

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：计算机 2001 班

学 号：U202011641

姓 名：刘景宇

指导教师：辜希武

2020 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 and 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

2.1 需求背景

在关系数据库系统里，索引是检索数据最有效率的方式。但对于搜索引擎，它并不能满足其特殊要求：(1) 海量数据：搜索引擎面对的是海量数据，像 **Google**，百度这样大型的商业搜索引擎索引都是亿级甚至百亿级的网页数量，面对如此海量数据，使得数据库系统很难有效的管理。(2) 数据操作简单：搜索引擎使用的数据操作简单，一般而言，只需要增、删、改、查几个功能，而且数据都有特定的格式，可以针对这些应用设计出简单高效的应用程序。而一般的数据库系统则支持大而全的功能，同时损失了速度和空间。最后，搜索引擎面临大量的用户检索需求，这要求搜索引擎在检索程序的设计上要分秒必争，尽可能的将大运算量的工作在索引建立时完成，使检索运算尽量少的。一般的数据库系统很难承受如此大量的用户请求，而且在检索响应时间和检索并发度上都不及我们专门设计的索引系统。

现代搜索引擎的索引都是基于倒排索引。它被用来存储在全文搜索下某个单词在一个文档或者一组文档中的存储位置的映射。它是文档检索系统中最常用的数据结构。通过倒排索引，可以根据单词快速获取包含这个单词的文档列表，从而大大提高了查找的效率。

2.2 需求细化

利用 **Java** 实现基于内存的搜索引擎，需要包括设计相应的数据结构以及查找算法，设计的数据结构要能够比较好的表示单词之间的关系，并且便于查找算法的实现。整个算法要具有比较高的复用性，以及模块化，能够充分体现面向对象程序设计的优势。

倒排索引主要包括两个大模块，一个是构建索引，另一个是索引查找。构建索引包括相应的数据结构 **index** 模块以及要用到的分词过滤 **parse** 模块，索引查找包括 **query** 模块。

对于构建索引模块，能够读取文本内容，将文本内容切分成单词，并且利用 **parse** 模块过滤掉不需要的单词，过滤掉过短或过长的单词、停用词以及格式不正确的单词。为了能够长期保存，要实现对象的序列化以及反序列化。其中 **parse** 模块使用装饰者和迭代器模式，能够不暴露内部具体存储，且不断返回下一个符合要求的对象。

对于索引查找 **query** 模块，能够对单个搜索关键词进行检索，可以计算每个命中文档的得分进行排序，并且可以实现两个关键词的搜索，可以选择是进行与查找还是或查找，按照命中率排序，返回用户最可能关心的内容。

二、系统设计

1. 概要设计

利用 Java 实现基于内存的搜索引擎，采用倒排索引的方式进行索引。这里将其分为构建索引 index 模块、单词过滤 parse 模块、排序索引 query 模块三个模块。其中 Index 模块在构建索引的过程中使用了 parse 模块进行单词过滤，query 模块对 index 模块构建好的索引进行搜索。（整体框架如图 1 所示，整体流程示意如图 2 所示）

构建索引 Index 模块，主要实现索引的数据结构以及构建索引。Index 模块包括 Term、Posting、Document、Index 等类，这些类主要作为内部数据结构，用来实现存储索引的内部逻辑。同时还包括了 DocumentBulider，这个类基于上面实现的数据结构，利用单词过滤 parse 模块的过滤器实现了对文本文档的解析，即将文本文档描述为 Java 语言，IndexBuilder 类则能够利用 DocumentBuilder 来实现一组文本文档的解析，并根据获得的 Document 对象来构建索引。

单词过滤 parse 模块，主要用在构建索引时对不符合要求的三元组进行过滤。这里采用装饰者模式，实现多种过滤模式。parse 模块下包括 LengthTermTupleFilter、PatternTermTupleFilter、StopWprdTermTupleFilter 等过滤器类，主要用来过滤掉长度不符合要求、格式不符合要求以及停用词，这样可以尽可能地减小不关心的词语带来的干扰，从而提高命中率。

排序索引 query 模块，该模块主要实现倒排索引，即从单词找到文本文档，利用命中率来尽可能优先获取用户关心的文档。query 模块下包括 SimpleSorter、Hit、IndexSearcher 类。Hit 用来计算命中率并保存命中结果，SimpleSorter 类实现 Sort 接口，用来对命中的文档根据命中率进行排序，IndexSearcher 类主要用来进行索引，根据用户提供的关键词在构造好的 Index 索引中命中相应文档，并得到命中的对象数组。

