

华中科技大学

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：计卓 2101

学 号：U202112071

姓 名：王彬

指导教师：纪俊文

2023 年 4 月 27 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 and 接口实现类则不作任何限定。

(2) 自己实现的抽象类子类 and 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

对于功能 1，我们对指定目录下的一批.txt 格式的文本文件扫描并在内存里建立倒排索引。其中我们需要实现：

- (1) 对于每一个单词，我们构建包含此单词信息的三元组 (**Termtuple**) 的单词文本、出现位置和出现的次数；
- (2) 对指定目录下的每一文本文件，建立文档 (**document**) 结构，存储每一文档的文档编号、文档路径及文档内容；
- (3) 对倒排索引中每一个单词对应的文档列表给予实现，建立单词倒排索引列表 (**PostingList**) 结构，其中每个元素均为 **Posting** 结构成员，分别具有文档编号、出现频率和出现的位置列表这三个数据成员；
- (4) 实现内存中的整个倒排索引结构 (**Index**)，其中包括文档编号和文档绝对路径的映射关系，以及每个单词和其对应的倒排索引列表 (**PostingList**) 的对应关系；
- (5) 建立各个数据结构的构建方法和初始化方式，包括 **DocumentBuilder**，**IndexBuilder** 等等。
- (6) 在 **parser** 包中，我们建构基于装饰者模式的一系列过滤器 **TermTupleFilter**，并通过联合 **TermTupleScanner** 对文档实现解析。

对于功能 2，我们基于已经构建好的索引，实现单个搜索关键词的全文搜索。其中我们需要实现：

- (1) 我们通过 **Hit** 类的编写，实现对命中结果的显示和排序。其中需要存放文档编号、文档绝对路径、文档内容和命中结果得分，用以判断搜索结果在返回列表中的顺序；
- (2) 实现进行全文检索的类 (**IndexSearcher**)，其中分别装载有打开文档和实现搜索的方法。

对于功能 3，基于构建好的索引，实现二个搜索关键词的全文检索。其中我们需要实现：

- (1) 支持对两个关键词的与、或查询。我们首先对两个关键词分别查询，对于这两个关键词的查询结果的连接关系 (**AND/OR**) 进行结果的连接、合并；
- (2) 对查询结果根据文档的得分进行排序，使得查询结果能够反映查询情况。

二、系统设计

1. 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 总体设计流程图所示。

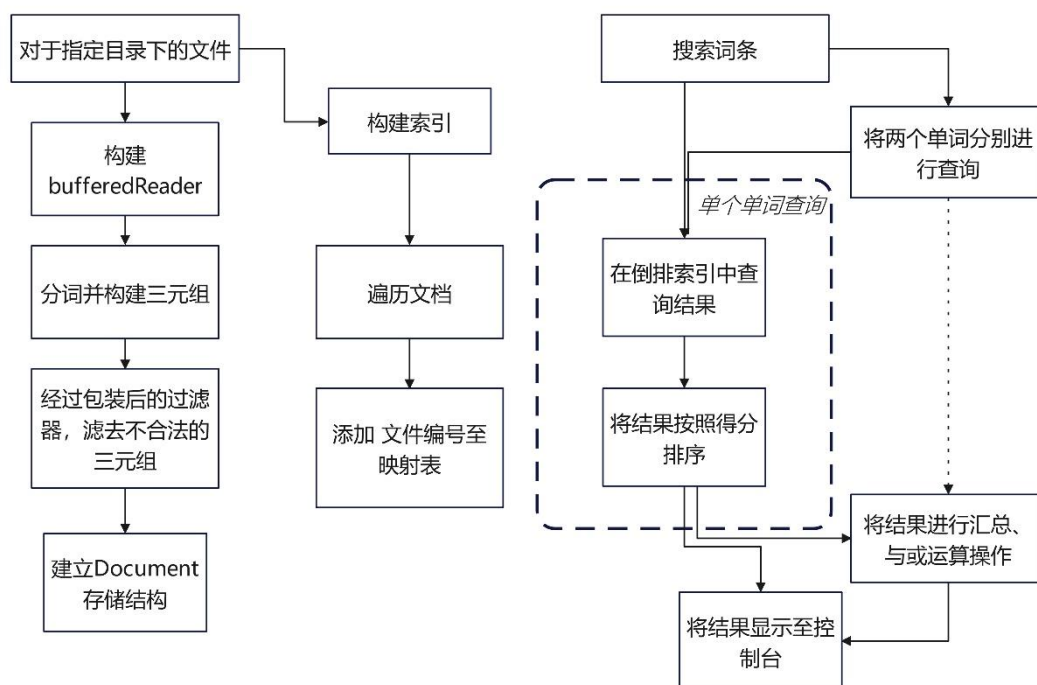


图 1 总体设计流程图

主体流程分为三个部分。工程将流程分为三个包(index, parse, query)，分别编写下述内容：

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，同时使用过滤器对单词进行筛选，而后将单词存储为 TermTuple 格式，并以此来构建 Document 对象。

