

**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 ： CS2201**

**学 号 ： U202215365**

**姓 名 ： 叶俊江**

**指导教师 ： 辜希武**

**2024 年 6 月 6 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。

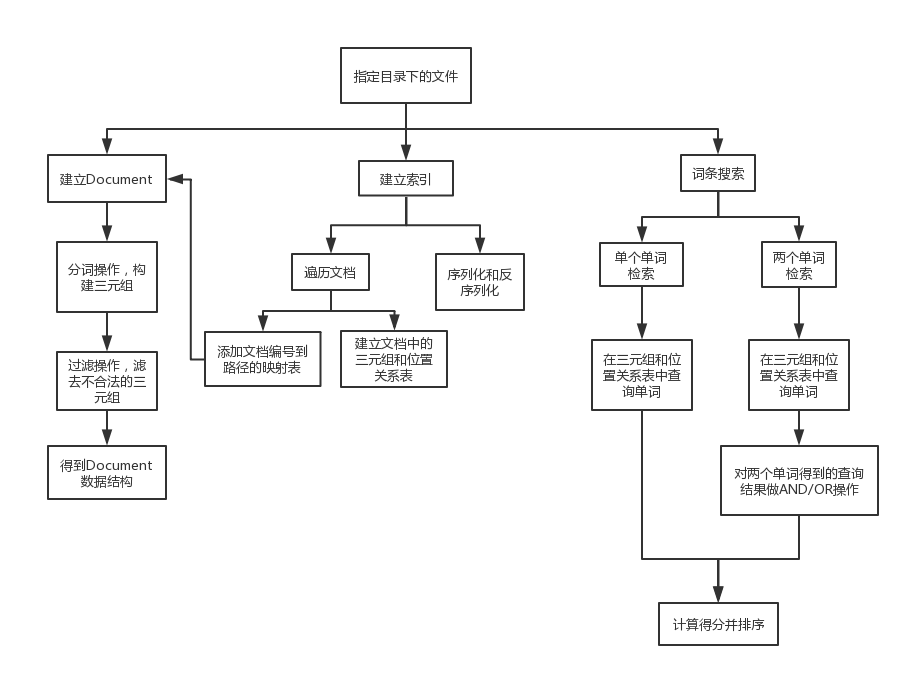


图 1 倒排索引流程示意图

主体流程分为三个部分，跟工程所分的三个package一样(即index, parse, query)，其工作流程分别为：

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**
5. **倒排索引模块**

构建索引用到的相关类的UML图如图 2 所示。

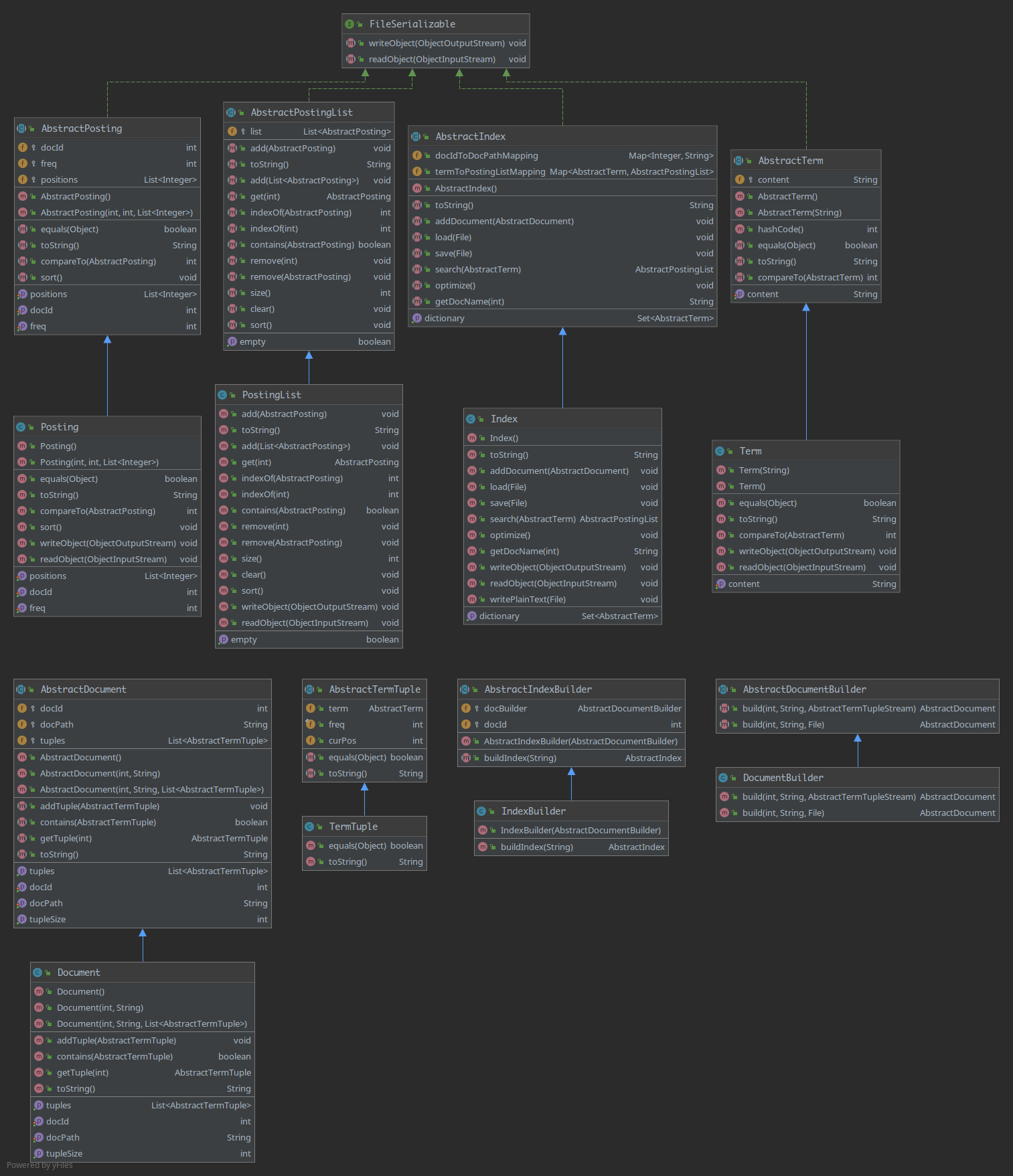


图 2 index包的UML图

下面是对index.impl包中实现对具体类对解释：

**·Term:**

对应着单词的结构类，主要是对String对象进行了简单对包装和一些简单方法实现的覆盖重写。

**·TermTuple：**

对某个单词出现情况的记录，主要数据成员有term（单词），freq（出现次数）以及curPos（单词出现位置），并给出了equal（），toString（）方法的重写。

**·Document：**

保存文档信息的Document类，主要数据成员有docId（文档id），docPath（文档绝对路径），tuples（文档中包含的三元组TermTuple对象列表），并给出了数据成员的set、get方法实现以及对tuples列表的add、get方法实现。

**·DocumentBuilder:**

Document的构造器，有两种build方法实现，根据传入参数不同选择不同的build方法，返回Document对象。此时对于参数列表为（int, String, File）的build方法要注意过滤器的装饰调用。

**·Index:**

对Document构造索引，主要的数据成员有docIdToDocPathMapping以及 termToPostingListMapping，均为Map类型，其中docIdToDocPathMapping存储内存中的docId和docPath的映射关系，termToPostingListMapping存储内存中的倒排索引结构，即Term和PostingList的映射关系。

其中主要实现了对Document对象的add方法，将Document中docId以及docPath添加到docIdToDocPathMapping中，并更新termToPostingListMapping。load和save方法主要是对构建好的索引实现文件读写操作。search方法则是对termToPostingListMapping进行对特定单词Term的搜索，若搜索到则返回对应的PostingList。getDictionary方法则是返回termToPostingListMapping转Set类型的对象。optimize方法则是对termToPostingListMapping中的PostingList进行排序优化。

