Programming Task P01: Information Retrieval system using Apache Lucene 7.1.0

To create my information retrieval system I used the following classes:

IndexWriter – to create and maintain the index

```
IndexWriter iwriter = new IndexWriter(directory, config);
```

Directory – to store the index. I used FSDirectory implementation, since in the task it is required to store the index on file system.

```
Directory directory = FSDirectory.open(Paths.get(indexFolder));
```

Analyzer – to analyze text.

```
analyzer = new EnglishAnalyzer(stopWords);

CharArraySet stopWords = EnglishAnalyzer.getDefaultStopSet();
```

I used EnglishAnalyzer, since the task was to consider English language and use a stemmer (EnglishAnalyzer uses PorterStemFilter out of the box).

Document – a collection of **Fields** to be indexed.

```
Document doc = new Document();
doc.add(new TextField("title", doc1.get("title"), Field.Store.YES));
doc.add(new TextField("content", doc1.get("content"), Field.Store.YES));
doc.add(new TextField("path", doc1.get("path"), Field.Store.YES));
w.addDocument(doc);
```

IndexSearcher – to search the index.

```
DirectoryReader reader = DirectoryReader.open(dir);

IndexSearcher searcher = new IndexSearcher(reader);
```

In order to allow the user to choose the ranking model, I used **setSimilarity** method to let Searcher use either Vector Space model:

```
IndexWriterConfig config;

config = new IndexWriterConfig(analyzer);

config.setSimilarity(new ClassicSimilarity());
```

or Okapi BM25 model:

```
config.setSimilarity(new BM25Similarity());
```

QueryParser – to parse text query strings into an abstract **Query** class.

```
QueryParser qp = new MultiFieldQueryParser(fields, this.iwc.getAnalyzer());
...

Query = qp.parse(searchQuery);
```

TopDocs (contains references to top results of the search) and **ScoreDoc** (pointer to a single document and its score) – to display search results.

```
TopDocs docs = searcher.search(stemmedQuery, 10);
ScoreDoc[] scored = docs.scoreDocs;
```

Then the program prints top 10 results with their rank, title, summary, score and path.

```
System.out.println(results.size() + " hit(s):");

        for (HashMap<String, String> result : results) {
            System.out.println(result.get("rank") + ". " + result.get("title"));
            System.out.println("Rank: " + result.get("rank"));
            System.out.println("Score: " + result.get("score"));
            System.out.println("Summary: " + result.get("summary"));
            System.out.println("Path: " + result.get("path"));
            System.out.println("---------------");
        }
```

I used **score** method to get score values and **Highlighter** class to create a summary for each document.

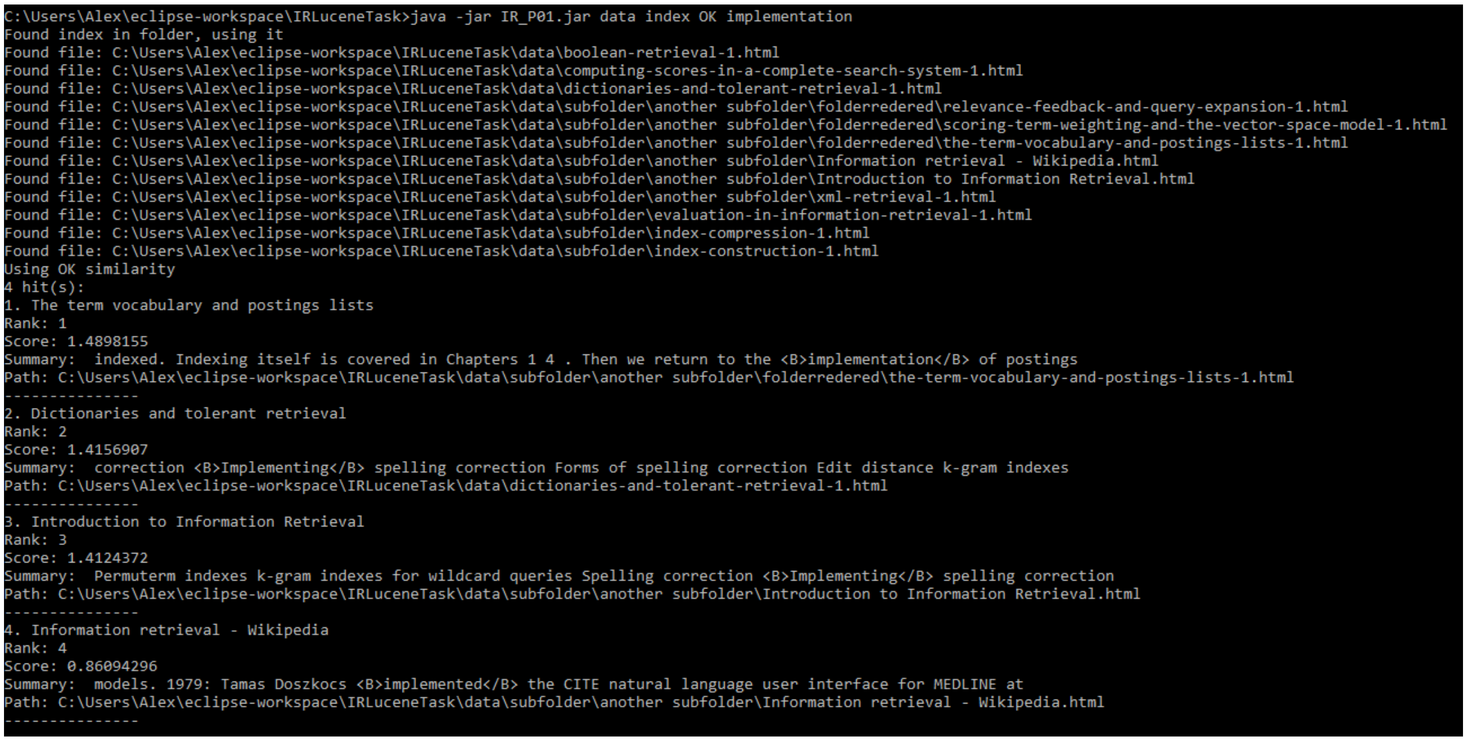
```
SimpleHTMLFormatter htmlFormatter = new SimpleHTMLFormatter();
Highlighter highlighter = new Highlighter(htmlFormatter, new QueryScorer(stemmedQuery));

TokenStream stream = TokenSources.getAnyTokenStream(reader, docid, "contents", analyzer);

String[] frags = highlighter.getBestFragments(stream, text, 10);
```