2. 对每一个 Document 建立索引，并将其分为两个部分：对于文档自身的索引以及对于词条的索引。我们将通过不断地读取 Document 中的 TermTuple 来构建针对每一个单词的 Postinglist。

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。

2. 详细设计

（一）相关数据结构

构造索引所需要的类及其方法的 UML 图如图 2 倒排索引结构 UML 图所示。

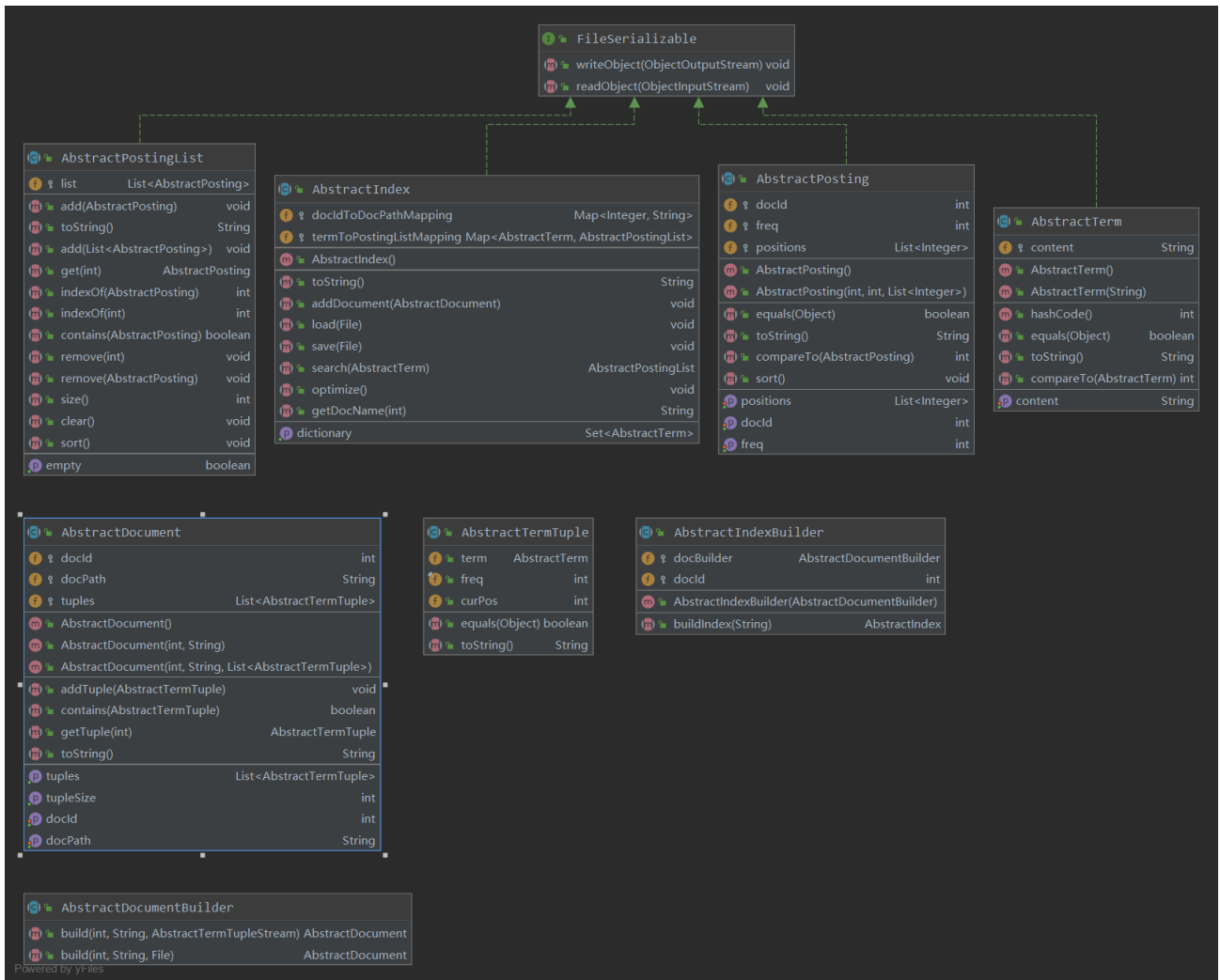


图 2 倒排索引结构 UML 图

(1) Term 类

其具体子类实例为一个单词 term，本质上是对一个字符串进行包装。

(2) Posting 类

其具体子类实例为倒排索引里的一个 Posting，其中包含三个数据成员 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

