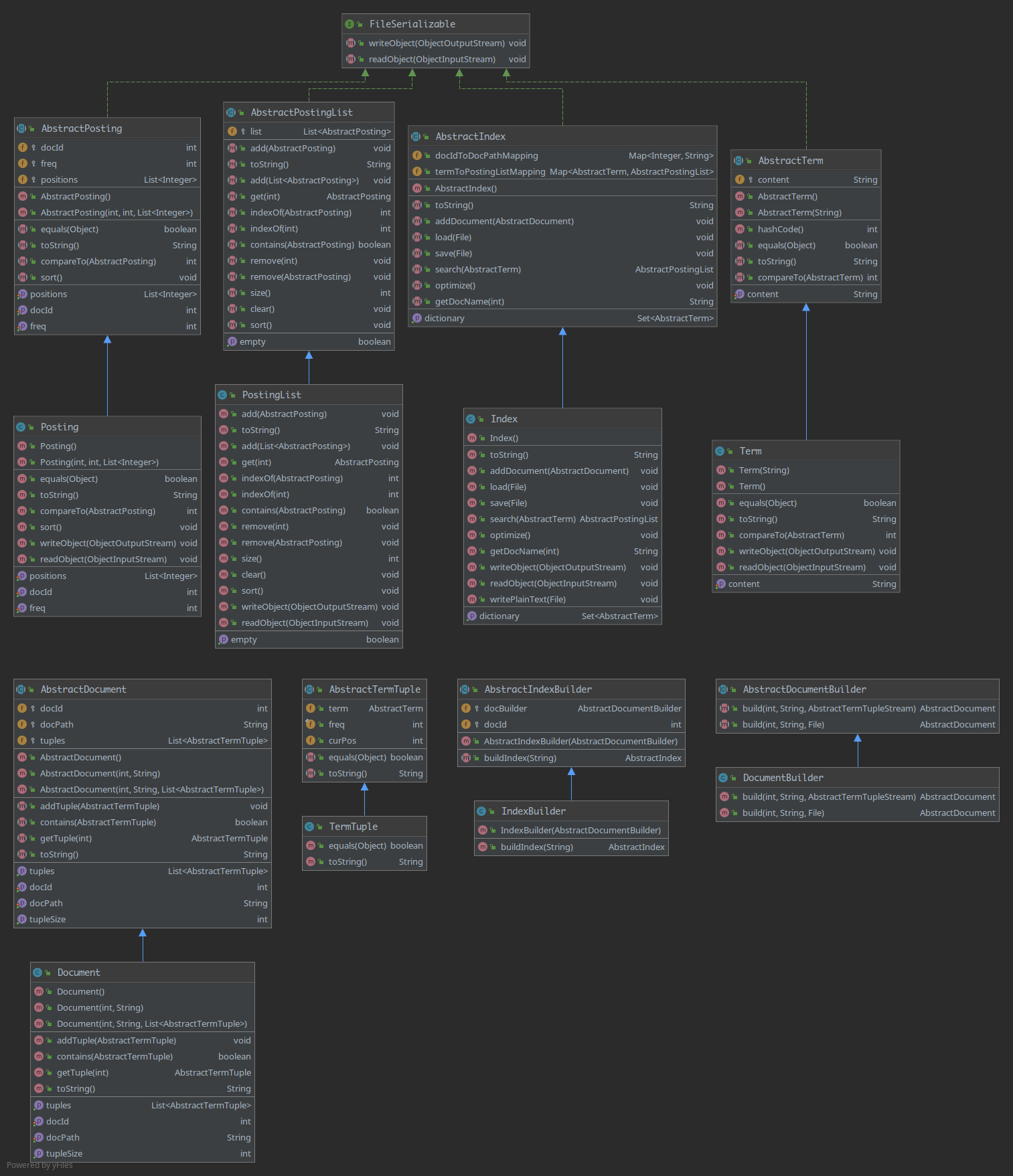


图 2 倒排索引各结构UML图

**Term**：

本质上是对于String类型的简单包装，所以实现起来非常简单，在此不再赘述；

**TermTuple**：

对于某一次单词出现情况的包装，主要有三个成员，term, freq, curPos，分别表示单词本身，出现的频率(默认为1)，以及在某一个文档中出现的位置。

**Document**：

保存每一个文档的基本信息，包括文档所在路径docPath，文档唯一编号docId，同时还有组成整个文档的所有单词所构成的TermTuple的List。

addTuples方法：将传入的Tuple添加到三元组中，contains方法判断三元组中是否存在传入的Tuple。

getTuple方法：根据传入的下标获取Tuple。

**Posting**：

主要用于存储某一个Term在每个文档中的信息，主要的数据成员有三个：文档编号docId，这个Term在文档中出现的次数freq，每次出现的位置positions。

