華中科技大學

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 ：　CS2203

学　　号 ：　U202215643

姓　　名 ：　王国豪

指导教师 ：　辜希武

2024 年 **5** 月 **31** 日

一、需求分析

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。

2. 需求分析

对于上面题目中提出的要求，在此细化如下：

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）用一个方法将**给定目录**下的所有文件加载出来。

（2）根据（1）中加载出来的结果，使用合适的方法分成一个一个的 Term。

（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）”——这里规定了 3 种不同的过滤方式，实际编程时需要 3 个类分别与之对应。

（4）用一个 map 来存储 **Term** 到与之对应的**倒排索引**的映射。

（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”——利用 Java 内置的 FileSerializable 接口。

（6）完成索引对象的 toString 方法，能够良好地展示索引的内部结构。

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）使用 Java 中内置的 Collection 类，将命中的文档存入其中。

（2）根据单词的命中次数等指标，给每个命中的文档赋分，并以赋分为依据付 Collection 中的命中文档排序。

（3）完成每个命中的 toString 方法，以可视化的形式展现命中的结果情况。

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——这应该在功能 2 的代码基础上进行拓展。

（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——这也应该在功能 2 的代码基础上进行拓展。

（3）完成每个命中的 toString 方法，以可视化的形式展现命中的结果情况。

功能 4：暂未实现。

二、系统设计

1. 概要设计

上面所阐述的功能 1~3 之间存在递进关系，即功能 2 依赖于功能 1 的执行结果，功能 3 依赖于功能 2 的执行结果。

功能 1 首先需要操作者给出需要建立倒排索引的目录；在获得该目录后，会获取目录下

每个文件的文本内容。紧接着，程序将这些文本内容处理成 TermTuple 三元组列表，按照用户设定的过滤方式，舍弃其中不合法的三元组。过滤后的三元组将会被用于建立每个文档的 document 对象，最后再根据每个文档的 document 对象建立倒排索引数据结构；该倒排索引结构可以通过 toString 方法有组织地展示在控制台中。同时，用户可以选择将建立的倒排索引序列化到文件中；倒排索引的构建也可以从预定的文件中反序列化获得。

功能 2 首先需要获得倒排索引数据结构，这一结构可以由功能 1 完成。在获得倒排索引后，用户指定搜索的单词，程序根据单词在倒排索引中查找相应的文档，最终形成命中文档的集合。这一集合将会在控制台被展现。

功能 3 也需要先获得倒排索引数据结构，同样由功能 1 完成。此后，用户需要指定搜索的两个单词与逻辑联结词（AND 或 OR）；程序将复用功能 2 的代码，分别查找两个单词所对应的文档，并根据逻辑连接词将两组文档取交或并，得到最终的命中文档集。该集合也能够在控制台被展现。