(3) PostingList 类

其具体子类实例为倒排索引里一个单词对应的 PostingList，其中包含一个 List<AbstractPosting>类型的数据成员存放这个 PostingList 包含的多个 Posting

(4) Index 类

其具体子类实例为内存中的整个倒排索引结构，其中包含了二个数据成员

- i. 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（开始构建索引时我们只有每个文档的绝对路径，因此内部需要维护一个文档 Id 的计数器，每将一个文档加入到倒排索

引，文档 Id 计数器加 1)

- ii. **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。需要特别说明的是这里没有必要用专门的数据结构来存放字典内容，我们直接通过 `termToPostingListMapping.keySet()` 方法就可以得到字典。

(二) 过滤器

在通过 `bufferReader` 进行文本中的单词读取后，我们获得了构建好的三元组 `TermTuple`。而后我们需要采用“装饰者模式”对其进行过滤，将不合条件的三元组排除在外。Parse 包中，过滤器的 UML 图如图 3 过滤器 UML 图所示。

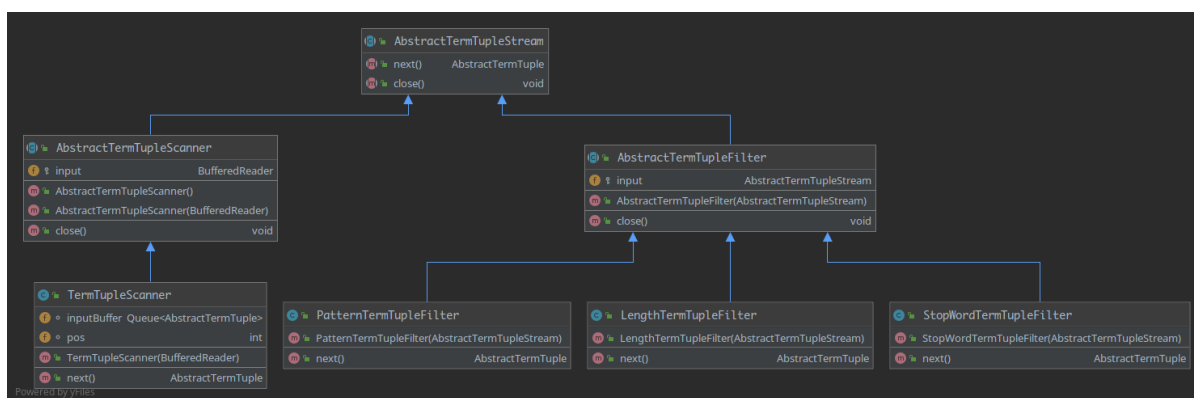


图 3 过滤器 UML 图

我们在原始文档的基础上，构建 `bufferedReader` 对象实现文本的读取，并构建 `TermTupleScanner` 子类对象，再通过调用 `next()` 方法以获得下一组三元组对象，直至返回 `null` 为止。其中包含有下列类：

(1) `TermTupleScanner` 类

我们将解析文档过程产生许多单词的三元组，即将一个文档看出三元组流，其中定义两个基于流的抽象获得方法 `next()` 和关闭方法 `close()`。

通过 `AbstractTermTupleScanner` 类，这个类定义了如下数据成员和构造函数：

- a) `protected BufferedReader input` ;
- b) `public AbstractTermTupleScanner(BufferedReader input)`

即通过 `java.io.BufferedReader` 关联到文件，`BufferedReader` 的 `readLine` 方法可以让我们一次从文本文件里读取一行，当读取到文件末尾返回 `null`。

(2) `StopWordTermTupleFilter` 类

通过 `StopWords` 类中的 `STOP_WORDS` 字段来判断是否是停用词，使得停用词被过滤去，提高查询与建立索引的效率

(3) `LengthTermTupleFilter` 类

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，保证所有合

法的三元组的单词满足规定

(4) PatternTermTupleFilter 类

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。

(5) TermTupleFilter 类

通过 AbstractTermTupleFilter 子类对象。同时采用设计模式“装饰者模式”来设计代码，去除文本文档里停用词、非英文单词如数字、过长或过短的单词对应的三元组。