sort方法：用于将positions进行排序。

**Postinglist**：

指定目录下所有文档中对应的Term的信息，本质上是对一个由posting所构成的list的包装。

add方法：可以一次加入一个posting也可以一次加入多个posting（由posting组成的list），需要注意要进行去重。

get方法：通过下标获取对应的posting。

indexOf方法：由Posting或由docId获取。

remove方法：由Posting或由docId删除。

sort方法：根据docId大小对posting进行排序。

**Index**：

实现索引，最主要的两个个数据成员：以map实现文档索引的docIdToDocPathMapping和实现关键词索引的termToPostingListMapping。

实现了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。

load和save方法：实际山就是将File类型的参数转化为Object的流类型，从而调用自身的writeObject/readObject方法。

search方法：传入Term返回对应的postinglist（如果不存在返回null）。

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。

getDocName方法：根据传入的docId获取完全路径名。

writeObject和readObject方法：对所建立的索引进行序列化和反序列化操作。

**DocumentBuilder**：

构建文档的Document，主要是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Filter对得到的三元组进行过滤。

**Indexbuilder**：

构建整个倒排索引，在这个类中利用上述结构和类，对于传入的目录进行操作最终得到整个倒排索引结构。比较重要的数据成员是用于构建每个文档的docBuilder和用于给文档赋予Id号的docId，这个类的主要方法buildIndex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，最终进行返回，需要注意目录为空的情况。

### 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。

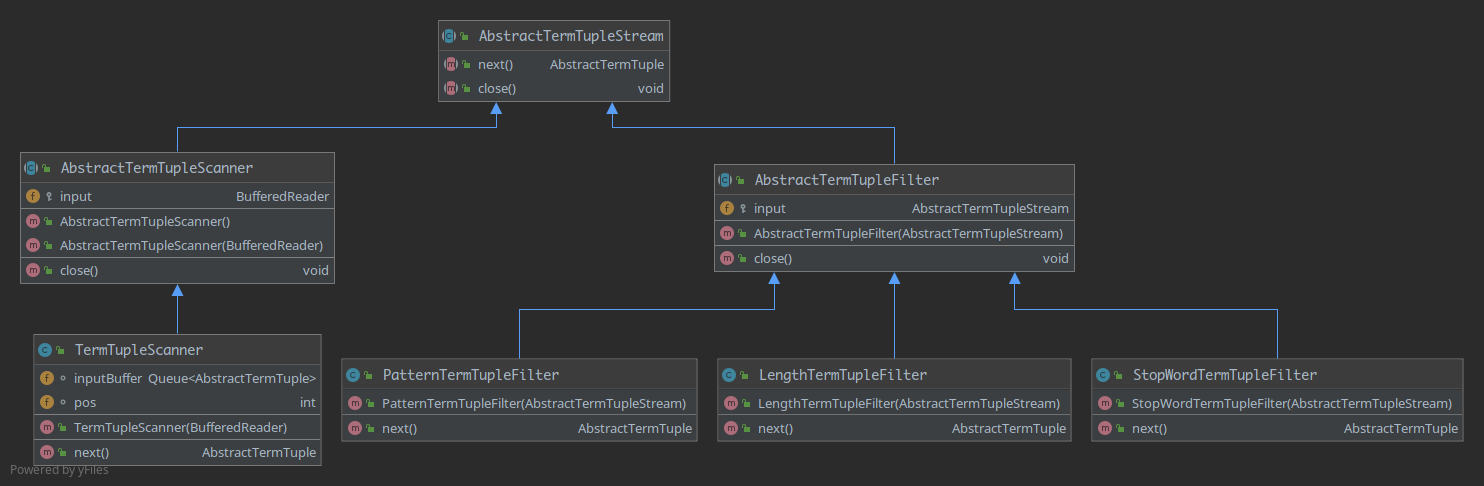


图 3 过滤器UML图

**TermTupleScanner**：

主要数据成员：pos用于统计单词位置，inputBuffer用于存放临时读到的内容，由于读取文档的时候是一行一行读取，但是取出来的时候是一个单词一个单词的取出来，所以就需要临时存放多余的内容，当调用next方法的时候，需要进行判断，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，需要注意的是需要根据config中的选项来判断是否需要转化大小写