**·IndexBuilder:**

Index索引的构造器，对给出的目录下的文件进行遍历检索，得到Document对象，然后根据Document对象构建索引，将Document对象加入index索引返回。

**·Posting：**

保存特定Term在每个Document的信息，主要的数据成员有：docId（包含单词的文档id），freq（单词在文档中出现的次数），positions（单词在文档中出现的位置列表，为List类型）。

**·PostingList:**

对Posting的List集合包装，保存所有文档中的各个Term的信息。给出了add、get、indexOf、contains、remove等方法的实现。

该模块最主要的功能是建立Document相关的数据存储结构，然后再建立倒排索引便于后续检索操作。

1. **过滤器模块**

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所示。

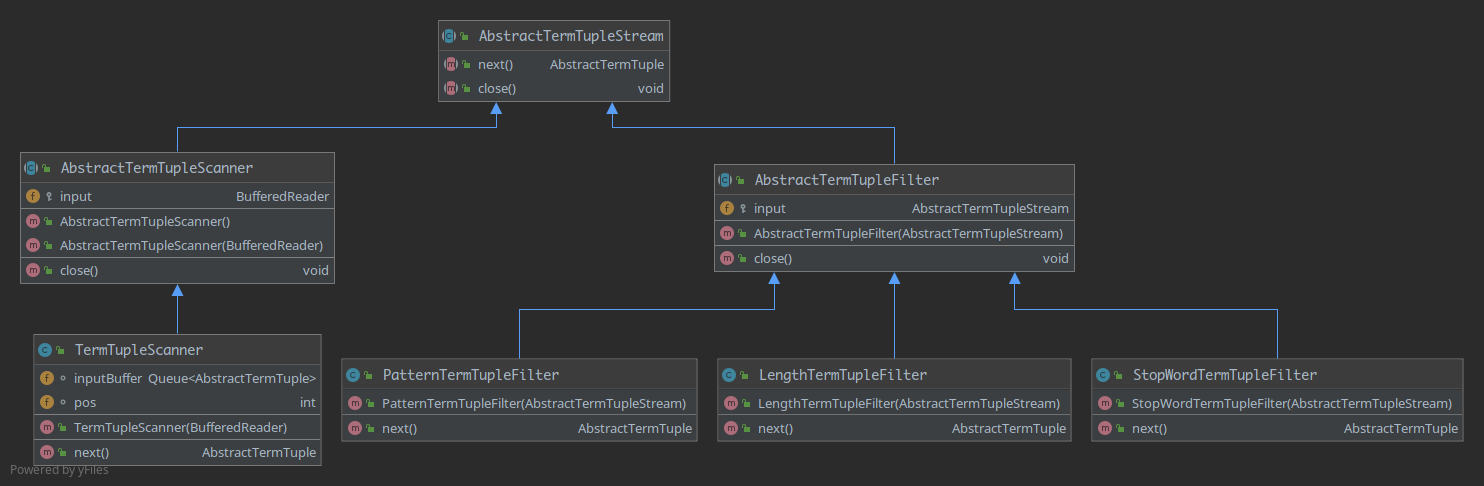


图 3 parse包的UML图

下面是对parse.impl包中实现对具体类对解释：

**·LengthTermTupleFilter:**

通过Config中的TERM\_FILTER\_MAXLENGTH和TERM\_FILTER\_MINLENGTH的参数来判断单词长度是否合乎规范，不合规范的单词丢弃。