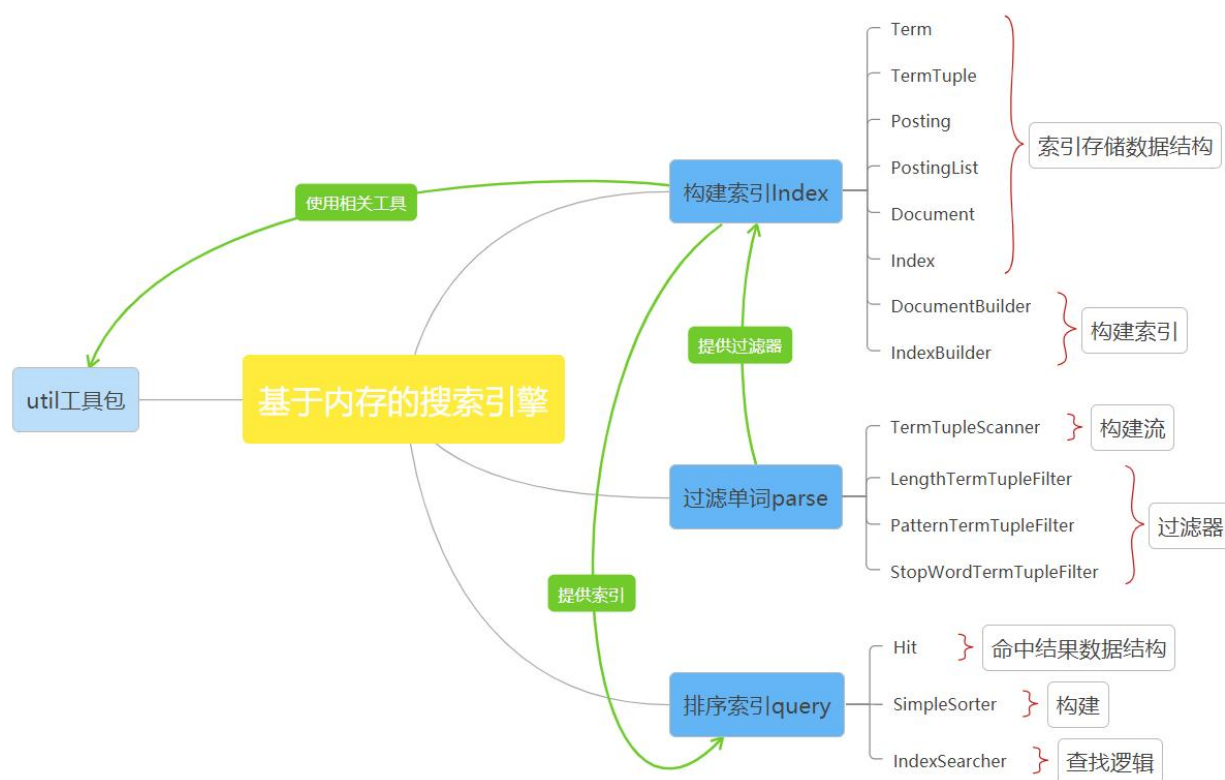


图 1 基于内存的搜索引擎框架

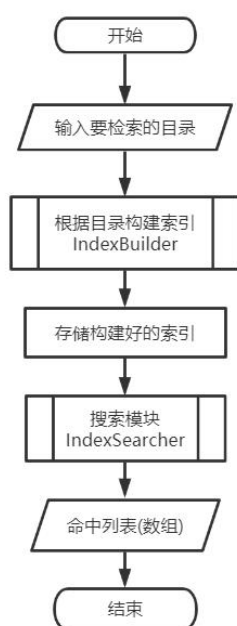


图 1 基于内存的搜索引擎流程示意图

2. 详细设计

2.1 构建索引 Index 模块

该模块主要实现索引的数据结构以及构建索引。包含了 Term、Posting、Document 等主要用来表示数据结构类，它们实现了 FileSerializable 接口，能够序列化到文件或从文件反序列化，同时包含了 DocumentBuilder 等构建索引的类。（如图 3 所示）

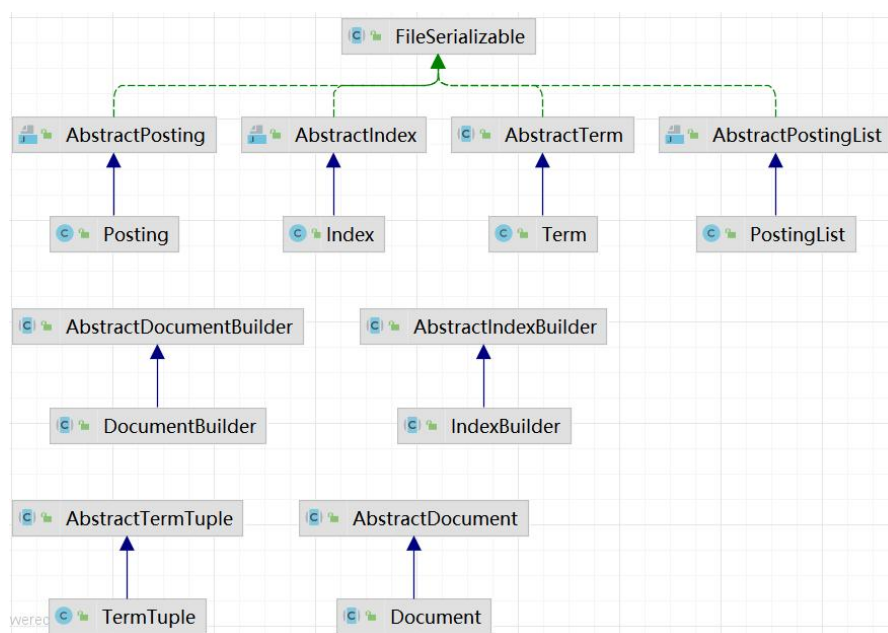


图 3 Index 模块类图

（1）Term

主要功能：表示文本文档里的一个单词。

属性：protected String content

关键方法：

public int compareTo(AbstractTerm o)，该方法实现了按照字典序对两个 Term 对象内容的比较，调用 String 类实现的 compareTo 方法得到两个单词的字典序差值。