(三) 查询过程

查询过程（query 包）逐一需要实现三个类，分别用于查询命中、存储索引和对结果进行排序。查询的相关 UML 图如图 4 查询过程 UML 图所示。

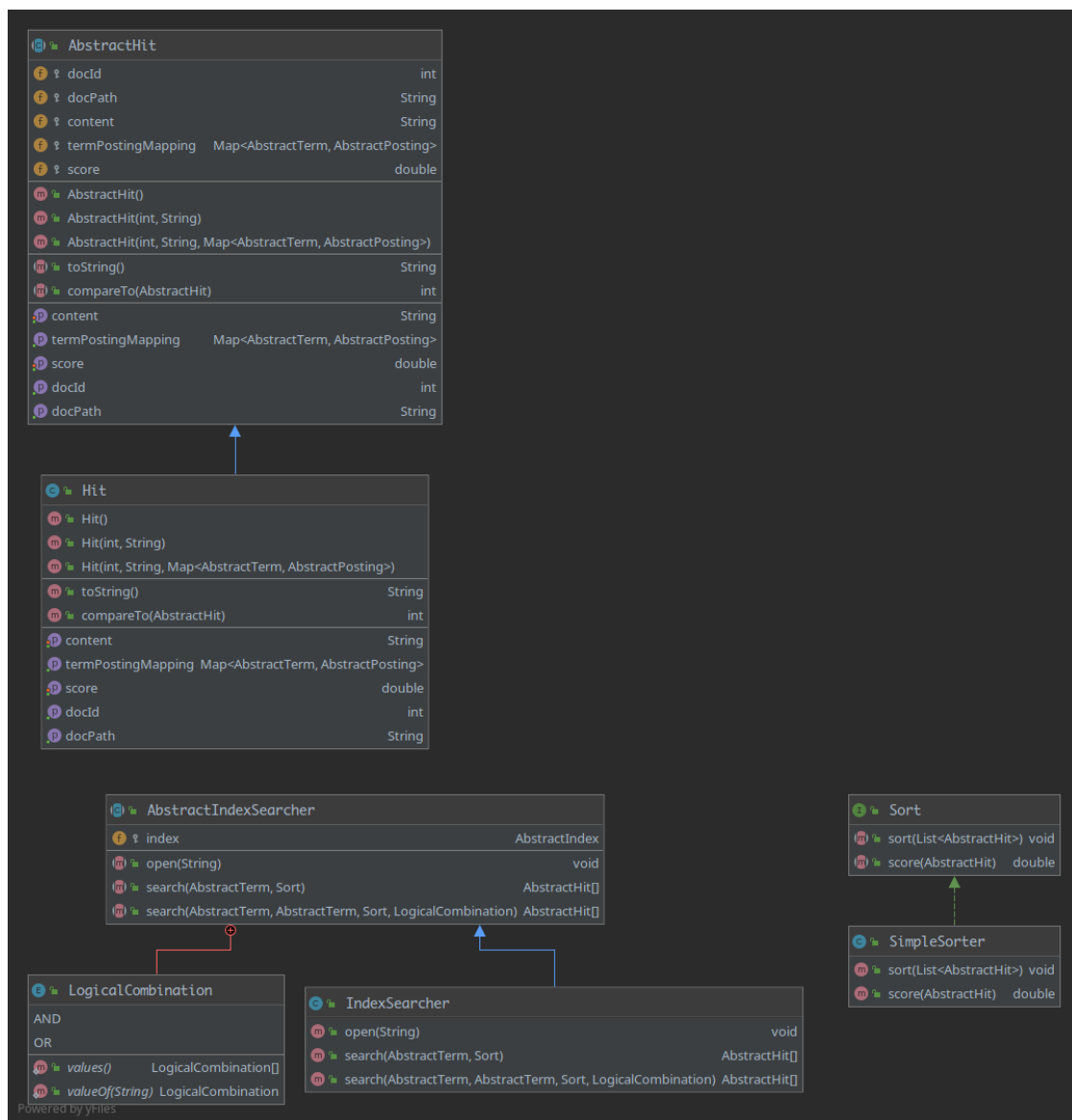


图 4 查询过程 UML 图

(1) Hit 类

Hit 是搜索命中结果的类。命中的文档必须要有特定的数据结构来支持命中结果的显

示和排序，因此不能简单地用原始文本文档的内容作为命中结果。类里定义的数据成员有文档 Id、文档绝对路径、文档内容、命中结果得分、命中的单词和对应的 Posting 键值对。

(2) IndexSearcher 类

该类用于进行全文搜索。它需要支持下列方法或数据结构：定义了多个检索词之间的与或关系的嵌套静态枚举类型、打开指定的索引文件方法、单个检索词的检索方法、二个检索词的检索方法

(3) SimpleSorter 类

该类用于对命中结果计算得分和排序。这里的设计模式使用了策略模式，将计算文档得分的方法放入接口 Sort 中，并将文档的得分值会保存至 Hit 对象中。同时，为了保证倒序排序的正确性，我们需要将得分设为负值，使得可以按照得分从高到低排列。

三、软件开发

使用 IntelliJ IDEA Community Edition 2022.3.3 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 jdk-17 版本。