上面三个功能的流程图如下所示：

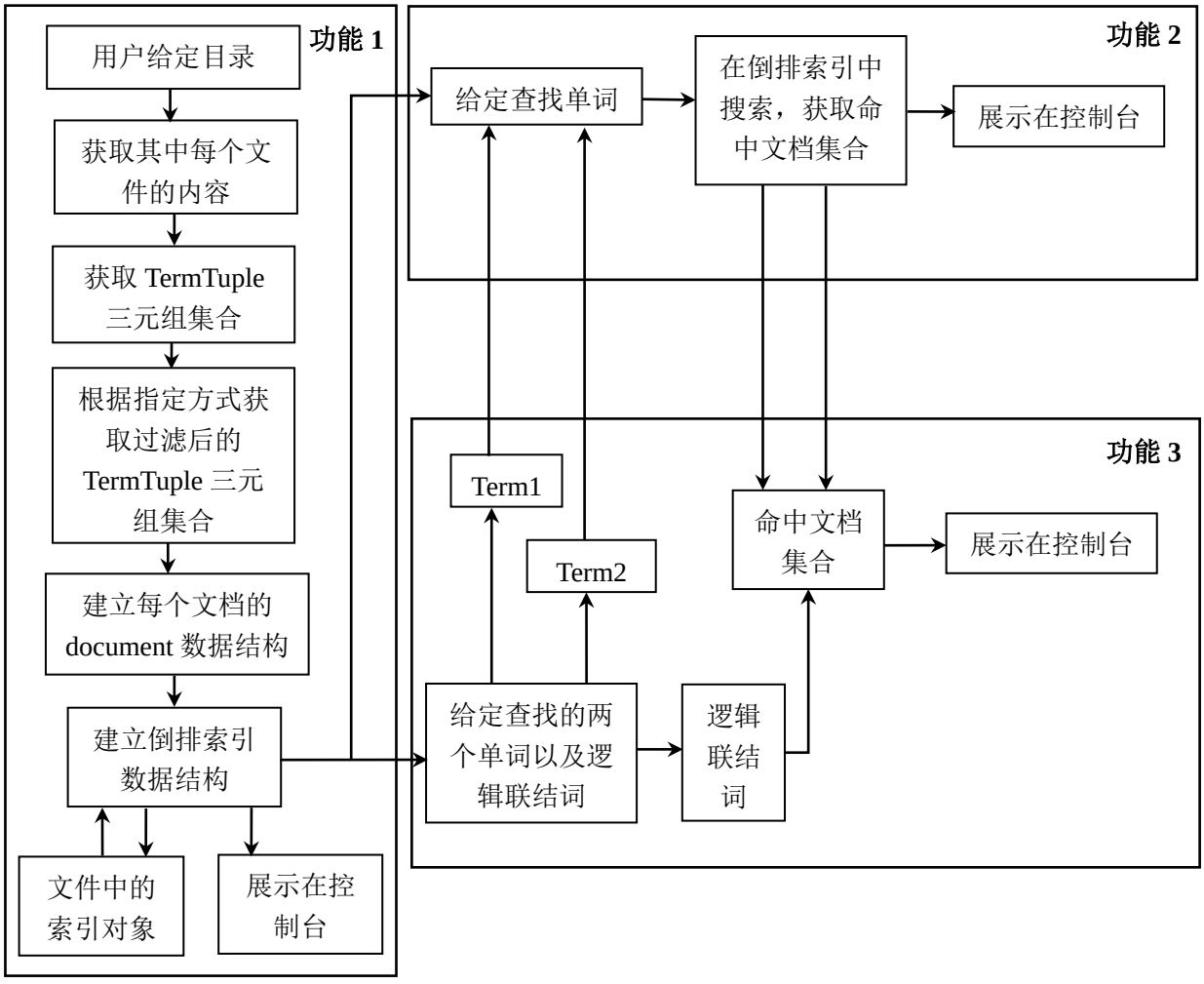


图 1 功能设计流程图

2. 详细设计

上面的功能在工程文件中分为了三个部分：索引部分（index）、解析部分（parse）和询

问部分（query）。下面依次介绍这三个部分的内容。

（1）索引部分（index）

下图为索引部分构建过程中编写的类和接口的 UML 图。

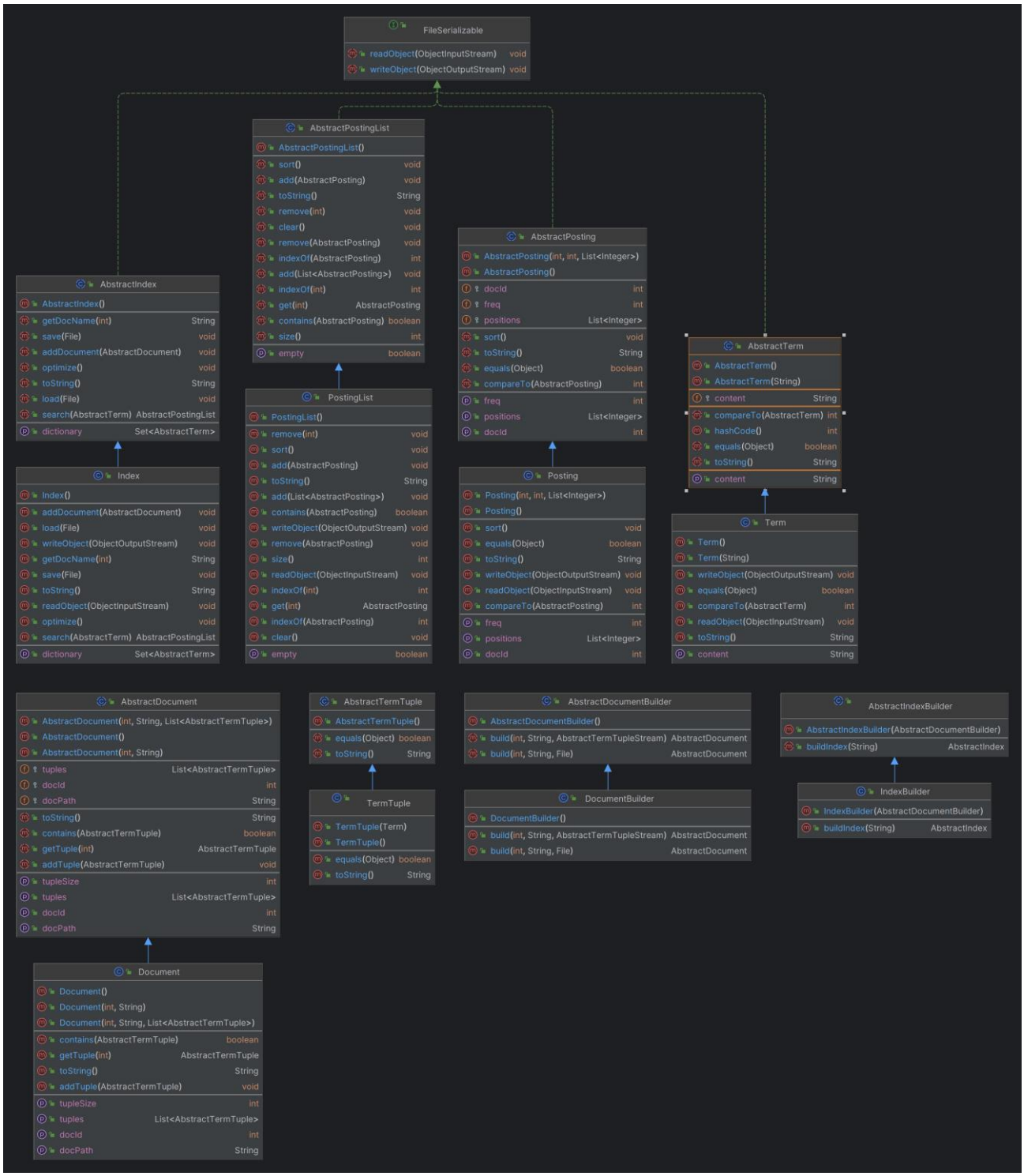


图 2 索引部分 UML 图

这一部分的核心功能是实现功能 1 中“建立每个文档的 document 数据结构”及其之后的功能模块，核心功能是“建立每个文档的 document 数据结构”。对其中各个**具体类**的解释如下：

Term：

可以认为它就是一个单词，其中的重要数据成员是一个 String 串，指向一个单词。

TermTuple：

单词三元组，是对单词在文档中出现这一事件的简单记录。重要数据成员是 term，freq 和 curPos。成员 freq 是常量 1，代表每一次出现；term 和 curPos 分别是出现的单词和单词出现的位置。

Document：

记录每一个文档的具体情况。其中重要数据成员是 docId，docPath 和 tuples；三者分别代表文档的编号、绝对路径和文档所包含的 TermTuple 三元组列表。

Posting：

记录某一个 term 在**某一篇**文档中出现的情况。它包含三个重要数据成员，docId，freq 和 positions；表示的含义是这个单词在编号为 docId 的文档中出现了 freq 次，这 freq 次出现的位置存入了 positions 列表中。