三个具体的Filer都是通过对AbstractTermTupleStream不断调用next方法来获取单词，进而将符合要求的返回，不符合要求的则直接丢弃。

**StopWordTermTupleFilter**：

通过StopWords类中的STOP\_WORDS字段来判断是否是停用词，如果是则丢弃，不是则保留。

**PatternTermTupleFilter**：

通过Config中的TERM\_FILTER\_PATTERN，进行正则匹配，符合则保留，不符合则丢弃。

**LengthTermTupleFilter**：

通过Config中规定的TERM\_FILTER\_MAXLENGTH以及TERM\_FILTER\_MINLENGTH来判断单词是否在规定的长度范围之内，丢弃不满足要求的单词。

### 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。

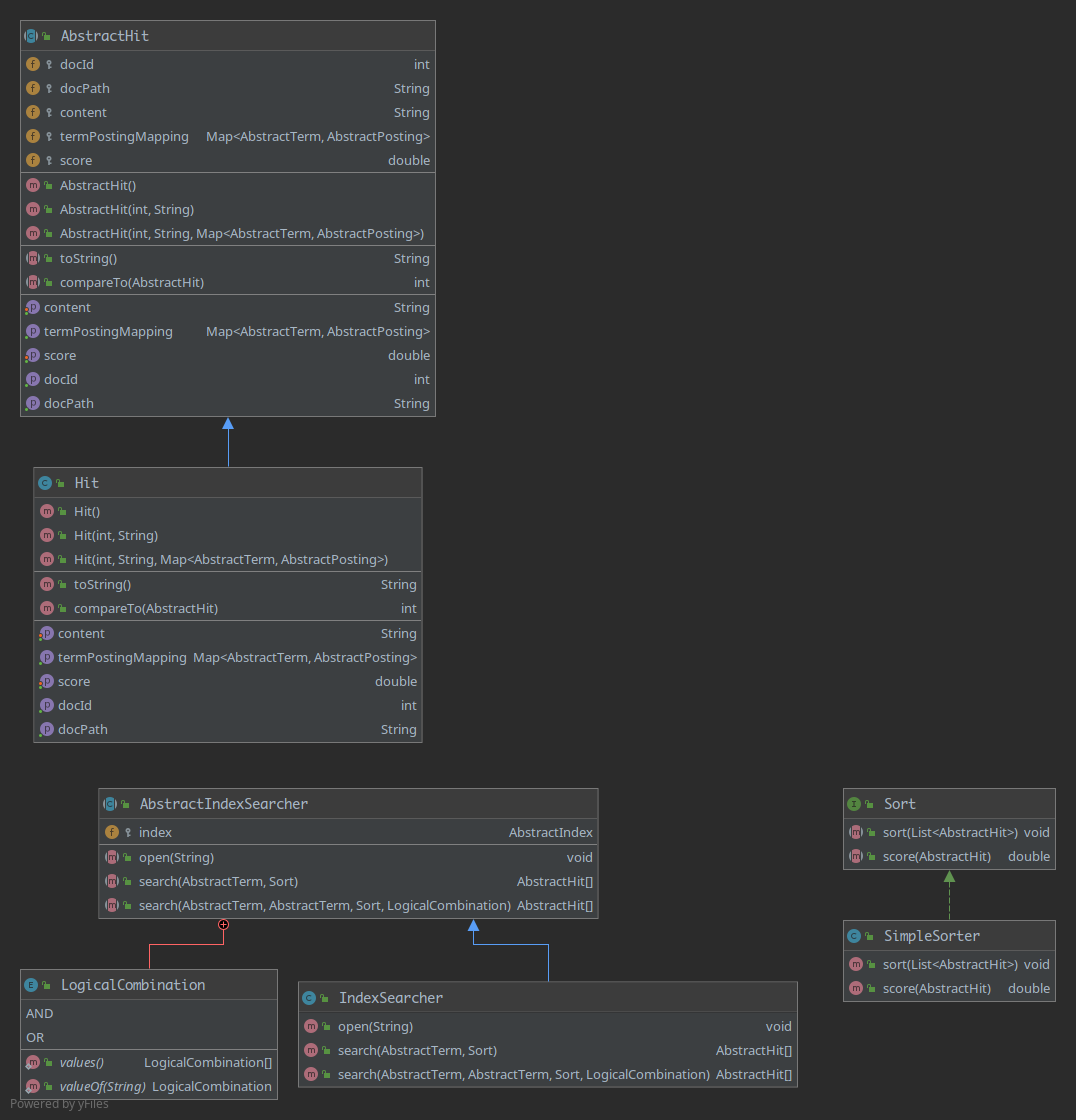


图 4 查询UML图

**Hit**：

保存查询的一个结果，数据成员有docId、docPath、content、termPostingMapping、score，其中的content要保存对应的文档中的全部内容，termPostingMapping中保存查询到的命中的单词和对应的Posting键值对，查询结果返回一个Hit的数组。

**IndexSearcher**：

根据构建好的索引搜索词条，数据成员index用于保存要用的索引。

open方法：将索引文件实例加载到index中。

search方法：有两个，分别实现对一个词和对两个词的检索。

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。

**SimpleSorter**：

score方法计算命中文档的得分，根据词条出现的频率进行赋值，返回得分的负值，sort方法对传入的命中结果集合根据文档得分排序，需要注意的是两个词的搜索，这里采用了加和的方法，也即两个词的得分相加。

注：所有必要的基本方法都进行了实现，如toString，writeObjec, readObject, equal方法以及必要的构造函数

1. **软件开发**

使用IntelliJ IDEA Community Edition 2020.3.3 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-15.0.2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。

