`Executor` method of `MyIndexer` class is executed from the “main” method to run the program.

```
MyIndexer progAssig2 = new MyIndexer();
progAssig2.executor();
```

java -jar irproject-Prog-Ass-2-jar-with-dependencies.jar

This assignment will introduce the idea behind indexing and posting lists. Implement the following:
a) Implement assignment 4.1

Inverted “index” `Directory` object is created. `IndexWriterConfig` object is created using `Analyzer` with “standard” and “lower case” filters. `addDocument` method on `IndexWriter` Object is invoked to add the given document. When adding the documents `IndexOptions.DOCS_AND_FREQS_AND_POSITIONS_AND_OFFSETS` is specified.

Inverted index output - term : doc freq → postings list.

[a:1]->[2]
[always:1]->[6]
[be:1]->[3]
[berlin:3]->[4]->[5]->[6]
[exciting:1]->[6]
[girl:1]->[2]
[in:1]->[4]
[is:4]->[1]->[2]->[4]->[6]
[not:1]->[3]
[or:1]->[3]
[she:2]->[2]->[4]
[sunny:3]->[1]->[2]->[5]
[to:1]->[3]
[today:2]->[1]->[4]

Query search
`IndexReader` object is used to open the dictionary/index created.`search` method of `IndexSearcher` object is used to search terms in the dictionary.

Query string: sunny AND excited
Found 0 search_hits.

b) Print posting list (without skip pointers) for all terms indexed in above step for terms “sunny” and “to”. Print format [tokenname:total frequency:doc frequency]->[docid:frequency:[positions]]->[docid:frequency:[positions]]

`Terms` and `TermsEnum` to get and iterate over all the entries in “title” Field of index using `getTermVector` Index reader method. For term each entry the required stats are calculated by the `output` custom method.

format: [tokenname:total frequency:doc frequency]->[docid:frequency:[positions]]->[docid:frequency:[positions]]

[sunny:3:3]->[1:1:[3]]->[2:1:[4]]->[5:1:[1]]
[to:2:1]->[3:2:[1]]->[3:2:[5]]