PostingList：

记录某一个 term 在**所有**文档中出现的情况。重要数据成员是一个由 Posting 组成的列表。

Index：

这就是所谓的倒排索引数据结构。重要数据成员为两个 map，其一是 docId 到 docPath 的 map，将所有涉及到的文档的编号与文档的绝对路径联系起来；其二是 term 到 postingList 的 map，记录所有涉及到的单词与单词的 postingList 联系起来。

其中，重写了 toString 方法，调用这一方法的过程即为功能 1 中“展示在控制台”的模块。toString 方法通过复用 PostingList 和 Term 的 toString 方法，良好地展示了倒排索引的结构。

同时，这个类实现了 FileSerializable 接口，重写了接口中的两个抽象方法 writeObject 和 readObject，从而实现倒排索引数据结构（内存）与文件中的索引对象（外存）之间的交换。writeObject 方法有一个 ObjectOutputStream 形参 out，重写该方法时通过调用该类的 writeInt，writeChars，writeObject 等多种实例方法来向 out 中写入数据；readObject 方法有一个 ObjectInputStream 形参 in，重写该方法时通过调用该类的 readInt，readChars，readObject 等多种实例方法来从 in 中读取数据。

DocumentBuilder：用于构造 document 对象的类。有两个重载的 build 方法，两个方法均接收 docId，docPath 参数，第三个参数可以用来设置过滤方式；方法利用解析部分（parse）中的类的方法来辅助 document 构造。该类完成功能 1 中的“建立每个文档的 document 数据结构”，这之前的功能交由解析部分（parse）完成。

IndexBuilder：用于构造 index 对象的类。重要数据成员是 DocumentBuilder 类的 docBuilder，重要实例方法是 buildIndex。该方法接收一个目录路径，通过解析该路径下所有的文件及其路径，逐一利用 docuBuilder 构造 document 对象，构造出来的 document 对象加入到 index 中，并最终返回一个 index 对象。

（2）解析部分（parse）

下图为解析部分构建过程中编写的类的 UML 图。

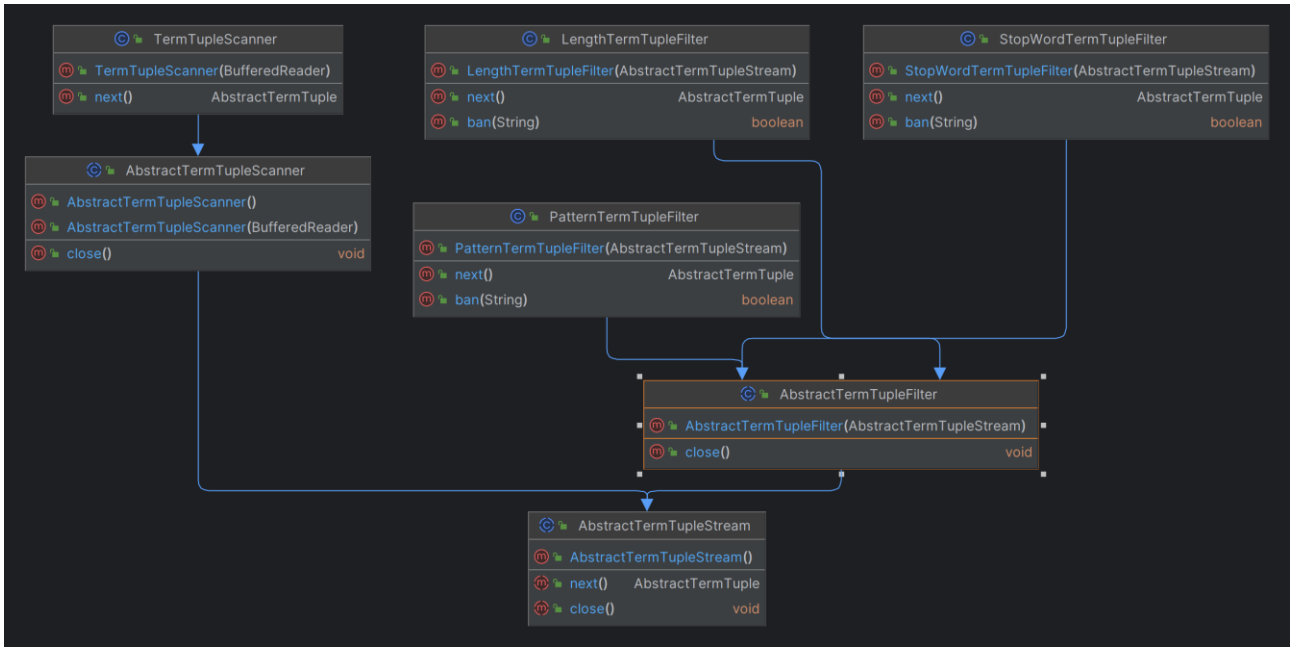


图 3 解析部分 UML 图

这一部分的核心功能是实现功能 1 中“根据指定方式获取过滤后的 TermTuple 三元组集合”及其之前的功能模块，核心功能是“根据指定方式获取过滤后的 TermTuple 三元组集合”。对其中各个**具体类**的解释如下：

TermTupleScanner：

从一个由文件对象产生的 BufferReader 实例对象 input 中扫描获取**所有** TermTuple。自定义了数据成员 buffer，它是 String 的列表，用于缓存从文件中读取的数据；重要方法是 next，每次调用 next 即返回 input 中的一个 TermTuple 三元组，实现功能 1 中“获取 TermTuple 三元组集合”模块。该模块之间的模块通过调用 java 库方法实现。

调用 next 的流程是：如果 buffer 不为空，则从 buffer 中取出第一个 String 对象，打包成 TermTuple 返回；如果 buffer 为空，从 input 中读取一行的字符串，将其按照切分规则（STRING_SPLITTER_REGEX）切分为多个 String，存入 buffer 中，随后进行 buffer 不为空时的操作。当没有下一个 TermTuple 时，返回 null。流程图如下所示：

图 4 TermTupleScanner 的 next 方法工作流程图

StopWordTermTupleFilter：

对 TermTuple 三元组集合进行禁止词的过滤。禁止词以 String 数组的形式存放在 StopWords 类的静态数据成员 STOP_WORDS 中。其中的 ban 方法传入一个 String，根据 STOP_WORDS 数组判断这个串是否被禁止（返回 boolean 值）。重要数据成员是一个引用类型为 AbstractTermTupleStream 的 input 变量。