（2）TermTuple

主要功能：表示三元组，包含单词、出现频率以及出现的当前位置。

属性：public AbstractTerm term、public final int freq、public int curPos

关键方法：

public TermTuple(AbstractTerm term,int curPos)，该方法为构造函数，将解析得到的单词扩展生成三元组。

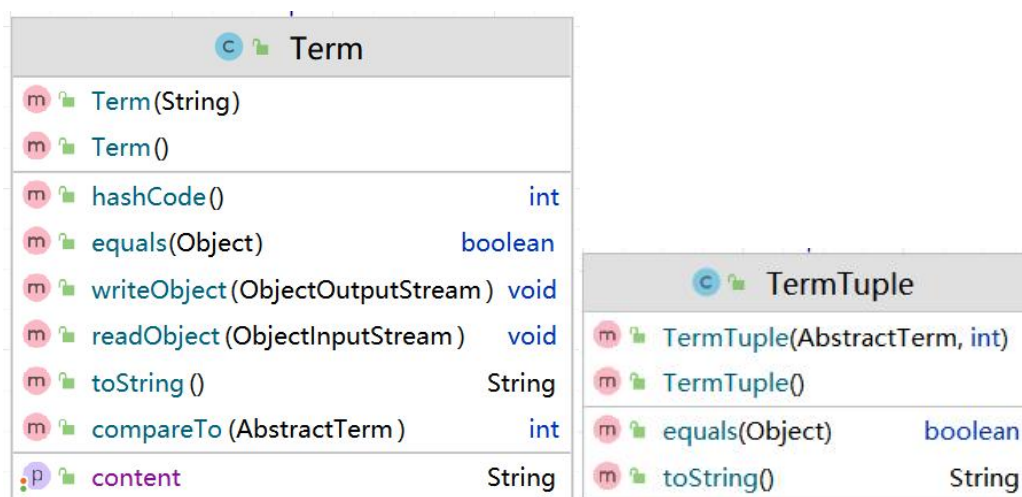


图 4 Term、TermTuple 类图

（3）Posting

主要功能：表示倒排索引里的一项，包括单词文档 id，出现次数，单词出现的位置列表。

属性：protected int docId、protected int freq、protected List<Integer> positions;

关键方法：

public void sort()，该方法将一个 Posting 对应的 positions 列表进行排序，通过调用 List 对象的 sort 方法实现。

（4）PostingList

主要功能：表示一个单词对应的 Posting 列表

属性：protected List<AbstractPosting> list

关键方法：

`public void sort()`，该方法将一个 `PostingList` 中所有的 `postings` 按照 `docId` 进行升序排序，通过调用 `List` 对象的 `sort` 方法实现，调用时传入

`Comparator.comparingInt(AbstractPosting::getDocId)` 实现排序功能。

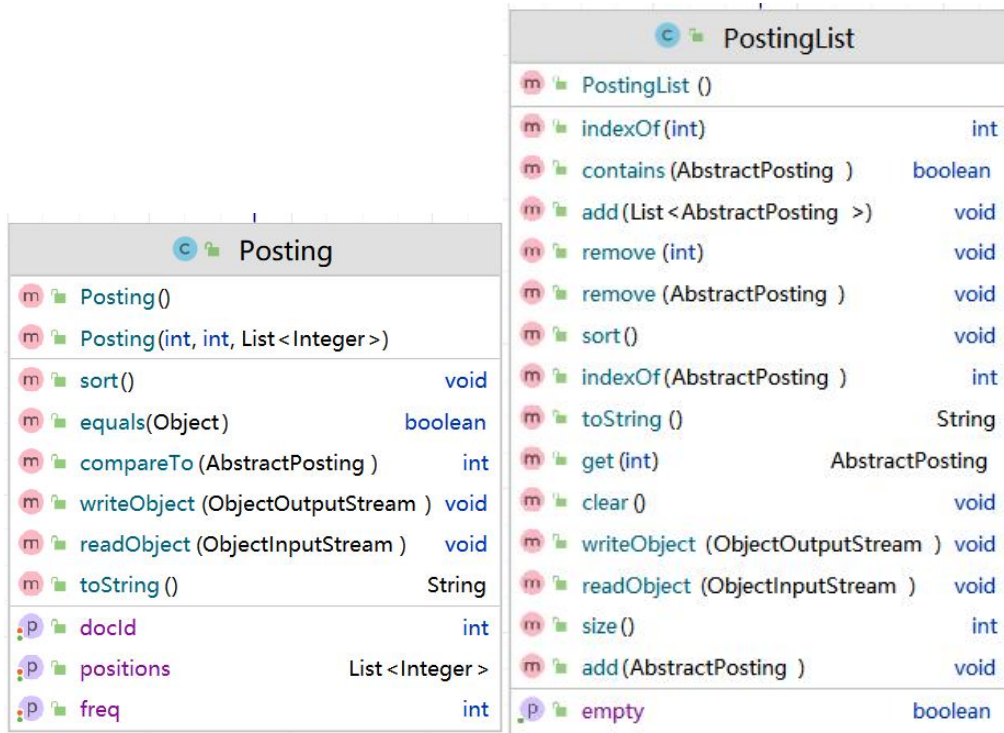


图 5 Posting、PostingList 类图

(5) Document

主要功能：表示一个文本文件解析得到的结果

属性：`protected int docId`、`protected String docPath`、`protected List<AbstractTermTuple> tuples`

关键方法：

`public void addTuple(AbstractTermTuple tuple)`，该方法用来实现向文档对象里添加三元组，并且不能重复，通过调用 `List` 对象的 `contains` 和 `add` 方法实现，如果 `tuples.contains(tuple)` 为假，那么就调用 `tuples.add(tuple)` 方法将 `tuple` 加入到 `tuples` 列表中，否则不加入。