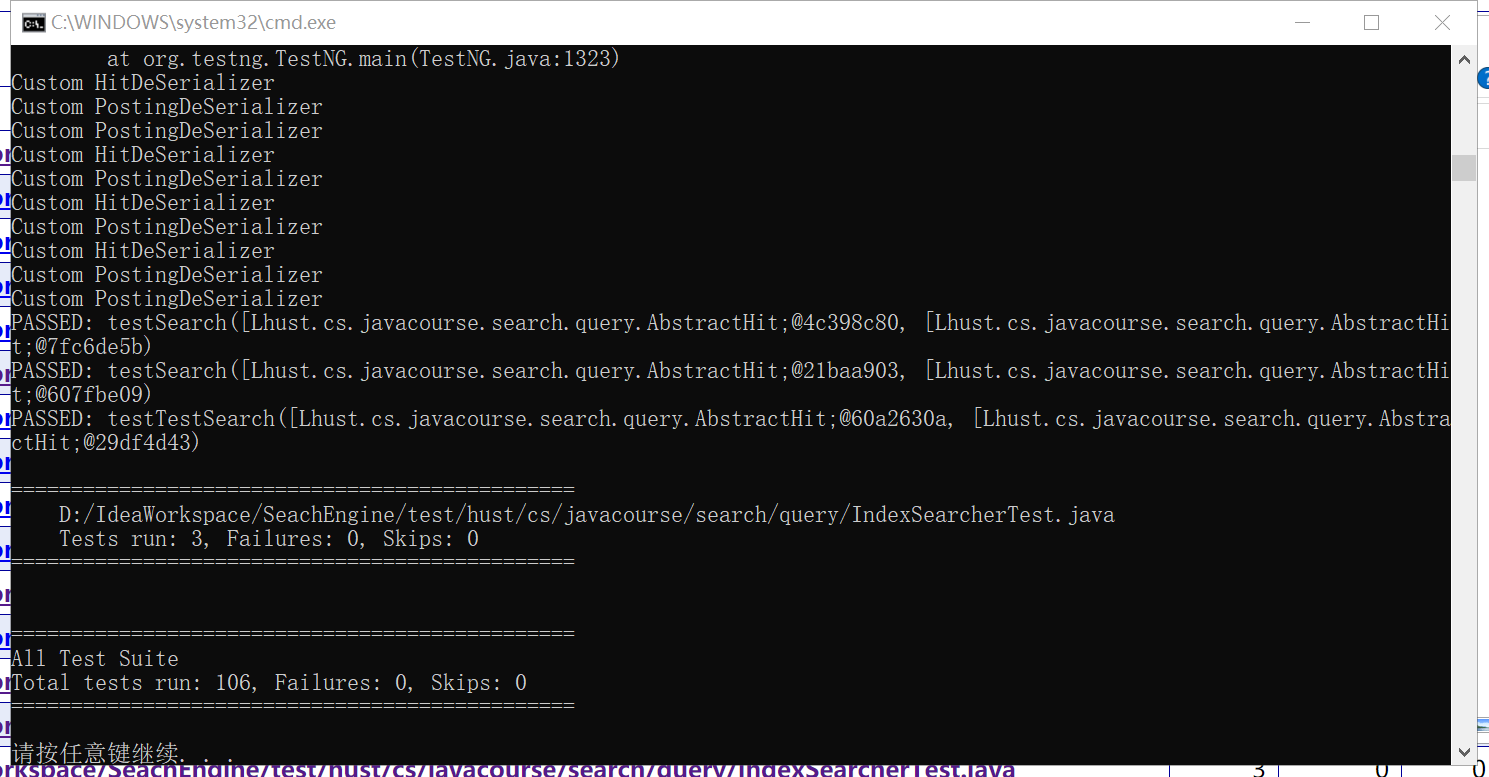


图 5 自动测试截图

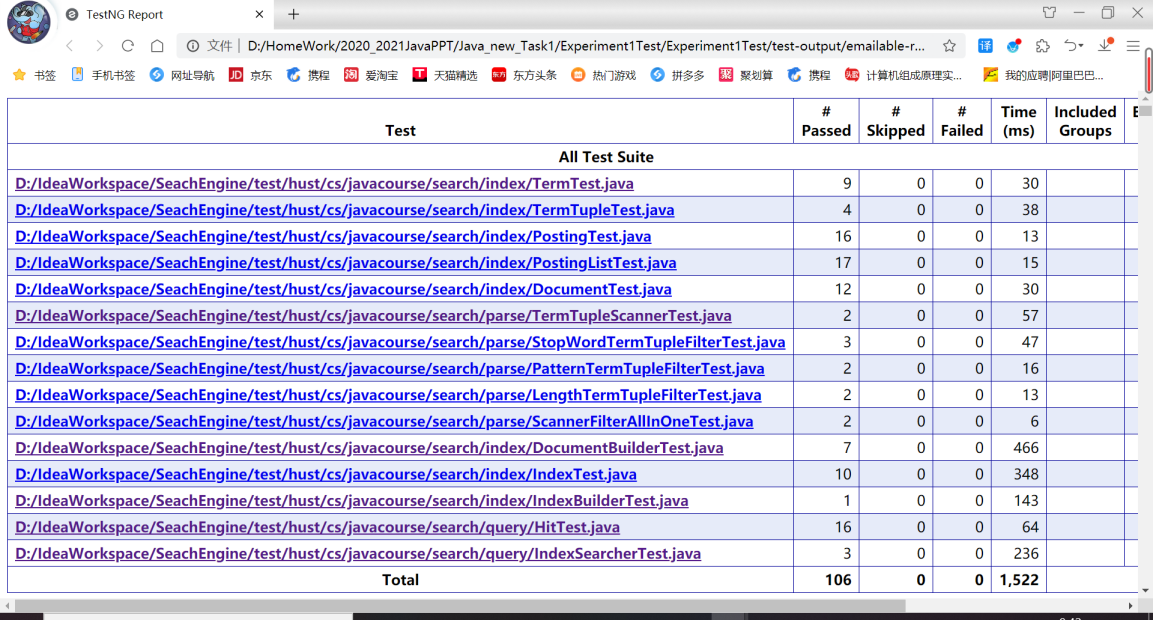


图 6 测试输出结果

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：

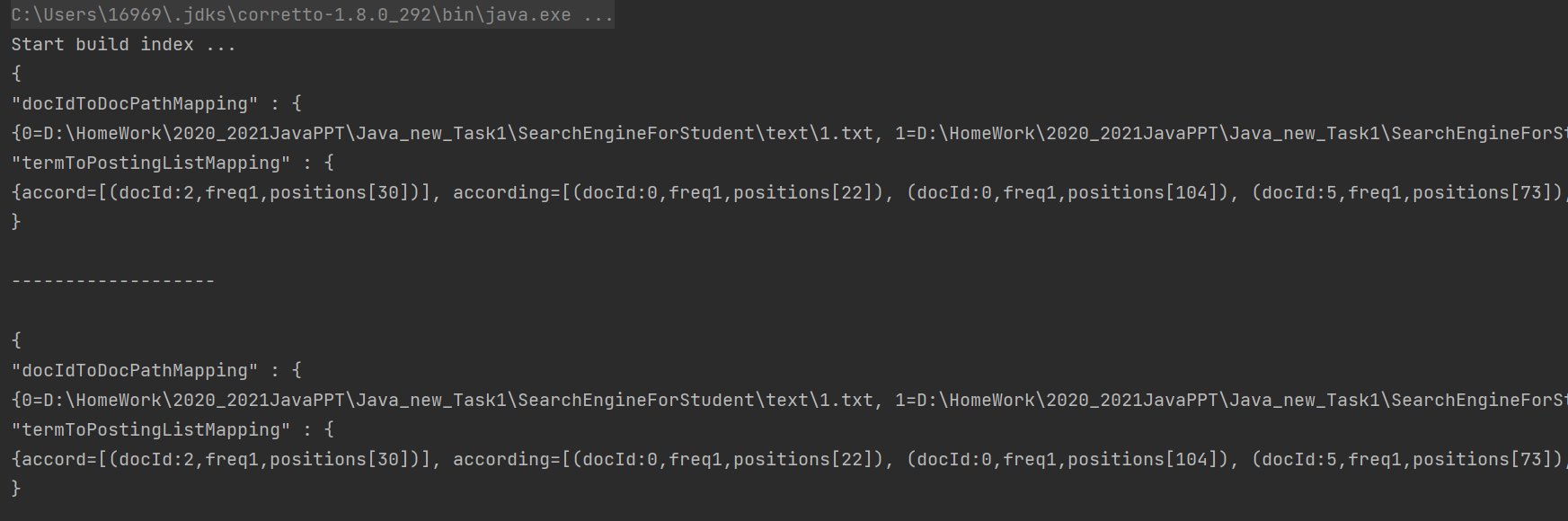