四、软件测试

自动测试结果为，测试用例用例 106 个，完成 106 个，失败 0 个，跳过 0 个。自动测试截图如下所示。

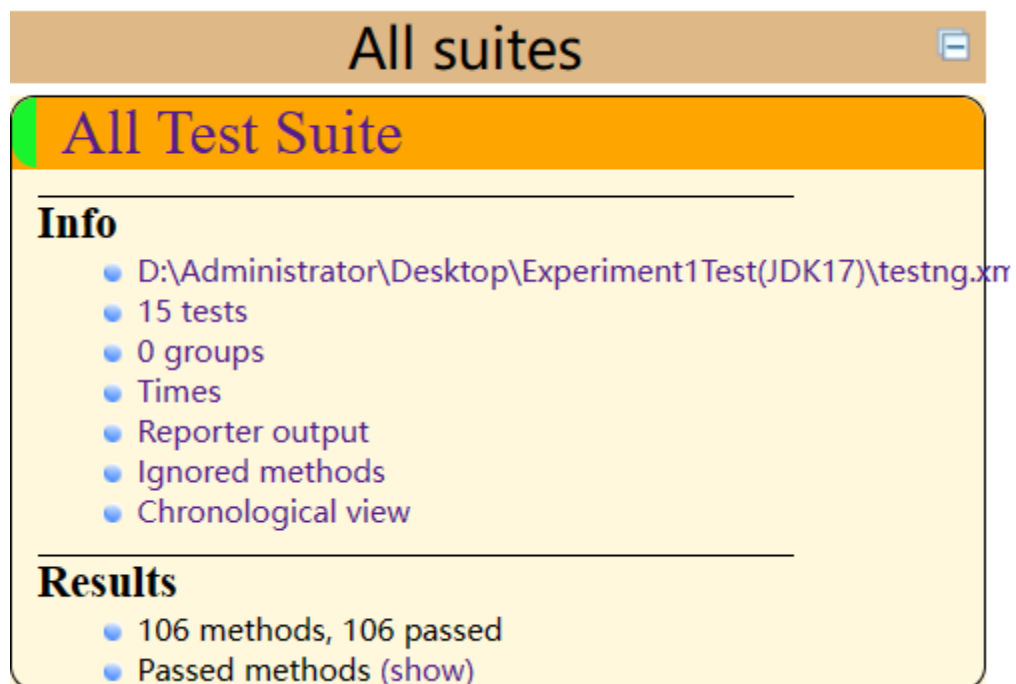


图 5 自动测试截图


```
;@5974109)
PASSED: testSearch([Lhust.cs.javacourse.search.query.AbstractHit;@27305e6, [Lhust.cs.jav
;@1ef3efa8)
PASSED: testTestSearch([Lhust.cs.javacourse.search.query.AbstractHit;@502f1f4c, [Lhust.c
ctHit;@6f8f9349)

=====

D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/query/IndexSearcherTest.
Tests run: 3, Failures: 0, Skips: 0

=====

All Test Suite
Total tests run: 106, Failures: 0, Skips: 0
=====
```

图 6 自动测试命令行截图

各个测试部分的测试结果如下所示。

Test	# Passed	# Skipped	# Failed	Time (ms)
All Test Suite				
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/index/TermTest.java	9	0	0	34
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/index/TermTupleTest.java	4	0	0	7
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/index/PostingTest.java	16	0	0	19
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/index/PostingListTest.java	17	0	0	13
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/index/DocumentTest.java	12	0	0	10
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/parse/TermTupleScannerTest.java	2	0	0	8
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/parse/StopWordTermTupleFilterTest.java	3	0	0	6
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/parse/PatternTermTupleFilterTest.java	2	0	0	4
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/parse/LengthTermTupleFilterTest.java	2	0	0	4
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/parse/ScannerFilterAllInOneTest.java	2	0	0	4
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/index/DocumentBuilderTest.java	7	0	0	509
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/index/IndexTest.java	10	0	0	1,898
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/index/IndexBuilderTest.java	1	0	0	116
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/query/HitTest.java	16	0	0	123
D:/IdeaWorkspace/SeachEngine/test/hust/cs/javacourse/search/query/IndexSearcherTest.java	3	0	0	187
Total	106	0	0	2,942