(6) DocumentBuilder

主要功能：由解析文本文档得到的 `TermTupleStream` 产生 `Document` 对象

关键方法：

`public AbstractDocument build(int docId, String docPath, AbstractTermTupleStream termTupleStream)`，该方法通过 `termTupleStream` 可以不断获取文档解析的下一个 `tuple`，利用类似迭代器模式的方法，将三元组流的 `next` 方法获取的三元组加入到新建的 `Document` 对象中。

`public AbstractDocument build(int docId, String docPath, java.io.File file)`，该方法通过利用

parse 模块中的过滤器将 File 对象转换成 AbstractTermTupleStream 对象, 并利用装饰者模式的思想方法, 用多种过滤方式装饰 filter, 通过复用前一个 build 方法, 将三元组流转换成 Document 对象。

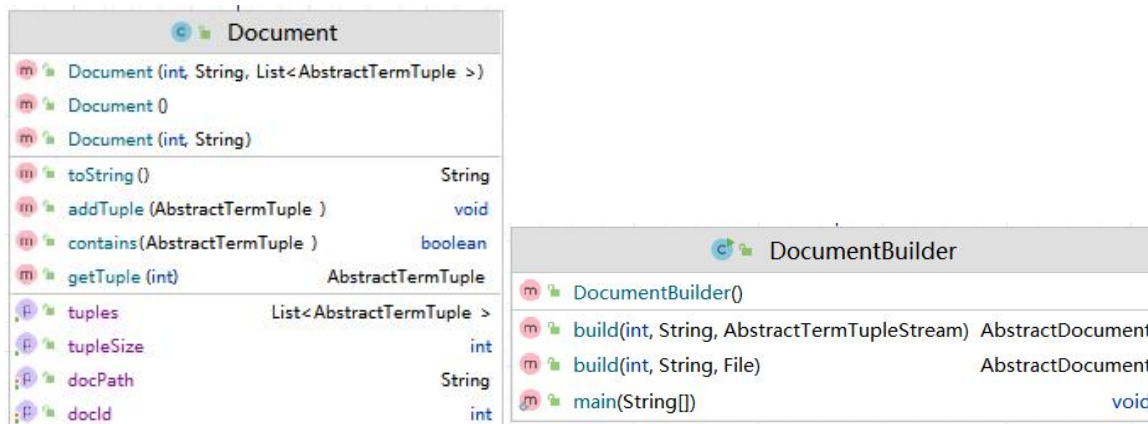


图 6 Document、DocumentBuilder 类图

(7) Index

主要功能: 表示构建的索引

属性: `public Map<Integer,String> docIdToDocPathMapping`、`public Map<AbstractTerm,AbstractPostingList> termToPostingListMapping`

关键方法:

`public void addDocument(AbstractDocument document)`, 该方法实现添加文档索引, 并更新内部的 HashMap。当加入一篇新文档时, 对其得到的每一个三元组 tuple, 要加入到 Index 对象中, 即要更新到 Index 对象的两个 Map 中。如果 tuple.term 已经在 termToPostingListMapping 中存在, 那么就只需要将该 term 对应的 Postinglist 更新, 更新时要判断该 term 是否包含在该 PostingList 中, 若存在就只需更新该 Posting 的 positions 列表, 若不存在就新建一个 Posting 对象, 将其加入到 PostingList 的末尾。如果 tuple.term 不存在于 termPostingListMapping 中, 那么就要新建一个 key-value 键值对, 并将其加入到 termPostingListMapping 中。

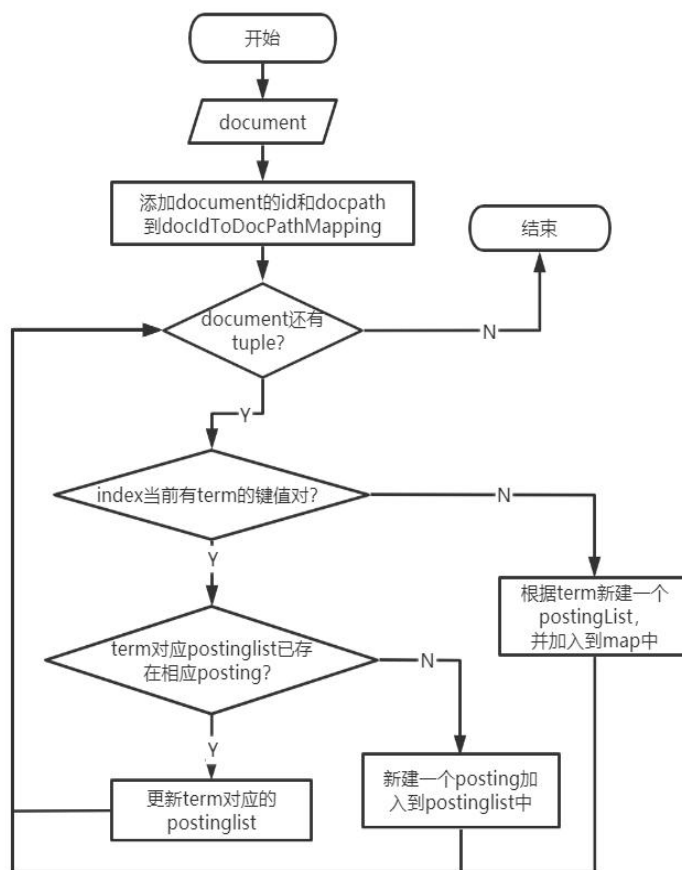


图 27 addDocument 方法流程图

public void optimize(), 该方法实现索引的优化, 对索引里每个单词的 PostingList 按 docId 从小到大排序, 同时对每个 Posting 里的 positions 从小到大排序。