图 7 TestBuildIndex测试结果

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：

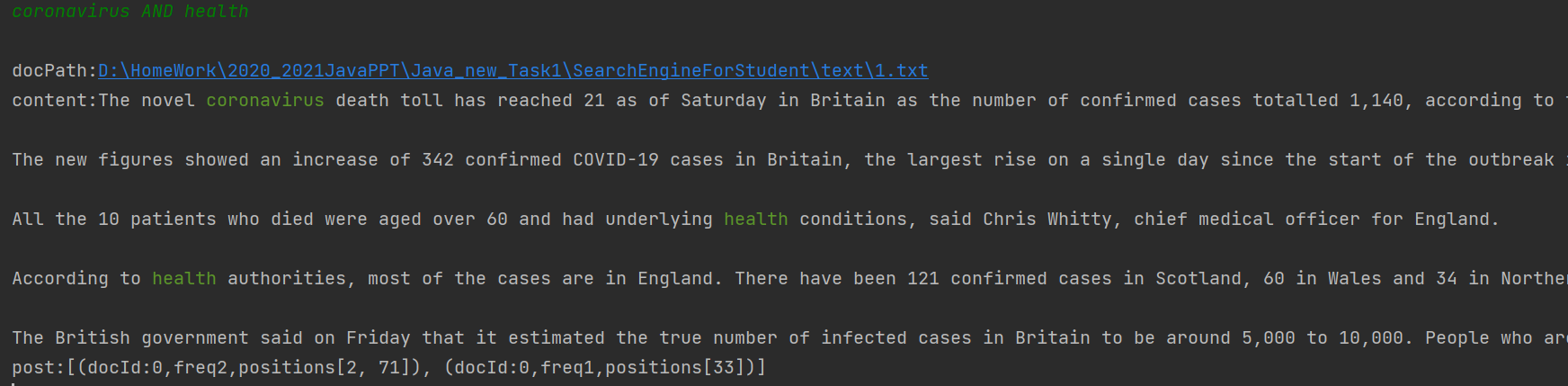


图 9 查询两个单词-同时存在

1. 查询两个单词OR：

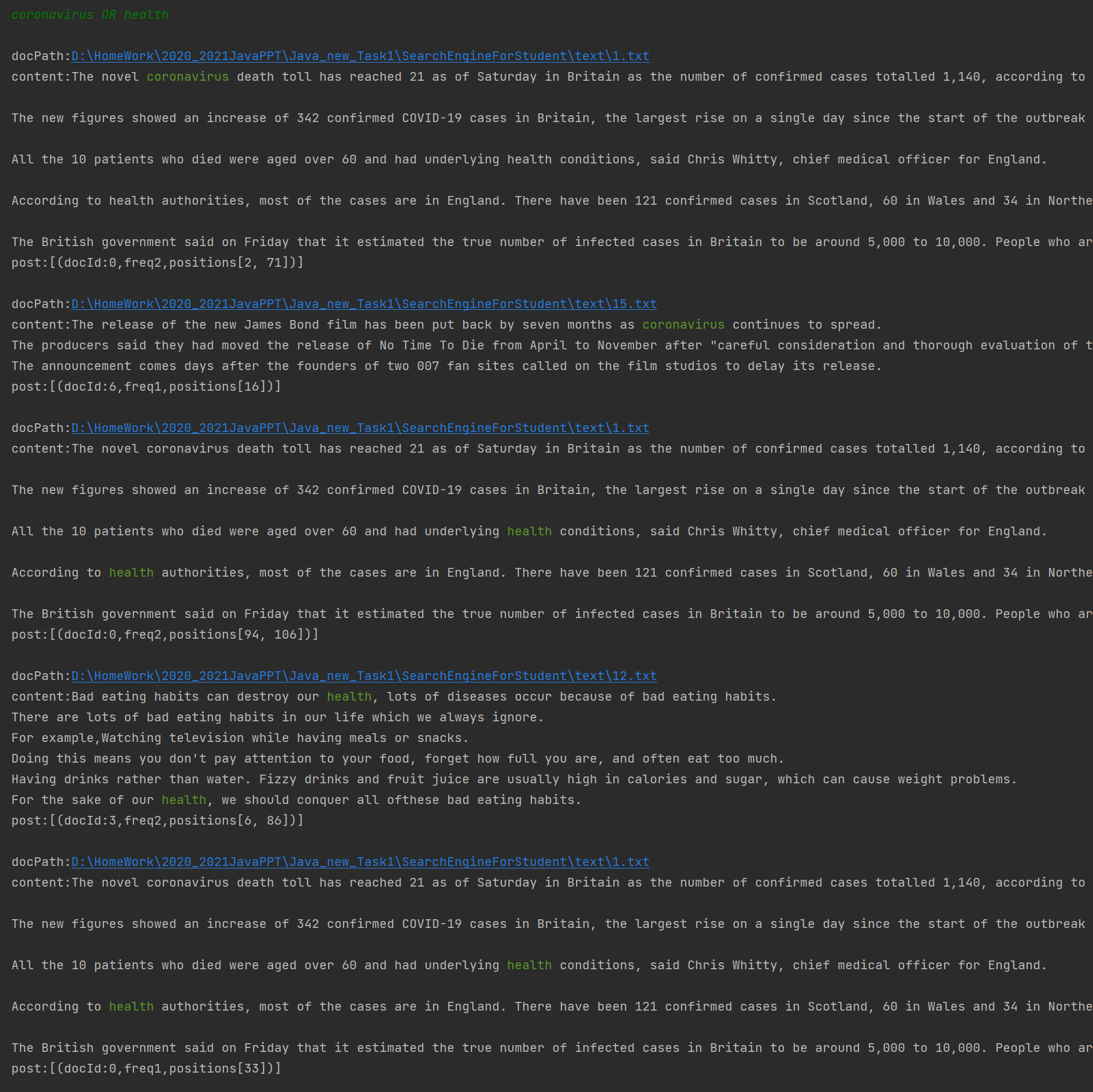


图 10 查询两个单词-存在一个

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**

（1）各种对象的序列化以及反序列化。在序列化过程中充分利用了成员实现的序列化反序列化方法，与在toString方法中调用成员的toString方法有异曲同工之妙。

（2）对于命中结果的得分是按照负分制进行打分，即所出现单词的频率越高，得分反而会越低，这样以来进行排序操作时，便可以直接使用系统的sort函数对文档按照得分进行降序排序，出现单词频率高的文档仍然会排在前面，省去了重新写sort函数的必要。

（3）对于TermTupleScanner三元组输入函数，所使用的方法是用队列读取，而非将文件全部转存到内存然后再分析读取，这样做的好处是减少了内存的开销，只有当队列中的三元组全部读取完毕时，才会继续加入文件中剩余的三元组。