**·PatternTermTupleFilter:**

通过Config中的TERM\_FILTER\_PATTERN参数进行正则表达式的匹配，不符合的单词丢弃。

**·StopWordTermTupleFilter:**

通过StopWords类中的STOP\_WORDS字段来判断单词是否在停用词之中，如果是的话就丢弃单词。

**·TermTupleScanner:**

对TermTuple进行扫描读取，其中主要的数据成员有：curPos（记录单词出现的位置）、tempParts(存放读取到的内容)。next（）方法中读取文档是readLine方法，一行一行的读取，但是取出时是一个单词一个单词逐个取出，所以就需要临时存放多余的内容。因而在调用next（）方法时就需要进行一个判空判断来进行不同的读取操作。在读取时还需要注意更新curPos字段，并且还要注意要根据Config中的IGNORE\_CASE参数来判断是否要转换大小写。

该模块主要的功能是对TermTuple的读取和过滤功能的实现。

1. **检索模块**

该模块主要实现了检索功能，检索过程有三个步骤：检索、保存检索结果、对检索结果进行排序。检索相关模块的UML图如图 4 所示。

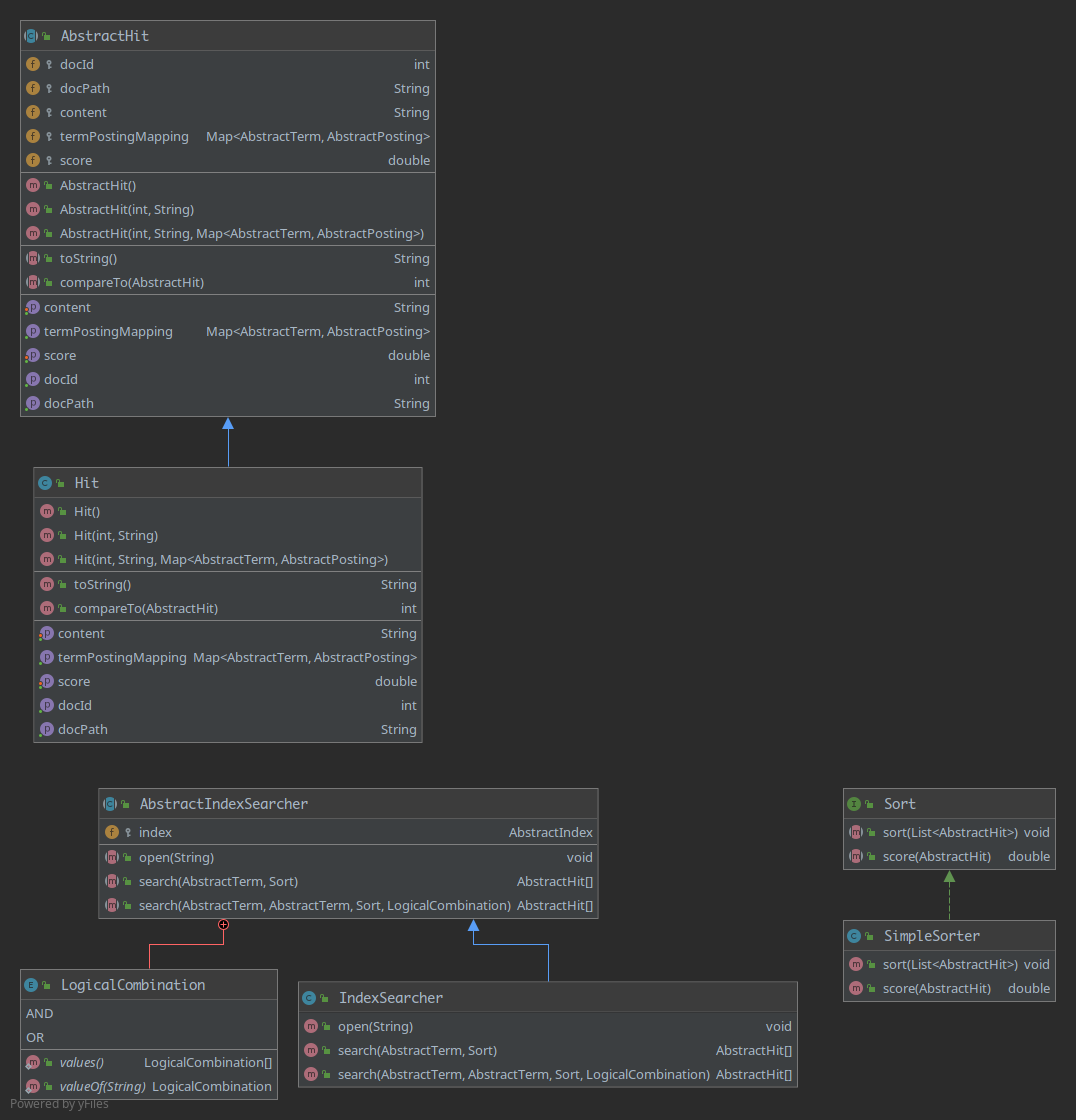


图 4 query包的UML图

下面是对query.impl包中实现对具体类对解释：

**·Hit:**

对检索结果进行保存，主要的数据成员有：docId（文档doc的id）、docPath（文档的绝对路径）、content（文档原文的内容）、termPostingMapping（命中的单词term和对应的posting的键值对映射，方便后续计算文档得分）、score（记录命中文档的得分），检索结果返回Hit数组。

**·IndexSearcher:**

根据构建好的索引对词条进行检索，数据成员index用于保存要使用的索引。其中主要的方法有：open方法将索引文件实例加载到index中。search方法则是对一个词/两个词的检索；对于一个词的检索则是直接使用index的search方法，利用返回的postingList构建Hit数组，随后对Hit数组进行排序得到结果；对于两个词的检索则是对两个词分别检索后针对结合规则对结果进行合并，或者筛选出重合的部分，利用SimpleSorter排序后返回Hit数组。

**·SimpleSorter:**

对命中得分进行排序，其中score方法用于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。sort方法则对传入对命中结果集合根据文档得分进行排序。