重写了 next 方法，根据 input 的 next 方法提供的 TermTuple 三元组，基于 ban 方法实现过滤。ban 方法返回 true 时丢弃这个三元组，否则保留。

PatternTermTupleFilter：

对 TermTuple 三元组集合进行正则表达式的过滤。正则表达式以静态 String 对象 TERM_FILTER_PATTERN 的方式存放在 Config 类中。其中的 ban 方法传入一个 String，判断它是否含有非字母字符并返回（返回 boolean 值）。重要数据成员是一个引用类型为 AbstractTermTupleStream 的 input 变量。

重写了 next 方法，根据 input 方法提供的 TermTuple 三元组，基于 ban 方法实现过滤。ban 方法返回 true 时丢弃这个三元组，否则保留。

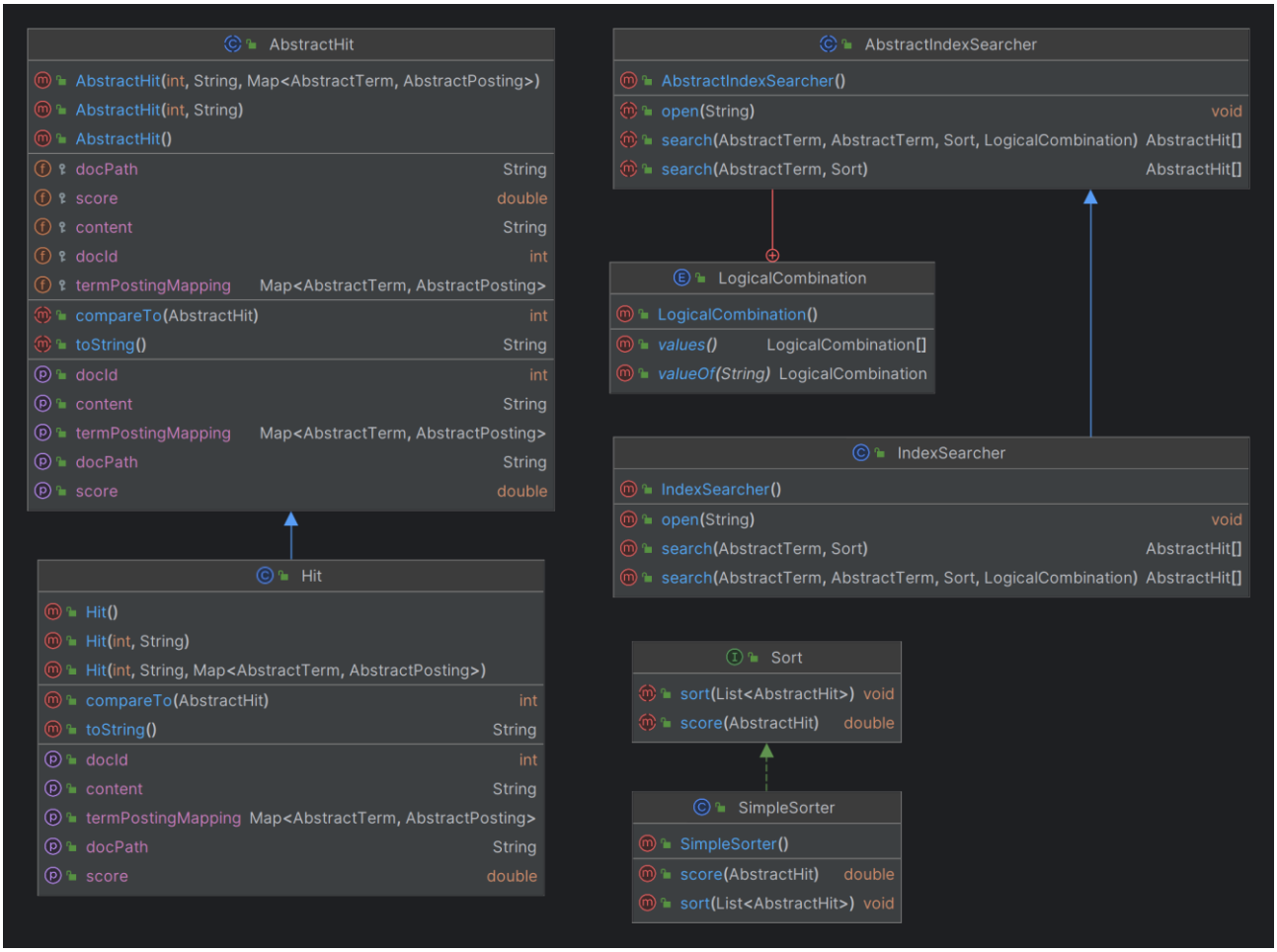


图 5 询问部分 UML 图

LengthTermTupleFilter：

对 TermTuple 三元组集合进行单词长度的过滤。单词的长度下限和单词的长度上限分别是

TERM_FILTER_MINLENGTH 和 TERM_FILTER_MAXLENGTH，以静态变量的方式存放在 Config 类中。其中的 ban 方法传入一个 String，判断它的长度是否在规定的范围内（返回 boolean 值）。重要数据成员是一个引用类型为 AbstractTermTupleStream 的 input 变量。

重写了 next 方法，根据 input 的 next 方法提供的 TermTuple 三元组，基于 ban 方法实现过滤。ban 方法返回 true 时丢弃这个三元组，否则保留。

（3）询问部分（query）

询问部分构建过程中编写的类和接口的 UML 图如图 5 所示。

该部分实现功能 2 和功能 3 的全部模块，核心功能是“在倒排索引中搜索，获取命中文档集合”。对其中各个具体类的解释如下：

Hit：

用于保存搜索命中的**单篇文档**的详细信息，搜索结果的载体。重要数据成员 docId 和 docPath 分别描述文档的编号和绝对路径；content 是文章的内容（为了提供更友好的 UI 接口，其中的命中词进行了高亮设置）；score 是根据这篇文章的搜索命中词的情况（个数等）赋予的分数值；还有一个 map 描述了所有命中的 term 到响应的 posting 的情况。

