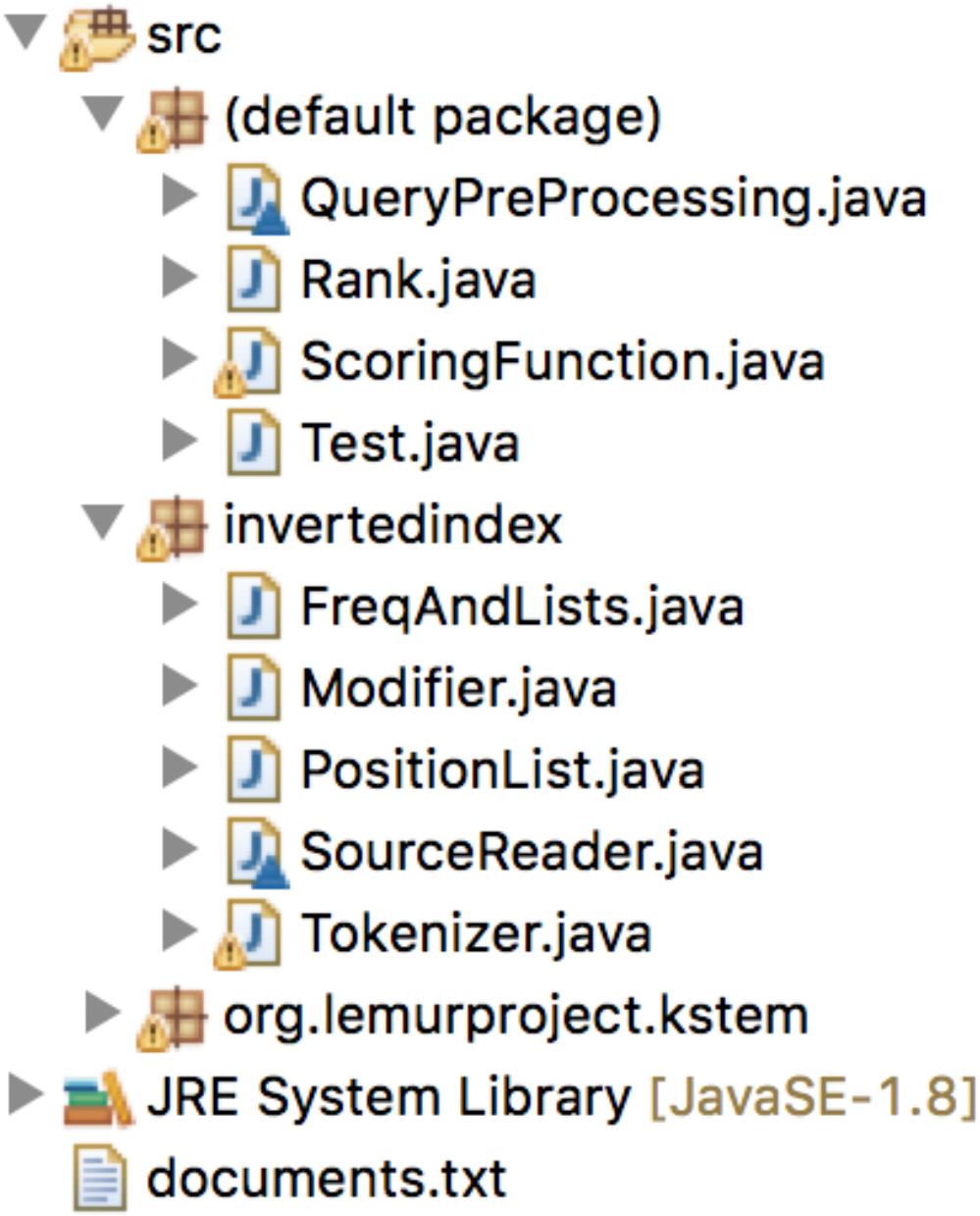
CSC849 Homework2 - Positional Inverted Index & Free-text Queries with Proximity Operator

Student name: Xuan Zhang
Student ID: 916409525
Email: xzhang8@mail.sfsu.edu

When I run it, the folder structure is like below:



There are 3 packages.
The 1st one is the package for stemmer: org.lemurproject.kstem.