图 7 自动测试详细结果

从自动测试的测试样例可以看出，我们基本实现了倒排索引的功能，并通过了所有测试样例。

我们继续验证程序构建索引并进行查询的情况。首先，我们在 TestBuildIndex 类中定义运行方法 main，并在控制台进行运行。我们首先进行题目中所给样例的测试，在 1.txt~3.txt 的倒排索引结构构建完毕后，我们进行查询操作。其结果如图 8 所示。

```
Run: TestSearchIndex x TestSearchIndex x TestBuildIndex x
"D:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:D:\Program Files\JetBrains\IntelliJ IDEA Community
Start building index ...
{
  "docIdToDocPathMapping" : {
    {0=D:\Administrator\Desktop\Java新实验一\SearchEngineForStudent\text\1.txt, 1=D:\Administrator\Desktop\Java
  "termToPostingListMapping" : {
    {accord=PostingList{list=[Posting{docId=2, freq=1, positions=[30]}]}, according=PostingList{list=[Posting{
  }
}
-----
{
  "docIdToDocPathMapping" : {
    {0=D:\Administrator\Desktop\Java新实验一\SearchEngineForStudent\text\1.txt, 1=D:\Administrator\Desktop\Java
  "termToPostingListMapping" : {
    {accord=PostingList{list=[Posting{docId=2, freq=1, positions=[30]}]}, according=PostingList{list=[Posting{
  }
}

Process finished with exit code 0
```

图 8 功能测试索引建立

注意到逐个文本的索引都已经建立，并且得到了文本所对应的文档 ID、文档路径、文档 TermTuple 的对应内容。我们继续对功能测试进行验证，如图 9 功能测试查询所示。

```
Run: TestSearchIndex x TestSearchIndex x
"D:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:D:\Program Files\JetBrains\IntelliJ IDEA Community
| 2. Query word(s) [AND/OR] |
aaa

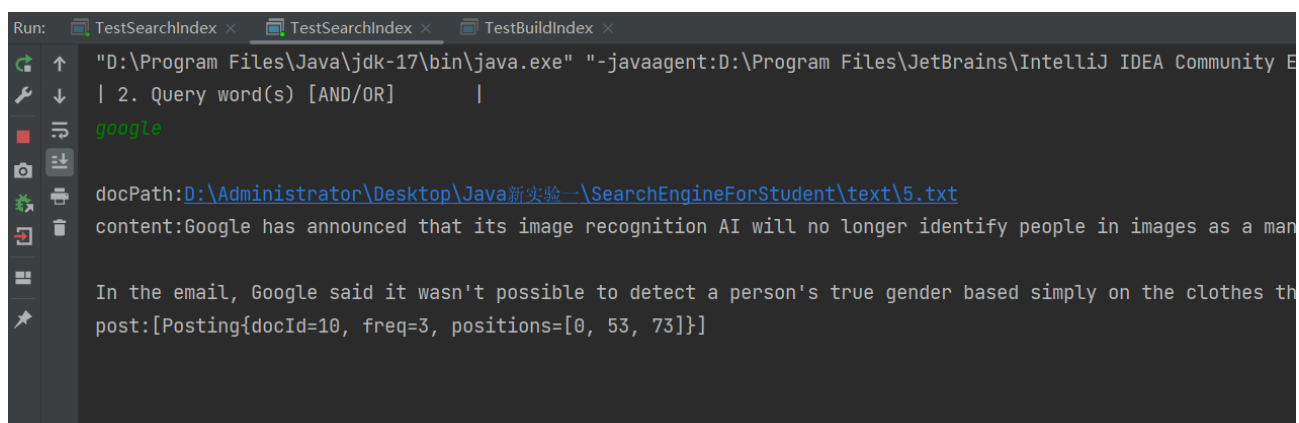
docPath:D:\Administrator\Desktop\Java新实验一\SearchEngineForStudent\text\1.txt
content:aaa aaa bbb ccc ddd eee
fff aaa bbb ccc ddd eee
post:[Posting{docId=0, freq=3, positions=[0, 1, 7]}]

docPath:D:\Administrator\Desktop\Java新实验一\SearchEngineForStudent\text\2.txt
content:aaa bbb ccc ddd eee
fff ggg fff ggg
post:[Posting{docId=1, freq=1, positions=[0]}]
```

图 9 功能测试查询

我们继续对程序实际功能进行测试。我们导入实际功能测试集中 1.txt~15.txt 共 15 个文本，并构建其对应的倒排索引。基于已经建立好的倒排索引，我们在编写 TestSearchIndex 程序作为测试入口，在控制台进行单词查询。其结果如下。