上述的情况都通过重写的 toString 方法，以功能 2 和 3 的“展示在控制台”模块展现出来。

SimpleSorter：

这个类实现了 Sorter 接口，实现了其中的两个方法：其一是方法 score，对给定的 Hit 对象的命中情况打分，SimpleSorter 直接令命中次数就是 Hit 对象的分值；其二是方法 sort，对给定的 Hit 列表，根据其 score 值进行降序排序。

IndexSearcher：

用于对给定的搜索词，获取搜索结果，是该部分中的核心类。其中提供的 open 方法，能够根据功能 1 中“文件中的索引对象”反序列化出倒排索引数据结构，存储在重要数据成员 index 中。重要方法还包括两个重载的 search 方法，分别用于一个单词、两个单词的搜索。

对于一个单词搜索的 search 方法，只需提供搜索单词 queryTerm 和一个实现了 Sort 接口的实例即可。方法会在 index 中搜索 queryTerm 得到一组 Hit 结果，并根据 Sort 提供的打分方法 score 和排序方法 sort 对 Hit 结果进行赋分和排序。

对于两个单词搜索的 search 方法，不仅需要提供两个单词 queryTerm1、queryTerm2，一个实现了 Sort 接口的实例，还需要提供逻辑联结词。逻辑联结词是 IndexSearcher 的抽象父类 AbstractIndexSearcher 中的静态枚举成员，包含 AND 和 OR，分别表示两个单词均要出现在某篇文档中、两个单词至少有一个出现在某篇文档中。

三、软件开发

在 window11 下，使用 IntelliJ IDEA Professional Edition 2023.3.4 x64 作为集成开发环境，使用 IDEA 提供的相关工具直接编译成字节码文件。

四、软件测试

1. 自动测试

配置 test.bat 文件中 JDK 的路径后，运行。结果如下：

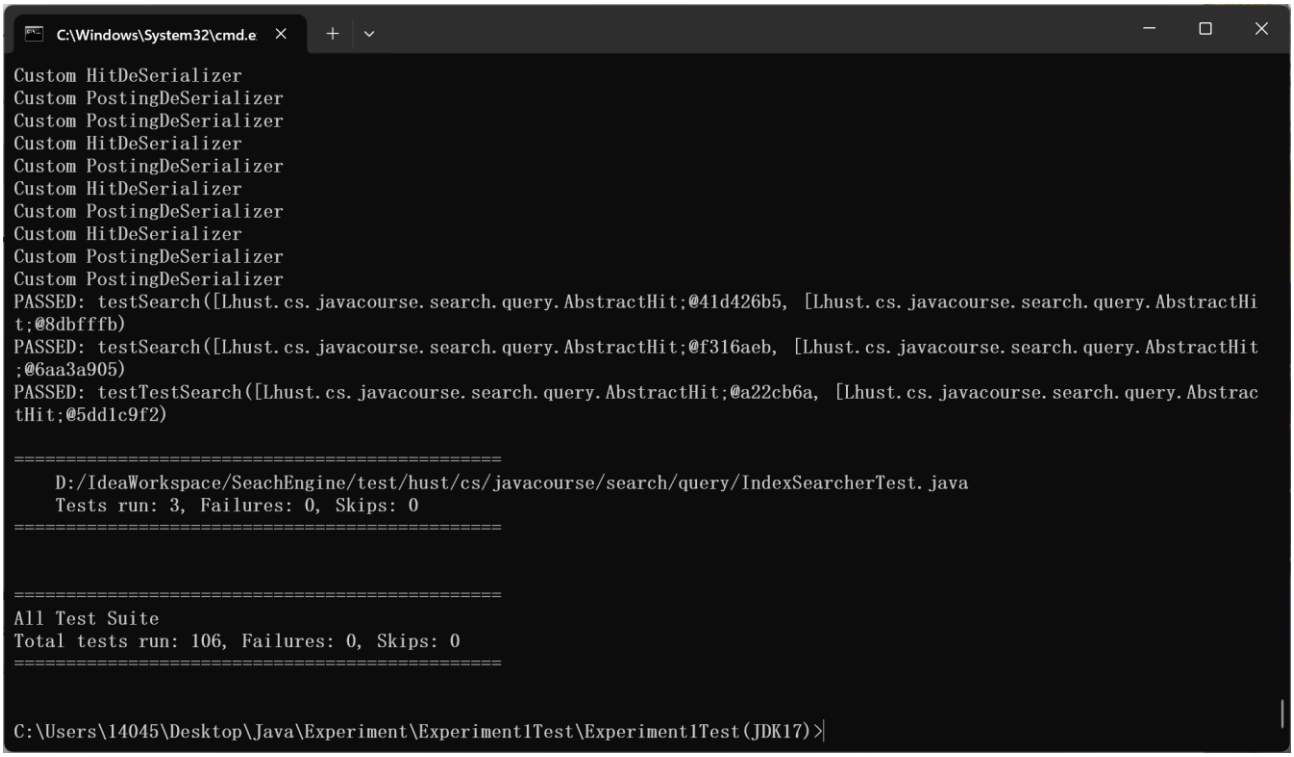
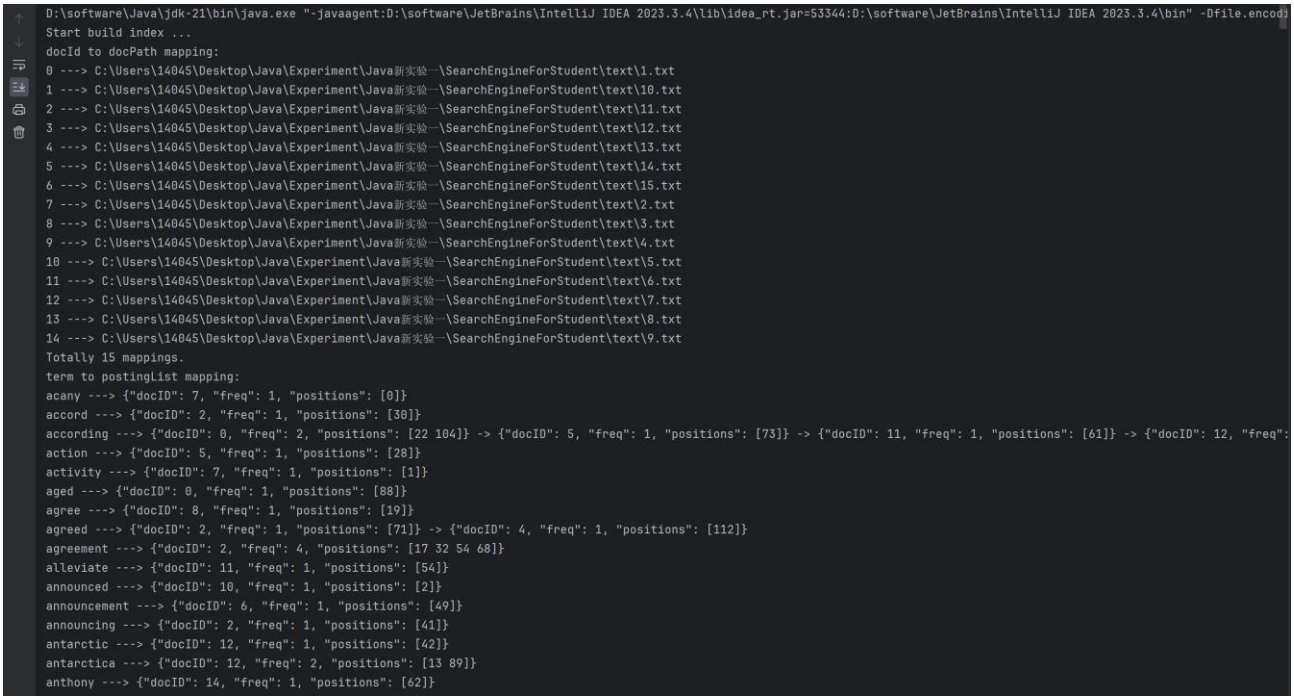