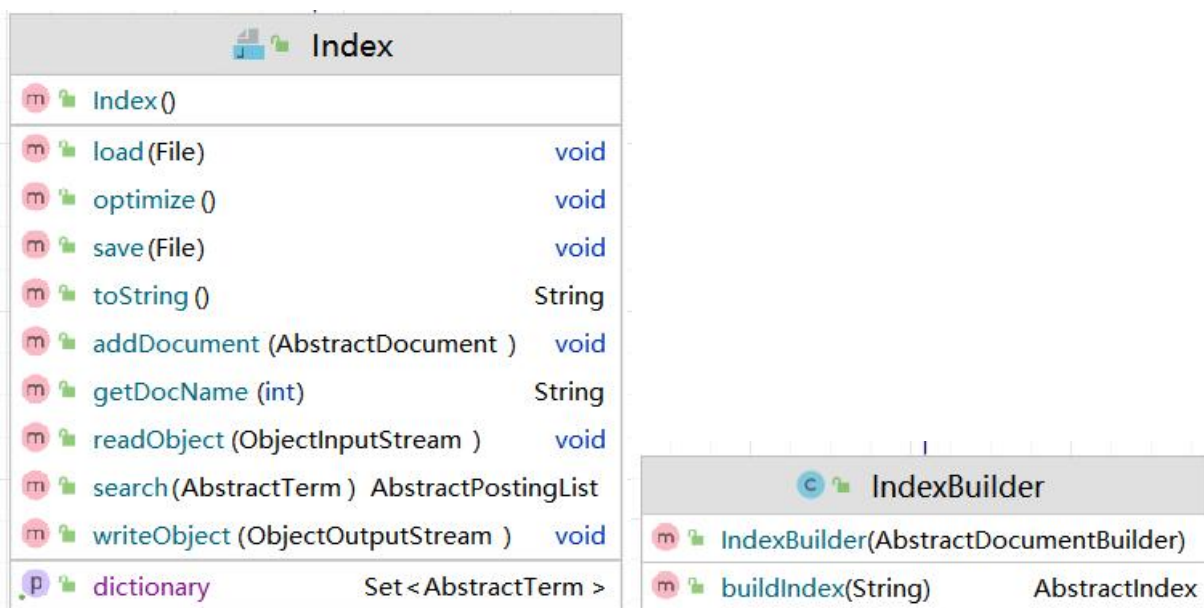


图 8 Index、IndexBuilder 类图

(8) IndexBuilder

主要功能：构建索引，将指定的文档构建成 Index 对象

属性：protected AbstractDocumentBuilder docBuilder、protected int docId

关键方法：

`public AbstractIndex buildIndex(String rootDicrectory)`，该方法构建指定目录下的所有文本文件的倒排索引。实现时要遍历和解析指定目录下的每个文本文件，得到 Document 对象，并将这些 Document 对象逐个加入到该 Index 对象。

2.2 单词过滤 parse 模块

该模块采用装饰者模式设计，所有过滤器都有一个祖先类 AbstractTermTupleStream，也就是所有过滤器都可以作为 AbstractTermTupleStream 类的对象，同时所有过滤器类的构造函数的参数为 AbstractTermTupleStream 类的对象。该模块下类的继承关系，以及主要方法如下图所示。

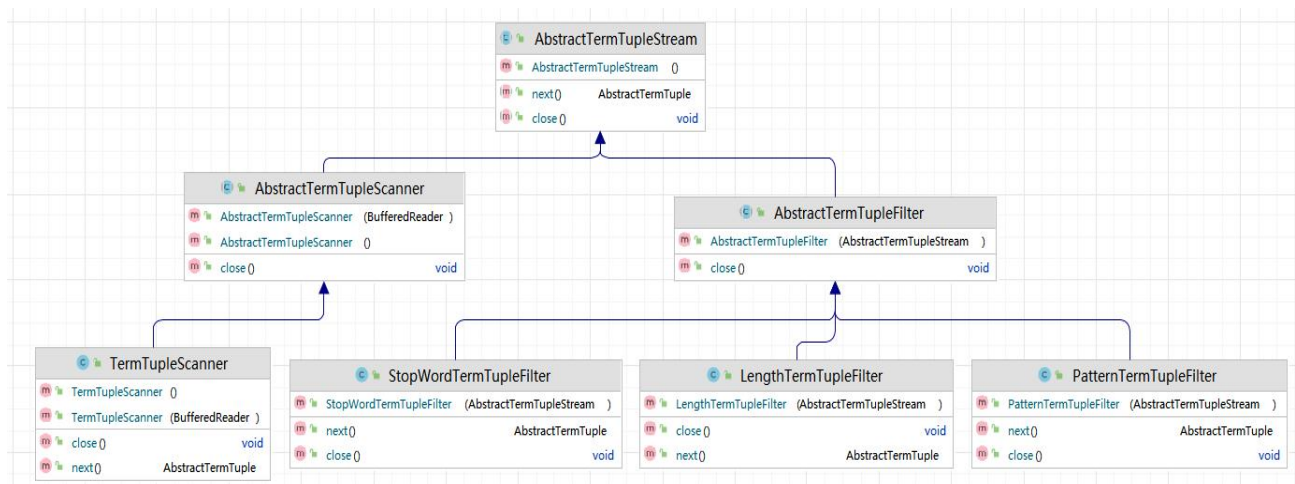


图 9 parse 模块 uml 图

(1) TermTupleScanner

主要功能：将文本文档转换成三元组列表

关键方法：

`public AbstractTermTuple next()`，该方法不断获取下一个三元组，该方法的实现需要一个 position，用来确定当前读取三元组的位置，进而判断是否还有三元组。