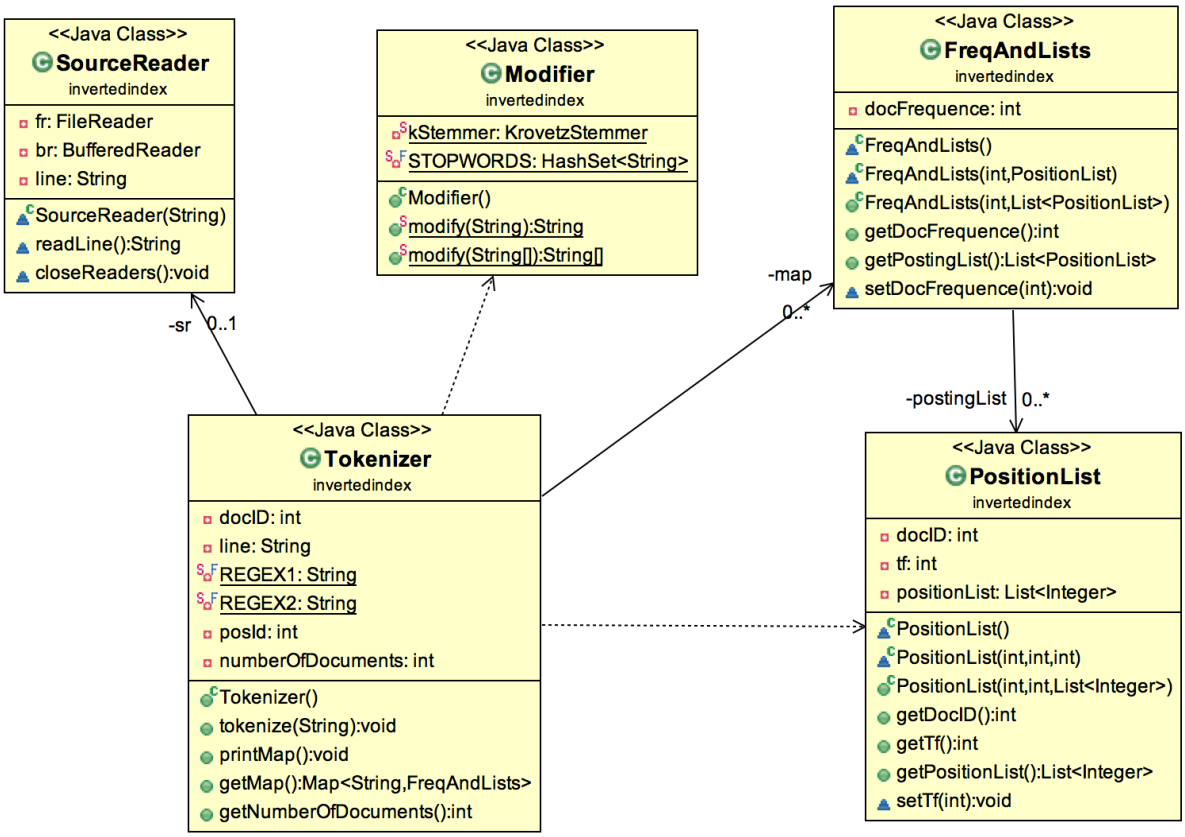
The 2nd package is inverted index package which contains 5 files.
The structure for the positional inverted index is like below:
TreeMap:
Key Value
Term FreqAndLists (the class I defined)
FreqAndLists is like:

Document Frequence I List<PositionList>

PositionList is a class I defined, which is like

<document ID I Term frequence I List<Integer> (Positions)>

The diagram for this package:



SourceReader:

This class is to help Tokenizer to read the file.Since in the java there might be some "try catch" cases that may throw exceptions as well as all readers should be closed finally,I define a seperate class to deal with all of these situations, then in other parts of the programs, we don't need to pay attention to those stuff.

Modifier:

This class is to modify (case normaloziation, stemming) for the given term or term array.

Tokenizer:

This class is to tokenize the given file and stem them to produce the dictionary and the posting lists (stored in a TreeMap).