In case an index already exists in the index folder, the program uses it instead of creating a new one (see comments in the code). The program only indexes *.html and *.htm files and ignores other file types, since it was required by the task (see comments in the code).

I used to Jsoup to parse HTML documents in order to extract text from title and body tags.

Use command line to run the program:

```
java -jar IR_P01.jar [path to document folder] [path to index folder] [VS/OK] [query]
```



```
C:\Users\Alex\eclipse-workspace\IRLuceneTask>java -jar IR_P01.jar data index OK implementation
Found index in folder, using it
Found file: C:\Users\Alex\eclipse-workspace\IRLuceneTask\data\boolean-retrieval-1.html
Found file: C:\Users\Alex\eclipse-workspace\IRLuceneTask\data\computing-scores-in-a-complete-search-system-1.html
Found file: C:\Users\Alex\eclipse-workspace\IRLuceneTask\data\dictionaries-and-tolerant-retrieval-1.html
Found file: C:\Users\Alex\eclipse-workspace\IRLuceneTask\data\subfolder\another subfolder\folderredered\relevance-feedback-and-query-expansion-1.html
Found file: C:\Users\Alex\eclipse-workspace\IRLuceneTask\data\subfolder\another subfolder\folderredered\scoring-term-weighting-and-the-vector-space-model-1.html
Found file: C:\Users\Alex\eclipse-workspace\IRLuceneTask\data\subfolder\another subfolder\folderredered\the-term-vocabulary-and-postings-lists-1.html
Found file: C:\Users\Alex\eclipse-workspace\IRLuceneTask\data\subfolder\another subfolder\Information retrieval - Wikipedia.html
Found file: C:\Users\Alex\eclipse-workspace\IRLuceneTask\data\subfolder\another subfolder\Introduction to Information Retrieval.html
Found file: C:\Users\Alex\eclipse-workspace\IRLuceneTask\data\subfolder\another subfolder\xml-retrieval-1.html
Found file: C:\Users\Alex\eclipse-workspace\IRLuceneTask\data\subfolder\evaluation-in-information-retrieval-1.html
Found file: C:\Users\Alex\eclipse-workspace\IRLuceneTask\data\subfolder\index-compression-1.html
Found file: C:\Users\Alex\eclipse-workspace\IRLuceneTask\data\subfolder\index-construction-1.html
Using OK similarity
4 hit(s):
1. The term vocabulary and postings lists
Rank: 1
Score: 1.4898155
Summary:  indexed. Indexing itself is covered in Chapters 1 4 . Then we return to the <B>implementation</B> of postings
Path: C:\Users\Alex\eclipse-workspace\IRLuceneTask\data\subfolder\another subfolder\folderredered\the-term-vocabulary-and-postings-lists-1.html
---------------
2. Dictionaries and tolerant retrieval
Rank: 2
Score: 1.4156907
Summary:  correction <B>Implementing</B> spelling correction Forms of spelling correction Edit distance k-gram indexes
Path: C:\Users\Alex\eclipse-workspace\IRLuceneTask\data\dictionaries-and-tolerant-retrieval-1.html
---------------
3. Introduction to Information Retrieval
Rank: 3
Score: 1.4124372
Summary:  Permuterm indexes k-gram indexes for wildcard queries Spelling correction <B>Implementing</B> spelling correction
Path: C:\Users\Alex\eclipse-workspace\IRLuceneTask\data\subfolder\another subfolder\Introduction to Information Retrieval.html
---------------
4. Information retrieval - Wikipedia
Rank: 4
Score: 0.86094296
Summary:  models. 1979: Tamas Doszkocs <B>implemented</B> the CITE natural language user interface for MEDLINE at
Path: C:\Users\Alex\eclipse-workspace\IRLuceneTask\data\subfolder\another subfolder\Information retrieval - Wikipedia.html
---------------
```