(2) LengthTermTupleFilter

主要功能：过滤掉三元组流中长度不符合要求的三元组

关键方法：

`public AbstractTermTuple next()`，该方法通过与 Config 中规定的最短字长和最长字长进行比较，从而获得下一个长度符合要求的三元组。

(3) PatternTermTupleFilter

主要功能：过滤掉三元组流中格式不符合要求的三元组

关键方法：

`public AbstractTermTuple next()`，该方法通过与 `Config` 中规定的单词格式（正则表达式）进行匹配，进而获得下一个格式符合要求的三元组。

(4) StopWordTupleFilter

主要功能：过滤掉三元组流中的停用词

关键方法：

`public AbstractTermTuple next()`，该方法通过与 `Config` 中设定的停用词表进行比较，来获得下一个非停用词的三元组。

2.3 排序索引 query 模块

(1) Hit

主要功能：表示搜索命中结果

属性：`protected int docId`、`protected String docPath`、`protected String content`、`protected Map<AbstractTerm, AbstractPosting> termPostingMapping`、`protected double score`

关键方法：

`Public int compareTo(AbstractHit o)`，该方法用来比较两个命中结果的大小，根据每个命中的得分进行比较，返回两个命中结果的差值。

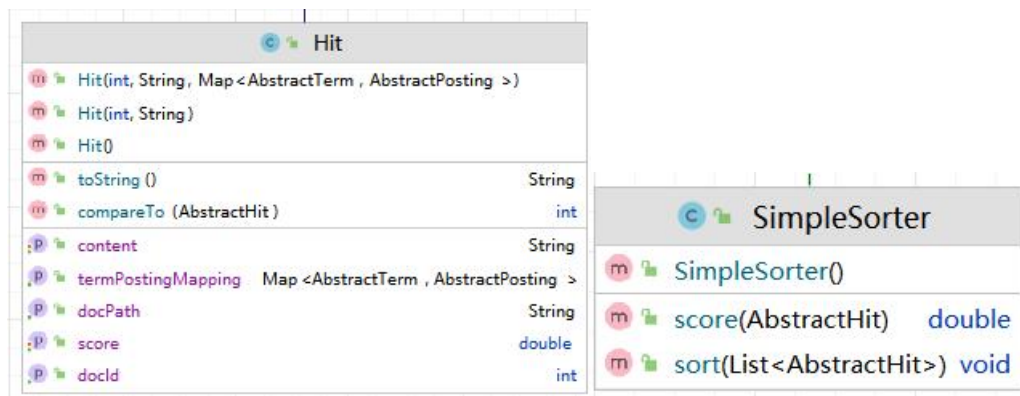


图 10 Hit、SimpleSorter 类图

(2) SimpleSorter

主要功能：根据得分进行排序，实现了 `sort` 接口

关键方法：

`public double score(AbstractHit hit)`，该方法实现对一个命中结果计算得分，得分主要用来排序，使得优先搜索出用户可能感兴趣的内容。利用 `hit` 中存储的 `posting` 的频率来计算该命中的得分。

`public void sort(List<AbstractHit> hits)`, 该方法用来实现对于一个命中列表的排序, 根据每个命中的得分按照降序进行排序, 利用 `List.sort()` 方法来排序。

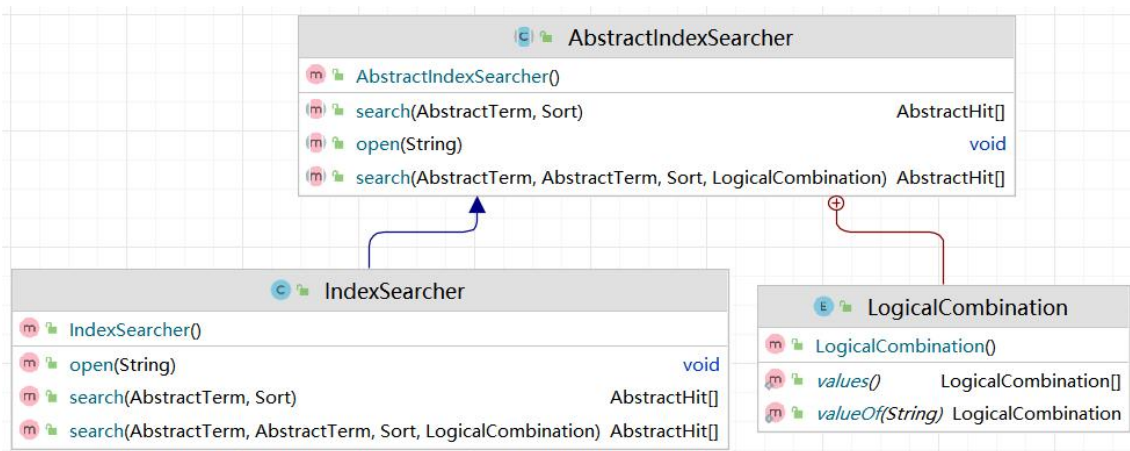


图 11 IndexSearcher 类图

(3) IndexSearcher

主要功能: 实现倒排索引, 用来进行查找

属性: `protected AbstractIndex index`

关键方法:

`public Abstract[] search(AbstractTerm queryTerm, Sort sorter)`, 该方法用来实现对于一个检索词的搜索, 并返回所有命中的对象。在构造函数中, 通过 `index.load()` 方法, 将索引文件打开, 加载 `index`。将给出的关键词 `queryTerm` 作为 `key` 在 `index` 中找到对应的 `PostingList`, 遍历得到的 `PostingList` 对象, 对每一个 `Posting` 对象构建一个命中对象 `hit`, 得到所有命中对象构成的列表, 并按照命中率得分进行排序。

`public AbstractHit[] search(AbstractTerm queryTerm1, AbstractTerm queryTerm2, Sort sorter, AbstractIndexSearcher.LogicalCombination combine)`, 该方法用来实现对于两个关键词的检索, 并且能够根据输入逻辑进行组合, 实现和或的逻辑关系查找。先得到两个检索词对应 `PostingList` 对象 `postingList1`、`postingList2`, 利用 `switch` 来实现对于逻辑关系的选择, 对于与逻辑, 在遍历 `postingList1` 的同时检查 `postingList2` 中是否有该 `Posting` 对象, 如果同时有那么就将其加入到命中列表中。对于或逻辑, 则将 `postingList1` 的所有 `Posting` 对象加入到命中列表, 并遍历 `postingList2`, 将命中列表中不含有的 `Posting` 对象加入到列表中。无论那种逻辑实现, 返回前都对命中列表进行排序。