1. **软件开发**

开发环境：Parallels Desktop虚拟机Windows11系统

开发平台：Intellij IDEA Ultimate 2024.1.1

本次实验中使用的JDK均为jdk17版本。

1. **软件测试**

自动测试截图如图5所示，测试结果out-put如图6所示。其中测试用例106个，完成106个，跳过0个。

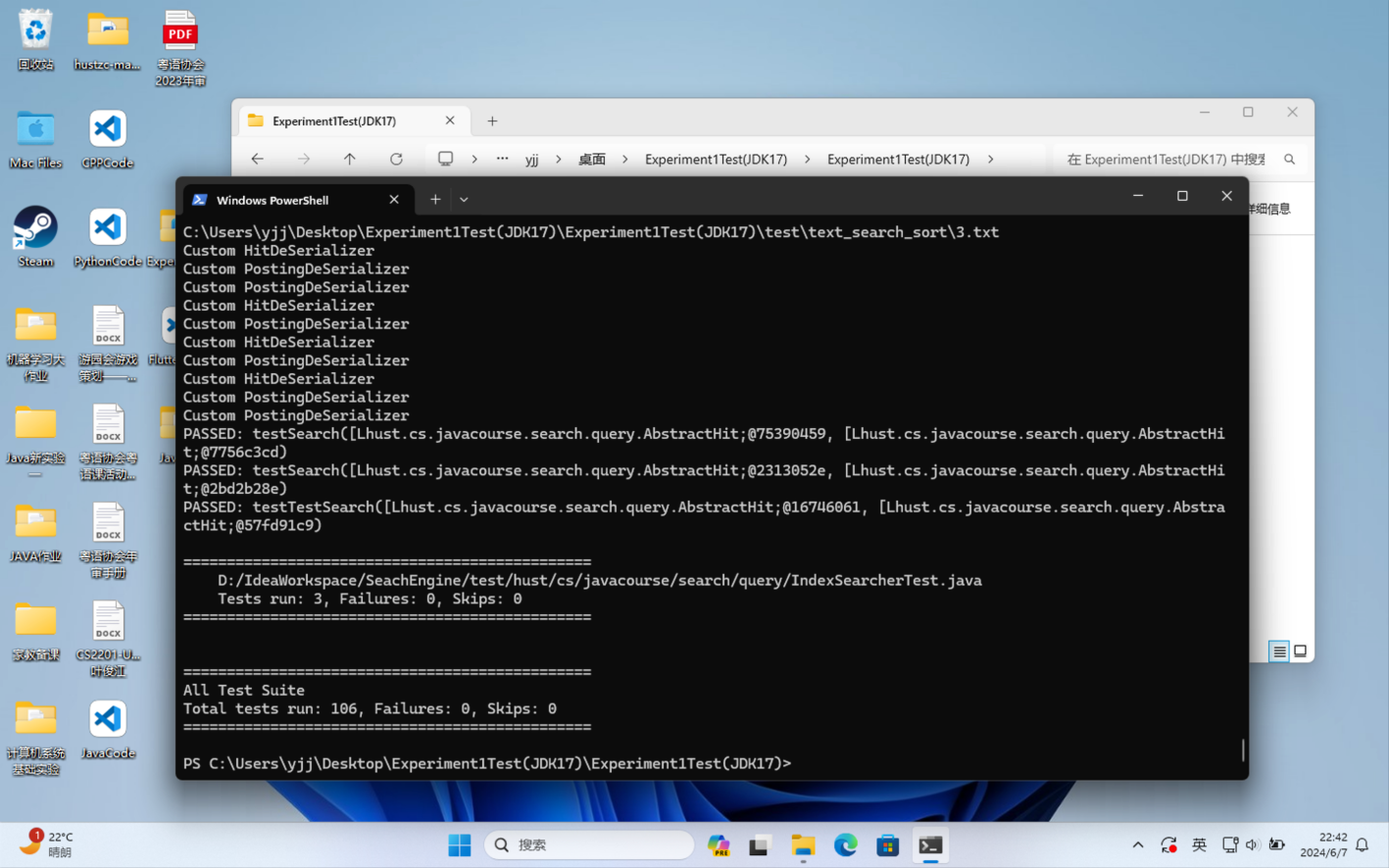


图 5 test.bat运行结果截图

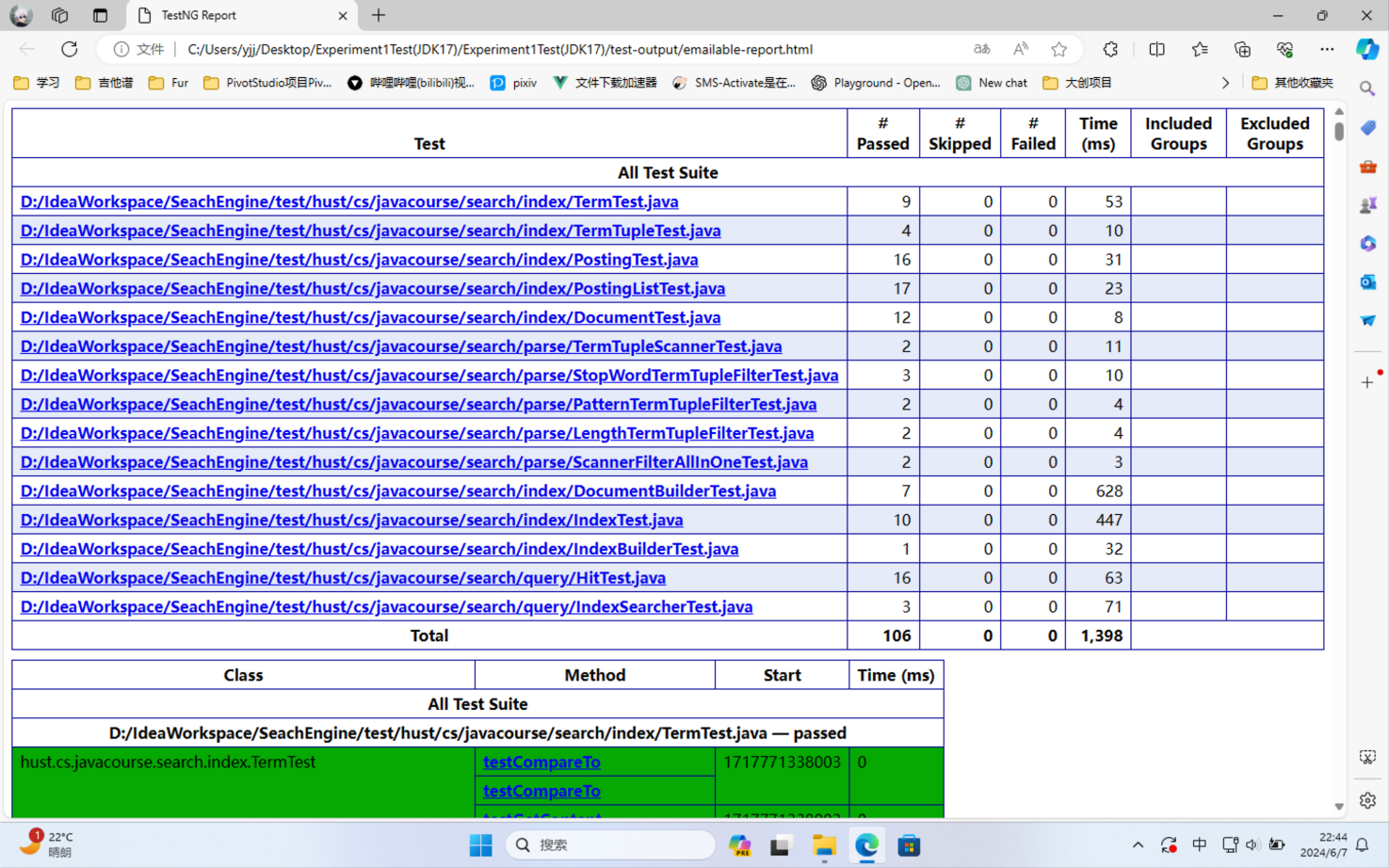


图 6 测试输出结果

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赋分制进行打分，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**

（1）对异常记录不完善：完善日志报告的记录。

（2）移植性不强：目前只适用英文。

1. **过程和体会**
2. **遇到的主要问题和解决方法**
3. 刚开始在mac上开发时有sh文件格式以及权限控制的问题，在测试包中的test.sh 的文件在使用时存在zsh:no such file的报错，因为没有给test.sh文件赋读写权限，需要在 