图 6 自动测试运行结果

可以看出，测试用例共 106 个，完成 106 个，失败 0 个，跳过 0 个，结果正确。

2. 功能 1 测试——建立倒排索引

将真实测试数据集中的 15 个文本文件复制到工程文件的 text 目录下，按照要求编写 TestBuildIndex 类，实现其 main 方法。运行结果如下：



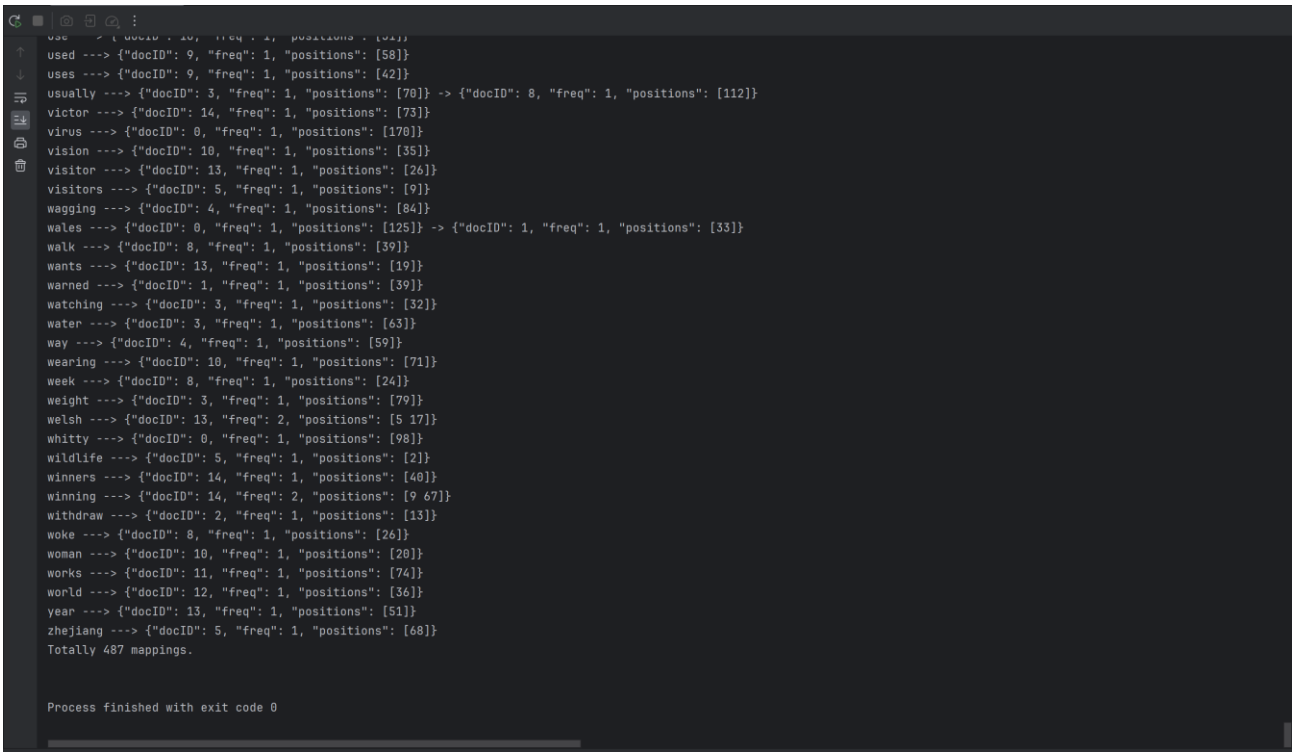


图 7 功能 1 测试结果

图 7 是对运行结果首尾分别的截图。可以看出刚好有 15 个 docId 到 docPath 的 mapping，并且建立了 487 个 term 到 postingList 的 mapping，说明功能 1 建立倒排索引是成功的；而从倒排索引序列化得到的文件 index.dat 中反序列化得到的结果与序列化之前的结果相同，说明其序列化、反序列化功能也是完备的。