(一) 查询一个单词

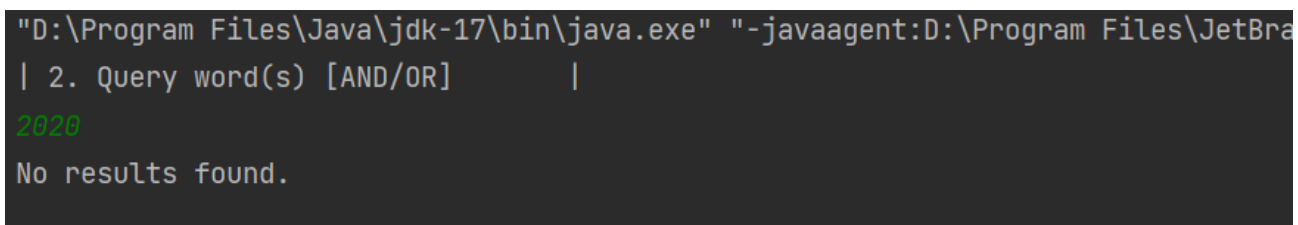


```
Run: TestSearchIndex x TestSearchIndex x TestBuildIndex x
"D:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:D:\Program Files\JetBrains\IntelliJ IDEA Community E
| 2. Query word(s) [AND/OR] |
google

docPath:D:\Administrator\Desktop\Java新实验一\SearchEngineForStudent\text\5.txt
content:Google has announced that its image recognition AI will no longer identify people in images as a man

In the email, Google said it wasn't possible to detect a person's true gender based simply on the clothes th
post:[Posting{docId=10, freq=3, positions=[0, 53, 73]}]
```

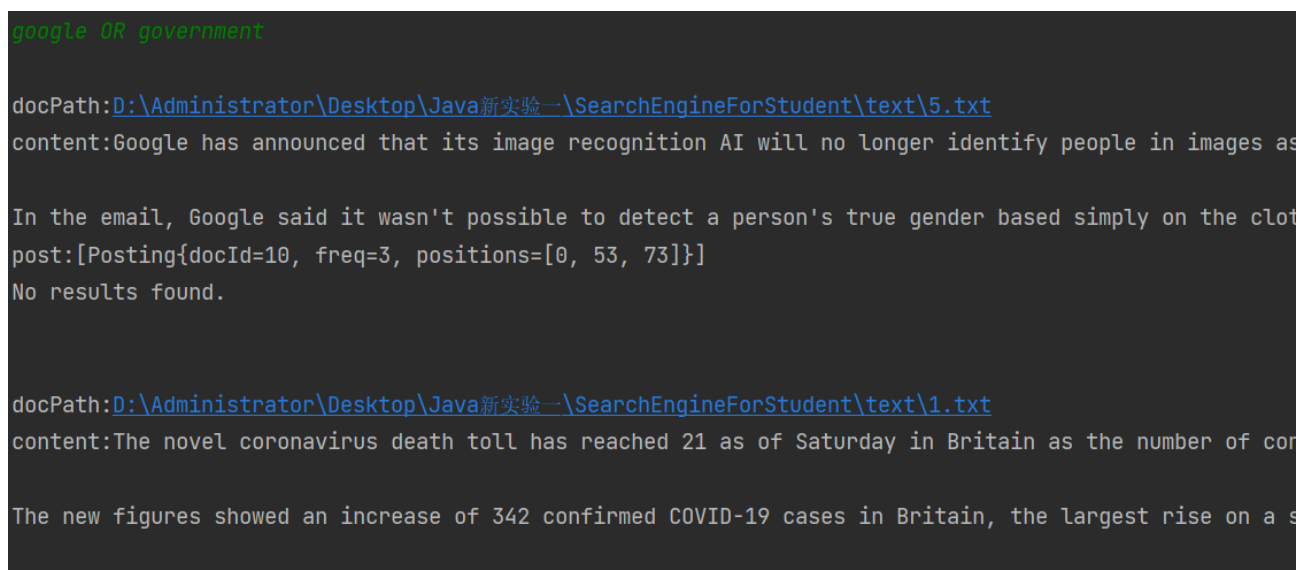
图 10 真实数据集测试单词输入



```
"D:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:D:\Program Files\JetBra
| 2. Query word(s) [AND/OR] |
2020
No results found.
```

图 11 真实数据集-非法输入

(二) 查询两个单词



```
google OR government

docPath:D:\Administrator\Desktop\Java新实验一\SearchEngineForStudent\text\5.txt
content:Google has announced that its image recognition AI will no longer identify people in images as

In the email, Google said it wasn't possible to detect a person's true gender based simply on the clot
post:[Posting{docId=10, freq=3, positions=[0, 53, 73]}]
No results found.

docPath:D:\Administrator\Desktop\Java新实验一\SearchEngineForStudent\text\1.txt
content:The novel coronavirus death toll has reached 21 as of Saturday in Britain as the number of cor

The new figures showed an increase of 342 confirmed COVID-19 cases in Britain, the largest rise on a s
```

图 12 查询两个单词 (OR)

```
TestSearchIndex x
↑
↓
google AND government
docPath:D:\Administrator\Desktop\Java新实验一\SearchEngineForStudent\text\5.txt
content:Google has announced that its image recognition AI will no longer identify

In the email, Google said it wasn't possible to detect a person's true gender based
post:[Posting{docId=10, freq=3, positions=[0, 53, 73]}]

docPath:D:\Administrator\Desktop\Java新实验一\SearchEngineForStudent\text\1.txt
content:The novel coronavirus death toll has reached 21 as of Saturday in Britain a

The new figures showed an increase of 342 confirmed COVID-19 cases in Britain, the

All the 10 patients who died were aged over 60 and had underlying health conditions

According to health authorities, most of the cases are in England. There have been

The British government said on Friday that it estimated the true number of infected
post:[Posting{docId=0, freq=1, positions=[133]}]
```