terminal中输入”chmod +x test.sh”来给test.sh文件设置权限；再其次就是格式问题，在修 改完权限后仍然不能运行测试文件，因为test.sh初始为doc格式，我们需要通过vim指 令进入test.sh后输入”set ff=unix”指令来将test.sh转换成unix格式才能继续执行shell指令。
4. 测试时发现存在run<106的情况，即没跑完测试集，后经检查发现是目录存在中文 的原因，将测试包放入不存在中文的目录下便正常跑完测试集。
5. 本人发现我写的代码在MacOS下使用test.sh文件来测试就是 run:106,failure:4,skip:0，但是在Windows系统下使用test.bat文件来测试就完美过关，一 时不能确定是哪里出现问题，现在依旧不清楚问题何在，现已使用虚拟机来测试。
6. **课程设计的体会**

本次Java实验给我感觉和实际开发工作项目很相像，我也是第一次接触这种大项目，起初和大部分同学一样也是非常迷惘，不过在老师和同学的指点下也是顺利的完成了实验。

完成实验后我感觉收获满满，对抽象类和接口有了更深刻的理解，对各个类之间的关系有了更好的把控，并且对文件处理更加熟悉，而且其中涉及到vim以及shell的问题也是给了我新的知识层面的覆盖，对我日后的学习有非常大的帮助。