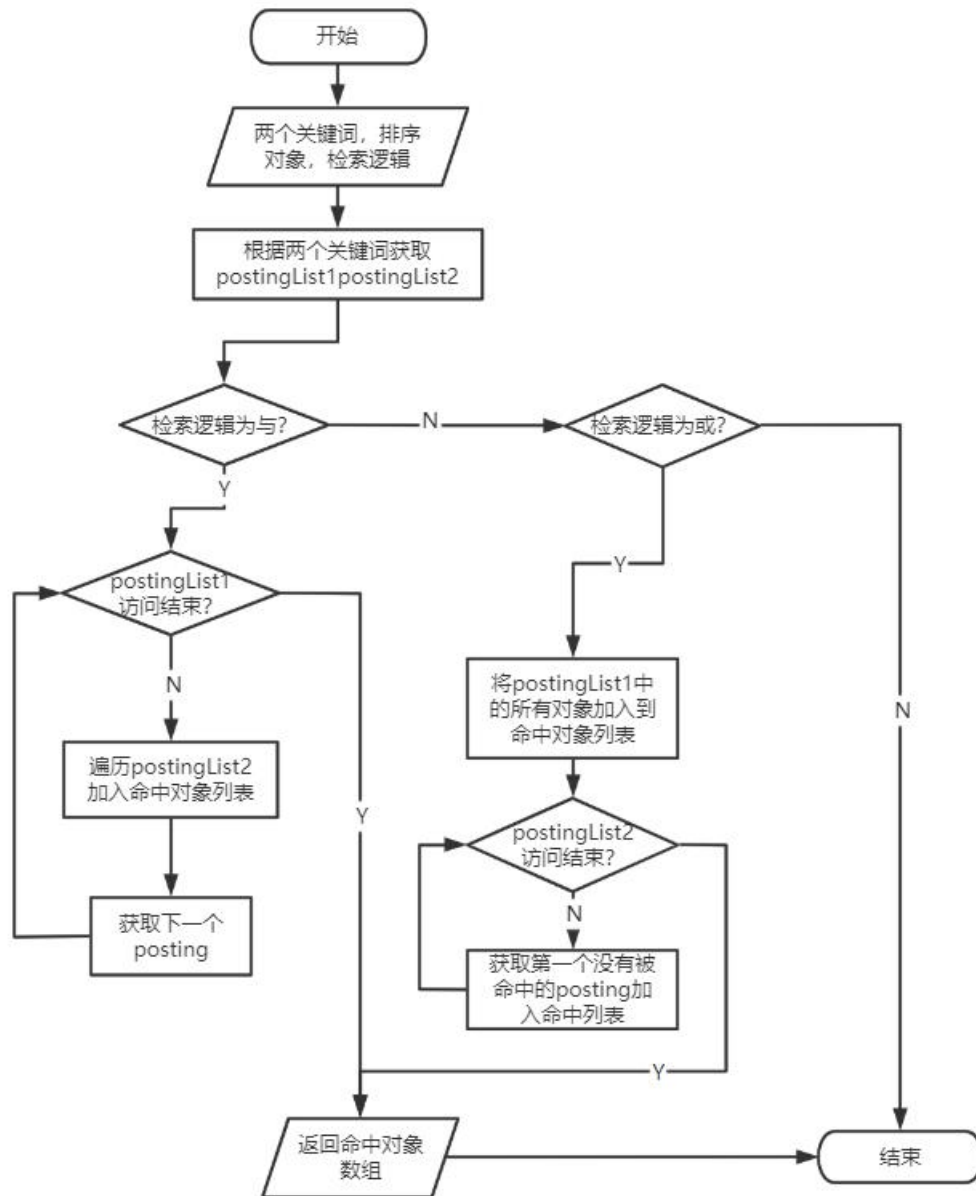


图 12 search 流程图

三、软件开发

硬件环境：PC 机，AMD Ryzen 5 2500U with Radeon Vega Mobile Gfx 2.00Ghz，8G 内存

操作系统：Windows10

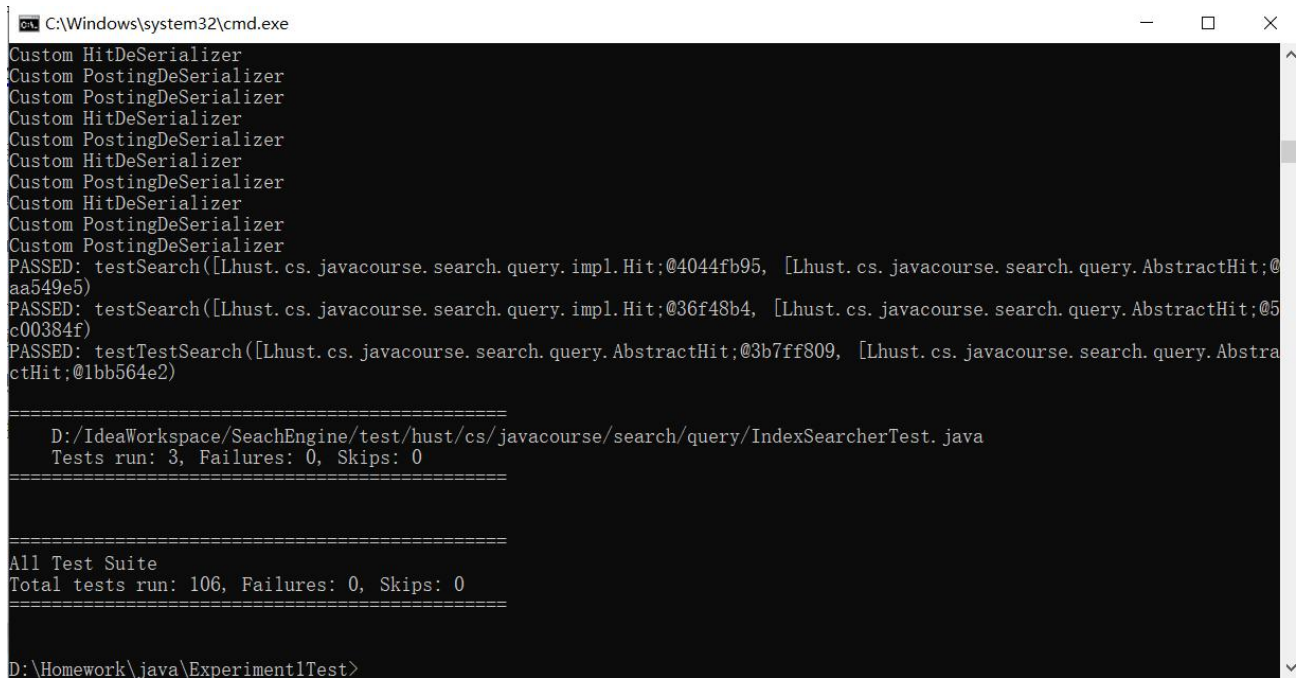
程序开发平台：IntelliJ IDEA 2021.3.2

JDK 版本：Java13

四、软件测试

利用自动测试程序进行测试，测试结果如下图所示。

图中表明所有测试用例均通过。



```
C:\Windows\system32\cmd.exe
Custom HitDeSerializer
Custom PostingDeSerializer
Custom PostingDeSerializer
Custom HitDeSerializer
Custom PostingDeSerializer
Custom HitDeSerializer
Custom PostingDeSerializer
Custom HitDeSerializer
Custom PostingDeSerializer
Custom PostingDeSerializer
PASSED: testSearch([Lhust.cs.javacourse.search.query.impl.Hit;@4044fb95, [Lhust.cs.javacourse.search.query.AbstractHit;@
aa549e5)
PASSED: testSearch([Lhust.cs.javacourse.search.query.impl.Hit;@36f48b4, [Lhust.cs.javacourse.search.query.AbstractHit;@5
c00384f)
PASSED: testTestSearch([Lhust.cs.javacourse.search.query.AbstractHit;@3b7ff809, [Lhust.cs.javacourse.search.query.Abstra
ctHit;@1bb564e2)

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

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

D:\Homework\java\Experiment1Test>
```

图 13 自动测试结果截图

五、特点与不足

1. 技术特点

充分利用了 Java 的面向对象的特点，设计多个抽象类、具体类、接口等，并且采用装饰者、迭代器等设计模式进行设计，使得整个程序设计比较高效。同时实现了倒排索引，简单体现了搜索引擎搜索的大致过程。

2. 不足和改进的建议

对于合理的输入能够给出正确的答案，但是对于出现异常情况的处理不足，代码的健壮性还有提升的空间，在计算得分时只是采用了简单的频率相加的方法，对于更复杂情况的考虑不足。

六、过程和体会

1. 遇到的主要问题和解决方法

对于工具类的方法的不熟悉，IDEA 集成开发环境代码自动补全在编程中提供了很大的帮助。仅凭借方法名字使用方法，结果用错方法，从而在编译甚至在运行中出现异常，导致程序崩溃，给代码调试带来不便，在使用过程中对于陌生的方法尽量查阅文档后再使用。

2. 课程设计的体会

通过整个实验过程，最大的收获是体验了动手实现一个项目，更加深刻的理解了一个项目从设计到落地的整个过程，提升了整体观、系统观，并且在编写程序的过程中，频繁使用

Java 的丰富的工具库，体验了 Java 面向对象编程的优势。

在编写程序的过程中，实现了装饰者、迭代器等设计模式，确实很巧妙的提高了编程效率，体会到了 Java 设计模式对于程序开发的帮助。

整个编写代码的过程中，IDEA 的代码自动补全等功能提供了非常多的帮助，在编程中熟悉了各个常见类的常见方法。