图 13 查询两个单词 (AND)

(三) 查询短语

```
health conditions

docPath:D:\Administrator\Desktop\Java新实验一\SearchEngineForStudent\text\1.txt
content:The novel coronavirus death toll has reached 21 as of Saturday in Britain as the n

The new figures showed an increase of 342 confirmed COVID-19 cases in Britain, the largest

All the 10 patients who died were aged over 60 and had underlying health conditions, said

According to health authorities, most of the cases are in England. There have been 121 con

The British government said on Friday that it estimated the true number of infected cases
post:[Posting{docId=0, freq=2, positions=[94, 106]}]
```

图 14 短语查询

可以看出，我们的查询程序 `TestSearchIndex` 可以实现对单个单词、两个单词的搜索操作。同时，搜索结果会按照 `Hit` 中的得分倒叙排列输出。

五、特点与不足

1. 技术特点

(1) 我们建立了每个三元组的倒排索引列表以加快查询单词的速度。通过各个数据结构和算法的实现，我们可以将查询结构封装为 Hit，并按照文档的负担多少进行降序排序，将出现频率较高的文档排列在前方；

(2) 我们构建了多个过滤器，并通过装饰者模式进行文本单词的逐个读取。这样提高了代码的可读性和可理解性，并使得类的封装安全、可靠。我们在 TermTupleScanner 输入函数内用队列读取文件，并加载至内存中；

(3) 我们灵活使用各个对象的序列化方法，将数据结构从单词 (term) 出发，逐步形成倒排索引的搜索结构。

2. 不足和改进的建议

不足之处为短语查询方面仍然存在一定的欠缺，得分计算和相邻结果的实现具有难度。

六、过程和体会

1. 遇到的主要问题和解决方法

- (1) 在自动测试的时候，最初出现了许多加载的 Index 对象为空的错误。反复的修正无果，随后发现是测试集使用错误，应当使用 JDK-17 的测试包；然后将测试文件使用新的测试包测试，运行后就能通过了；
- (2) 查询时最初无法识别逻辑关系连接词 AND/OR，输出结果和预期不同。使用 AND 的查询结果，反而比使用 OR 的要多。后来发现是函数调用的问题，修改正确即可；

2. 课程设计的体会

JAVA 实验使我收获良多。这是我第一次用 JAVA 编写规模较大的程序，对逐一调试各个类的方法也有了更好的理解。我也通过本次实验厘清各个类之间的联系，学会有实现目的地按部就班地实现每一个函数类，对自己学习 JAVA 编程很有帮助。

其次，我也对抽象类的指导作用有了更深刻的理解。这次实验中，我们的每一个具体类几乎都是在抽象类的指引下完成的，它无疑起着规范代码、明确实现的作用，让我们对于较大项目的实现有了一定的认识。

总之，这次 JAVA 实验课程让我收获很多，也有了很好的提升！尽管编写的过程很艰难，经常有调试不出的情况，但是最终完成了编写，感到很有成就感。

七、源码和说明

1. 文件清单及其功能说明







 Experiment1Test(JDK17)	2023-05-15 2:48	文件夹	
 U202112071_王彬_源码	2023-05-15 2:46	文件夹	
 测试结果截图	2023-05-15 2:45	文件夹	
 readme.txt	2023-05-15 2:44	文本文档	1 KB
 U202112071_王彬_报告.docx	2023-05-15 2:55	Microsoft Word ...	1,215 KB
 U202112071_王彬_报告.pdf	2023-05-15 2:50	Microsoft Edge ...	1,278 KB

图 15 提交文件清单

提交的文件目录如图 15 所示。

- (1) Experiment1Test(JDK17)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。可以通过阅读其中的两个 html 文件获取测试结果
- (2) 源码文件夹下有 SearchEngineForStudent 项目文件，本实验中所编写的源文件.class 全都包含在此文件夹下，可以使用 IDEA 打开 SearchEngineForStudent.iml 进行代码的审阅。
- (3) 文件包中的.docx/.pdf 文件为实验报告。

2. 用户使用说明书

- (1) 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。
- (2) 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 TestBuildIndex 类和 TestSearchIndex 类，分别是索引建立和单词查询。在运行 TestBuildIndex 后即可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即可以得到运行结果。

3. 源代码

见附件中的 SearchEngineForStudent 文件夹。