PROGRAM 7

FILE RANKING

AIM

Write a program to generate a ranking for a given set of files to a given query.

PROGRAM

```
import org.apache.lucene.queryParser.ParseException;
import org.pdfbox.pdmodel.PDDocument;
import org.pdfbox.util.PDFTextStripper;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

import org.apache.lucene.analysis.standard.StandardAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.document.Field;
import org.apache.lucene.index.IndexWriter;
import org.apache.lucene.index.IndexWriterConfig;
import org.apache.lucene.index.Term;
import org.apache.lucene.store.FSDirectory;
import org.apache.lucene.util.Version;

import org.apache.lucene.index.IndexReader;
import org.apache.lucene.queryParser.QueryParser;
import org.apache.lucene.search.*;

public class SimpleSearch {
        // location where the index will be stored.
        private static final String INDEX_DIR = "src/main/resources/index";
        private static final int DEFAULT_RESULT_SIZE = 100;

        public static void main(String[] args) throws IOException, ParseException {
            File folder = new File("src/resources");
            File[] listOfFiles = folder.listFiles();

              for (int i = 0; i < listOfFiles.length; i++) {
                if (listOfFiles[i].isFile()) {


          File pdfFile = new File("src/resources/"+listOfFiles[i].getName());
          IndexItem pdfIndexItem = index(pdfFile);

          // creating an instance of the indexer class and indexing the items
          Indexer indexer = new Indexer(INDEX_DIR);
```

```
              indexer.index(pdfIndexItem);

               indexer.close();
                    }
                    else
                    {
                         System.out.println("NO files found For Indexing");
                    }
             }
               // creating an instance of the Searcher class to the query the index
               Searcher searcher = new Searcher(INDEX_DIR);
               System.out.println("Enter your Query for searching");
               Scanner a=new Scanner(System.in);
               String data=a.nextLine();
               List<IndexItem> result = searcher.findByContent(data, DEFAULT_RESULT_SIZE);
               prints(result);


               searcher.close();
             }

             //Extract text from PDF document
             public static IndexItem index(File file) throws IOException {
               PDDocument doc = PDDocument.load(file);
               String content = new PDFTextStripper().getText(doc);
               doc.close();
               return new IndexItem((long)file.getName().hashCode(), file.getName(), content);
             }

            //Print the results
             private static void prints( List<IndexItem> result) {
                    System.out.println(result);
                   boolean length=result.isEmpty();
                  if(length)
                          System.out.println("NO DOCUMENTS FOUND");

                  else
                           System.out.println("The document Retrived with the search keyword");

             }

          }

class Indexer {
         private IndexWriter writer;

   public Indexer(String indexDir) throws IOException {
       // create the index
       if(writer == null) {
       writer = new IndexWriter(FSDirectory.open(
             new File(indexDir)), new IndexWriterConfig(Version.LUCENE_36, new
StandardAnalyzer(Version.LUCENE_36)));
```

```
        }
    }

    /**
      * This method will add the items into index
     * @return
      */
    public void index(IndexItem indexItem) throws IOException {

        // deleting the item, if already exists
        writer.deleteDocuments(new Term(IndexItem.ID, indexItem.getId().toString()));

        Document doc = new Document();

        doc.add(new Field(IndexItem.ID, indexItem.getId().toString(), Field.Store.YES,
Field.Index.NOT_ANALYZED));
        doc.add(new Field(IndexItem.TITLE, indexItem.getTitle(), Field.Store.YES,
Field.Index.ANALYZED));
        doc.add(new Field(IndexItem.CONTENT, indexItem.getContent(), Field.Store.YES,
Field.Index.ANALYZED));

        // add the document to the index
        writer.addDocument(doc);

    }

    /**
      * Closing the index
      */
    public void close() throws IOException {
        writer.close();
    }
}

class IndexItem {
            private Long id;
    private String title;
    private String content;

    public static final String ID = "id";
    public static final String TITLE = "title";
    public static final String CONTENT = "content";

    public IndexItem(Long id, String title, String content) {
        this.id = id;
        this.title = title;
        this.content = content;
    }

    public IndexItem(long parseLong, String title2, int parseInt) {
                    // TODO Auto-generated constructor stub
            }
        public Long getId() {
        return id;
    }

    public String getTitle() {
        return title;
    }

    public String getContent() {
        return content;
    }

    @Override
    public String toString() {
        return "IndexItem{" +
                "id=" + id +
                ", title='" + title + '\'' +
                ", content='" + content + '\'' +
                '}';
    }

}

class PDFIndexer {
        public IndexItem index(File file) throws IOException {
        PDDocument doc = PDDocument.load(file);
        String content = new PDFTextStripper().getText(doc);
        doc.close();
        return new IndexItem((long)file.getName().hashCode(), file.getName(), content);
    }

}

class Searcher {
        private IndexSearcher searcher;
    private QueryParser titleQueryParser;
    private QueryParser contentQueryParser;

    public Searcher(String indexDir) throws IOException {
        // open the index directory to search
        searcher = new IndexSearcher(IndexReader.open(FSDirectory.open(new File(indexDir))));
        StandardAnalyzer analyzer = new StandardAnalyzer(Version.LUCENE_36);



        // defining the query parser to search items by content field.
        contentQueryParser = new QueryParser(Version.LUCENE_36, IndexItem.CONTENT,
analyzer);
    }

    /**
```

```
     * This method is used to find the indexed items by the title.
     * @param queryString - the query string to search for
     */
   public List<IndexItem> findByTitle(String queryString, int numOfResults) throws
ParseException, IOException {
      // create query from the incoming query string.
      Query query = titleQueryParser.parse(queryString);
      // execute the query and get the results
      ScoreDoc[] queryResults = searcher.search(query, numOfResults).scoreDocs;

      List<IndexItem> results = new ArrayList<IndexItem>();
      // process the results
      for (ScoreDoc scoreDoc : queryResults) {
         Document doc = searcher.doc(scoreDoc.doc);
         results.add(new IndexItem(Long.parseLong(doc.get(IndexItem.ID)),
doc.get(IndexItem.TITLE), doc.get(IndexItem
               .CONTENT)));
      }

       return results;
   }
public List<IndexItem> findByContent(String queryString, int numOfResults) throws
ParseException, IOException {
      // create query from the incoming query string.
      Query query = contentQueryParser.parse(queryString);
       // execute the query and get the results
      ScoreDoc[] queryResults = searcher.search(query, numOfResults).scoreDocs;

      List<IndexItem> results = new ArrayList<IndexItem>();

      for (ScoreDoc scoreDoc : queryResults) {
         Document doc = searcher.doc(scoreDoc.doc);
         results.add(new IndexItem(Long.parseLong(doc.get(IndexItem.ID)),
doc.get(IndexItem.TITLE),queryString));
      }

       return results;
   }

   public void close() throws IOException {
      searcher.close();
   }
}
```

RESULT

The program is successfully implemented and required output is obtained.

OUTPUT

```
Enter your Query for searching
Hello
[IndexItem{id=-908356333, title='2Hell.pdf', content='Hello'}, IndexItem{id=1157535200,
title='3Hel.pdf', content='Hello'}, IndexItem{id=-2134654811, title='1Hello.pdf', content='Hello'},
IndexItem{id=216988876, title='SamplePDF.pdf', content='Hello'}]
The document Retrived with the search keyword
```