1. **不足和改进的建议**

不足之处就是在查询短语功能方面，有可能查询短语中的某些介词被当作停用词被删除，因此会导致查询的一些问题，暂时还未能解决。

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**

完成了JAVA实验后，十分有收获。首先就是对于abstract类有了更加深刻的理解，关于抽象类的定义以及其子类需要实现的函数功能都能够十分清晰的认知。

其次就是关于接口方面的知识在实验中也得到了充分的发挥，进行了对于文件序列化接口以及sort排序函数接口的实验，使得对于JAVA中的接口更加熟悉。

最后，整个实验虽然难度不大，但是也算是较为大型的项目了，需要对于各个类之间的关系有很好的把控，以及对于在类中已经实现的函数要发挥充分利用，而非再写一些冗余的代码，故整个实验做完后对JAVA编程方面的能力提升很有帮助，收获也是十分丰富的。

1. **源码和说明**
2. **文件清单及其功能说明**

提交的文件目录如图 9 提交文件目录所示

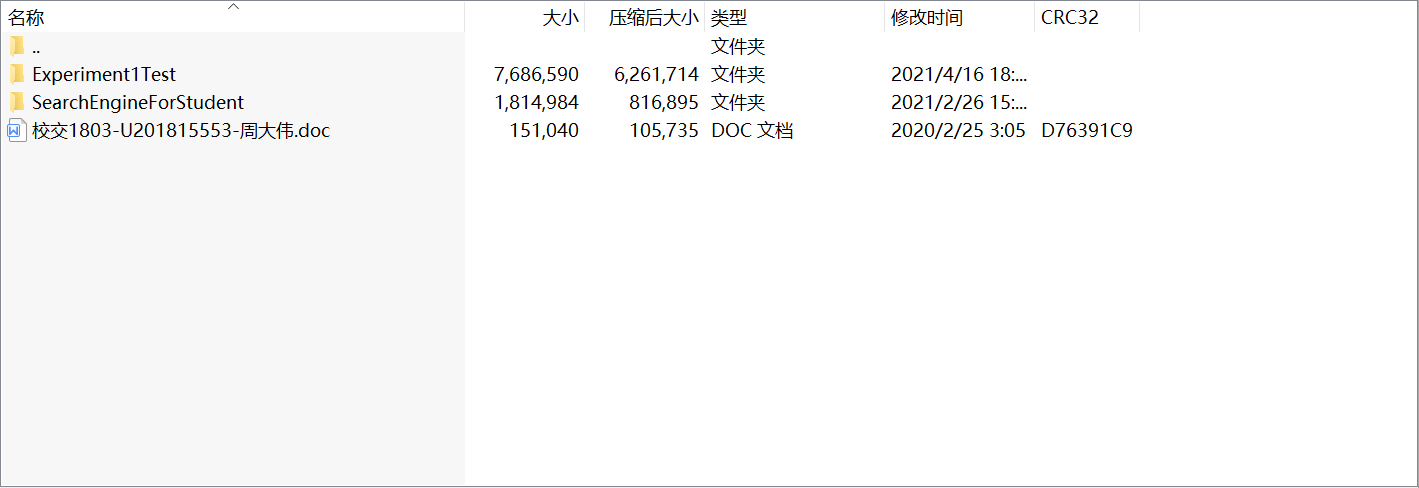


图 12 提交文件目录

1. Experiment1Test是自动测试文件，该文件夹下的test.bat即为自动测试的脚本，运行后能够得到测试的输出，输出目录在该文件夹下的test-output中。
2. SearchEngineForStudent是项目文件，所编的实验的源文件.class全都包含在此文件夹下。
3. 文件包中的.docx文件为实验报告。
4. **用户使用说明书**
5. 打开Experiment1Test目录打开，其中有一个test.bat脚本文件，运行后即在test-output目录中得到自动测试的结果。
6. 将SearchEngineForStudent在IDEA中打开即可看到实验所用到的.class文件，其中在run目录下含有两个可执行的Test类，分别是索引建立和单词查询。在运行TestBuildIndex后即可以建立起测试文件的索引，然后在运行TestSearchIndex，在控制台输入所需要的单词既可以得到对应文件的docId、文件路径和文件内容。
7. **源代码**

见附件中的SearchEngineForStudent文件夹。