1. **源码和说明**
2. **文件清单及其功能说明**

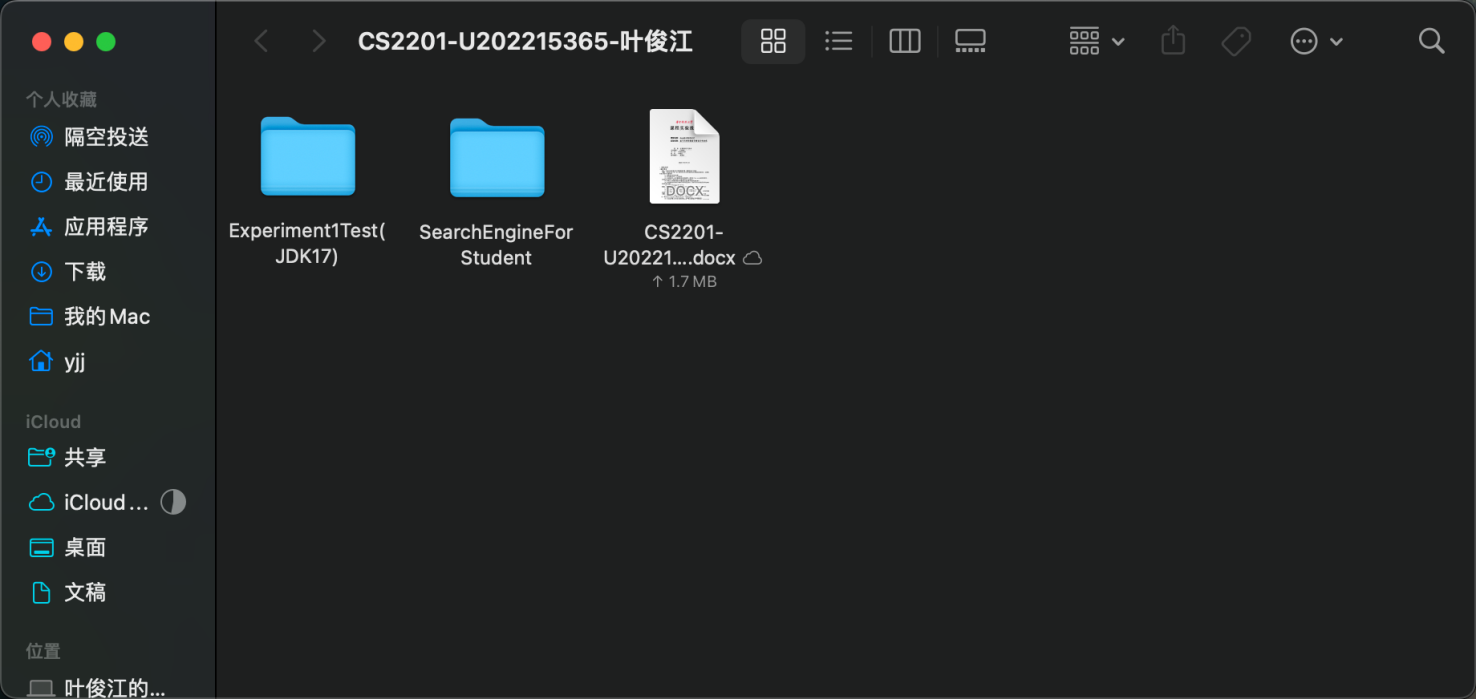


图 7 提交文件夹截图

如图7所示即为提交文件夹的内容，其中ExperimentTest（JDK17）为测试包，SearchEngineForStudent为实验源码包，doc文件即实验报告文档。

1. **用户使用说明书**
2. 打开Experiment1Test目录打开，其中有一个test.bat脚本文件，修改完JAVA\_HOME地址后运行即在test-output目录中得到自动测试的结果（目录不要有中文）。
3. 将SearchEngineForStudent在编译器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。
4. **源代码**

见附件中的SearchEngineForStudent文件夹。