3. 功能 2 测试——搜索一个单词

按照要求编写 TestSearchIndex 类，实现其 main 方法。搜索一个“coronavirus”单词，结果如下：

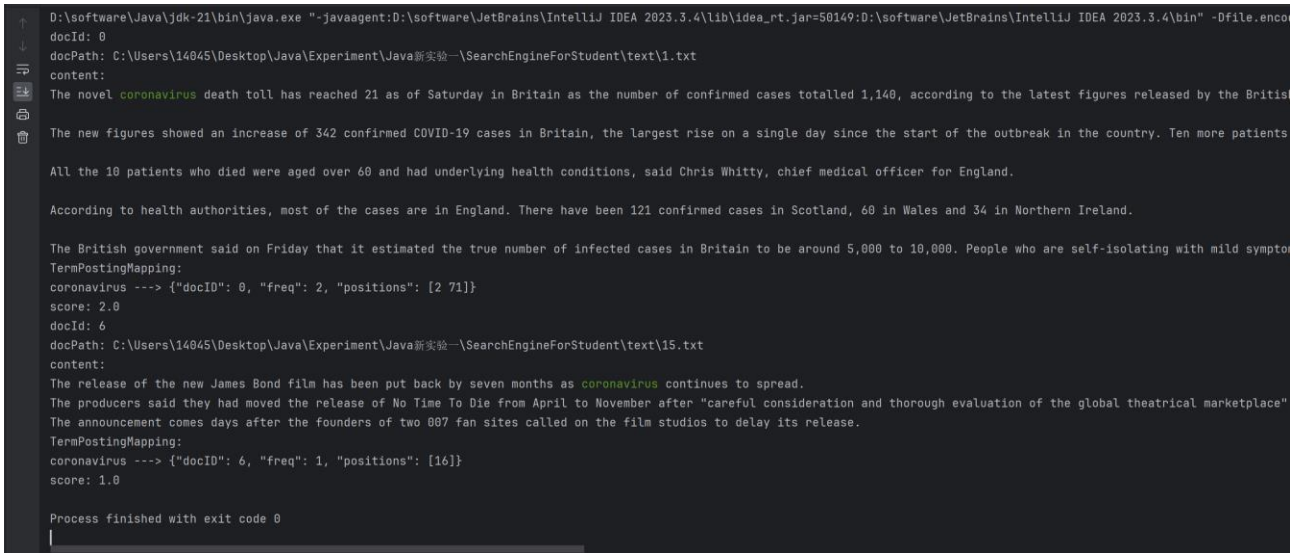


图 8 功能 2 测试结果

能够正确地找到单词出现的文档，按照设定的功能给命中结果正确地打分与排序。

4. 功能 3 测试——搜索两个单词

编写一个 TestTwoWordsSearchIndex 类，实现两个单词的搜索。不同于功能 2，功能 3 搜索两个单词的指令在命令行给出。下面是分别进行“与”“或”搜索的运行结果：