FreqAndLists:

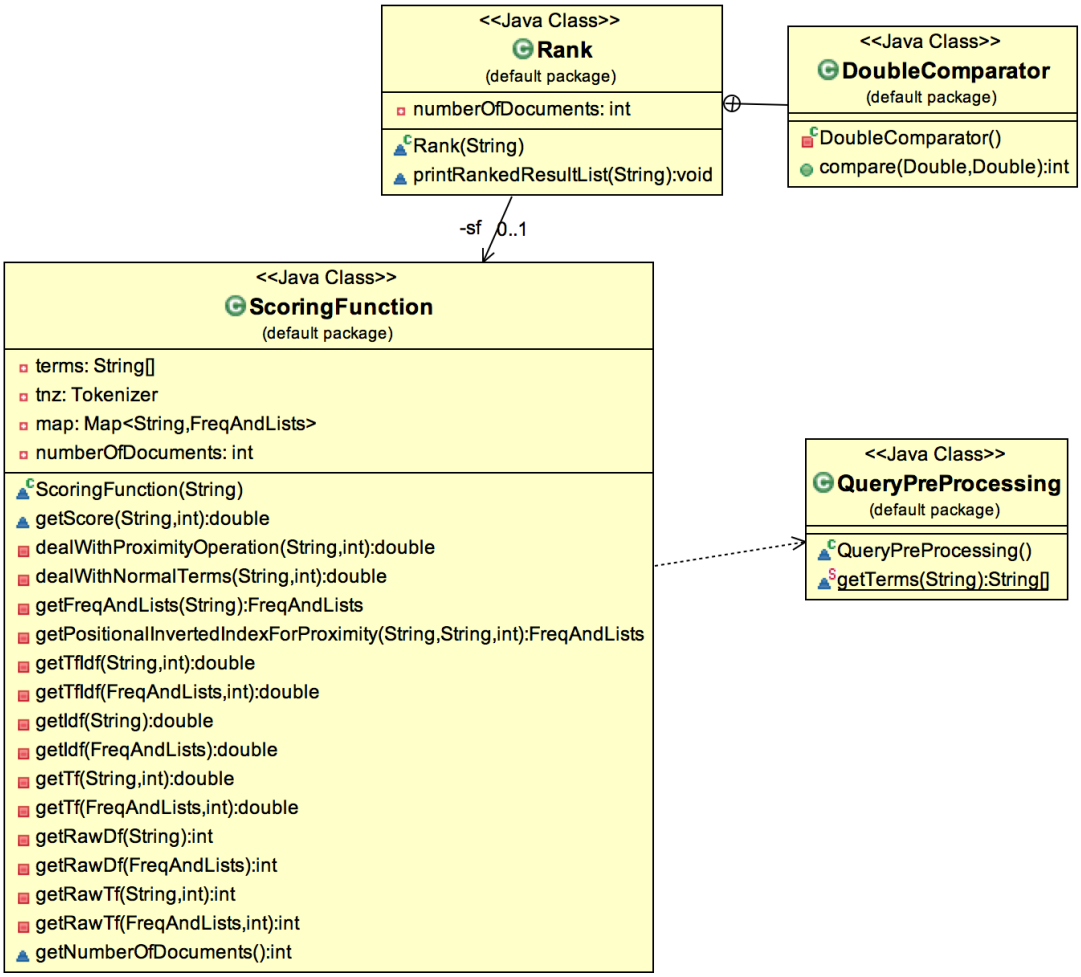
This class is to define the value class for the TreeMap.Because I use the String as the key, and the <doc freq | posting List> as the value in the TreeMap.

PositionList:

This class is to define the class for the PositionList.Because I use [docID | tf | position list] as each list in the posting list.

The 3rd one (default package) is for ranking the documents, which contains 3 classes.

The diagram is like below:



Rank:

This class is to rank all documents with a non-0 score using the class ScoringFunction and print the result to a file.

ScoringFunction:

This class is to get the score for a given query and docID in tf-idf way.

QueryPreProcessing:

This class is to help to deal with to pre-processing the query.

When I ran it , in the default package I add an additional class with the main function, like below, and would generate different files with the results by the query.

QueryPreProcessing. ScoringFunction.jav Rank.java Test.java

```java

public class Test {
    public static void main(String [] args) {
        Rank rank = new Rank("documents.txt");
        String query = "nexus like love happy";
        rank.printRankedResultList(query);
    }
}

```