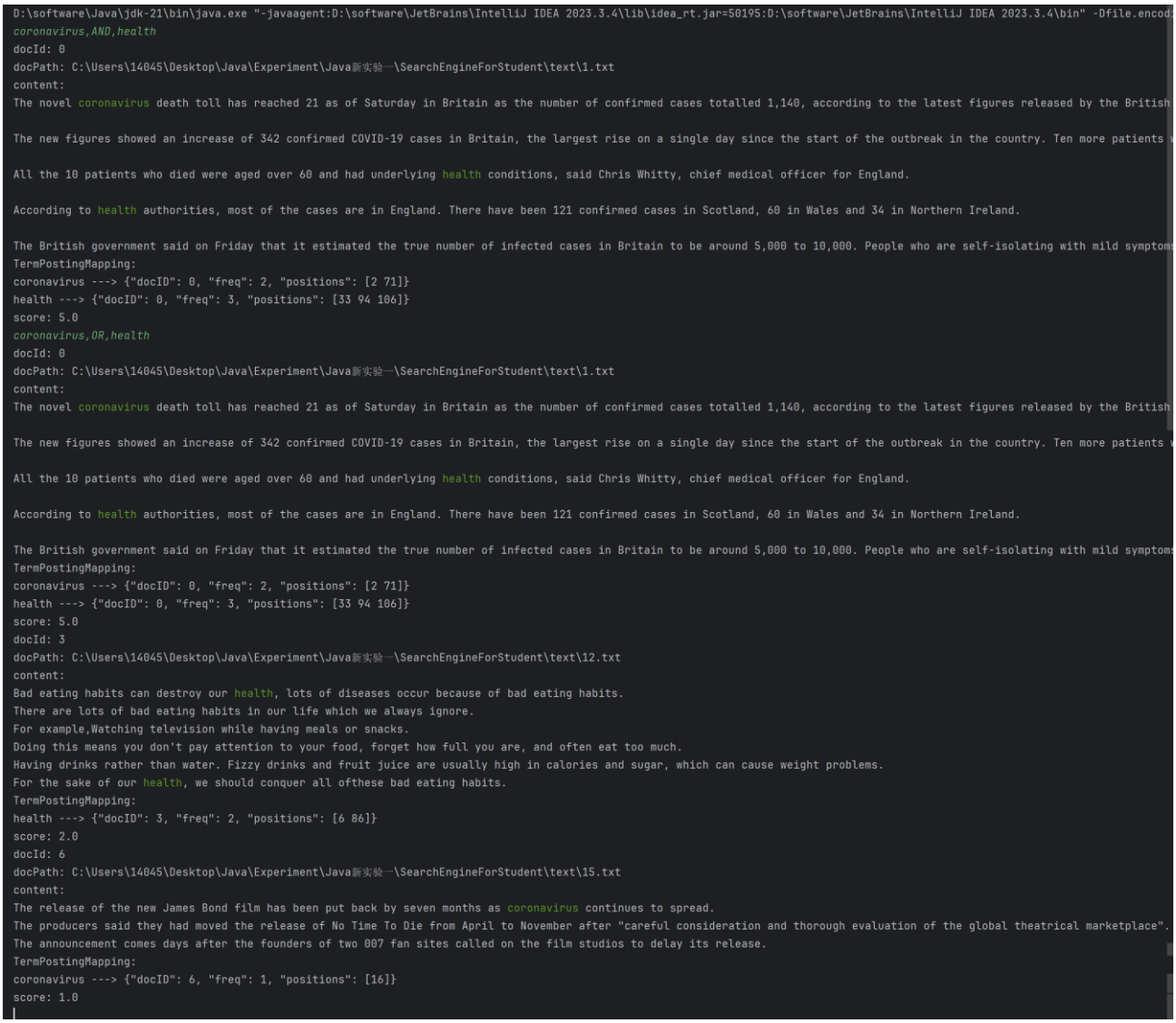


图 9 功能 3 测试结果

“coronavirus”与“health”的搜索结果是 docId 为 0 的文档，这个文档中同时出现了两个单词；“coronavirus”或“health”的搜索结果是 docId 为 0、3 和 6 的文档，这些文档中至少出现了其中的一个单词。而且，对于 Hit 的打分和排序功能也是正确的，按照 score 的值降序输出。

五、特点与不足

1. 技术特点

（1）使用匿名类减少代码量。在某些情况下，我们要写的类只需要被用到 1 次，可以使用匿名类。这一点主要在各种 sort 方法中体现，如 postingList 的 sort 方法，posting 的 sort 方法和 SimpleSorter 的 sort 方法。通过给 List 的 sort 方法传入实现了 Comparator 接口的匿名类，直接传入比较的方法。

（2）命中单词高亮显示。如图 9 所示，命中的单词都以墨绿色的形式在命令行中高亮显示，使用者可以清晰地看见每篇文章的命中情况。

（3）TermTupleScanner 的数据成员 buffer。通过建立 buffer 列表，相当于在外存于内存中建立了一个缓冲区；一个文件的所有内容不是一次性全部加载到内存中，而是一行一行地加载到内存中，当这一行的所有单词用完了再加载下一行。这样较好地减少了内存开销。

2. 不足和改进的建议

（1）由于时间的原因，功能 4 尚未完成。

（2）工程文件中提供了工具类的包 util，但是在实际编程时没有很好地运用到这些工具类——如 FileUtil 的 write 方法（1 usage）。

六、过程和体会

1. 遇到的主要问题和解决方法

（1）工程项目太大，不知道各个类要做什么。

主要是对工程没有一个整体的把握，一些简单的类按照说明确实可以容易地写出其代码，比如 getter 和 setter 这些不需要建立在理解之上就能完成代码；但是代码写到后面，遇到的类及其功能说明也越来越抽象了，迫使我停下来好好地分清楚每个类、类里面的方法究竟是用来做什么的。经过很长时间的仔细研究，也终于弄懂了。

（2）写了代码，但实际运行没有创建 index.dat 文件。

这个一开始是在 IDEA 本地运行出现的问题，弄了半天发现是路径写错了，固然就不会创建 index.dat 文件。后来在自动测试的时候抛出了一场，说找不到这个文件，想了很久也不知道为什么。后来经过搜索才发现是对应的工程目录下没有 index 文件夹，固然就没有办法在这个文件夹下面创建 index.dat 文件。解决办法是先用 mkdir 方法创建 index 文件夹，再创建文件。

（3）构建索引时多出了很多单词，并且排列顺序也不一样。

仔细对比发现是大小写的问题，询问老师后才发现 Config 类里有一个 IGNORE_CASE，需要我们忽略大小写（全部转换为小写）。

（4）从文件中读取索引总是抛出异常。

仔细看了一下网上对于 ObjectOutputStream 类中各种 write 的用法，发现最后都会有一个 flush 方法。原来是我没有写 flush，导致数据没有刷到外存中。加上这个 flush 就没问题了。

（5）一些 NullPointerException 常客。

根据 IDEA 中抛出异常的代码源追溯到异常的抛出点，从而发现问题、解决问题。

2. 课程设计的体会

这个实验是我第一次接触比较大的 Java 工程项目，一开始不知道要干什么，到后面逐渐理解了每个类的作用，形成了系统设计的观念和素养。同时，通过 parse 中各种（三个过滤器）

AbstractTermTupleFilter 的子类，深刻地体会到了**装饰者模式**的应用。

同时，这个实验的完成少说也花了一个多星期的时间，对于耐心和能力都是极大的考验。

七、源码和说明

1. 文件清单及其功能说明

提交的资料包括：

- 源码 SearchEngineForStudent 是工程文件的源代码文件。
- 文件夹 Experiment1Test(JDK17)是自动测试文件包，里面包含 betest 文件夹，可以使用教师提供的 test.bat 进行自动测试。(测试文件路径不能包含中文)
- 实验报告 word 版本
- 实验报告 pdf 版本

2. 用户使用说明书

无需安装，只需用 IDEA 打开工程文件根目录。根目录下的 src/hust/cs/javacourse/search/run 中包含三个类 TestBuildIndex， TestSearchIndex 和 TestTwoWordsSearchIndex 三个类，作用分别是建立索引、搜索一个单词、搜索两个单词。

倒排索引建立的源目录是根目录下的 text 文件夹。可以在这个文件夹里存放需要构建倒排索引的文本文件。按照 TestBuildIndex 中现有的源代码，你每创建一次倒排索引，响应的数据结构均会存入到根目录下的 index/index.dat 中。**后一次的结果会覆盖前一次的内容。**

在 TestSearchIndex 中，你可以看到被搜索的单词是 coronavirus。如果想要搜索别的单词，更改它。

在 TestTwoWordsSearchIndex 中，你需要与命令行交互以进行搜索。输入”term1”可以进行对 term1 的单个单词搜索；输入”term1,AND,term2”可以进行对两个单词的与搜索；输入”term1,OR,term2”可以进行对两个单词的或搜索；输入”a”结束程序。

3. 源代码

见附件中的 SearchEngineForStudent 文件夹。