NLP and IR

Building your first Search Engine with Lucene

Aliaksei Severyn

University of Trento, Italy

March 07, 2013

Plan for the lab

- Introduction to Lucene Search Engine
- Lucene concepts
- Hands-on experience with indexing and searching
  - HelloWorld example
- Using search engine to retrieve answer passages for Question Answering system
  - Indexing and searching 50k of QA corpus

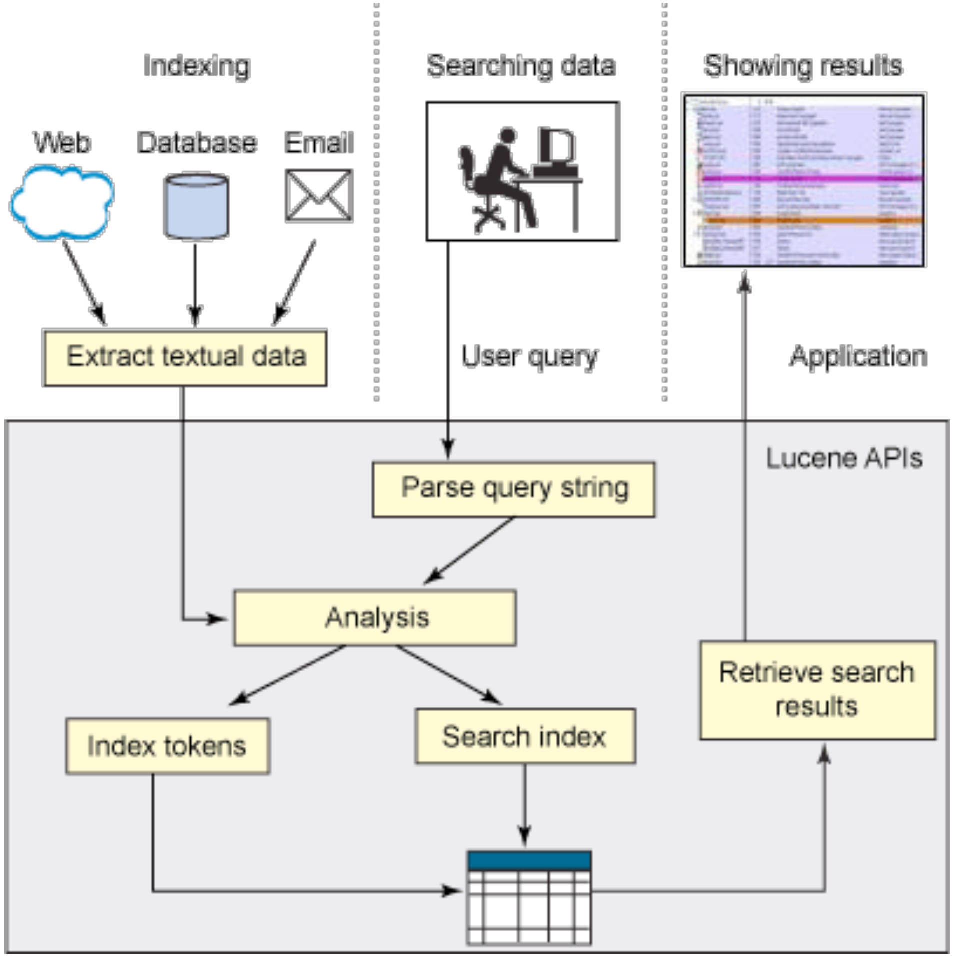
What is Lucene

- software library for search
- open source
- not a complete application
- set of java classes
- active user and developer communities
- widely used in research and also in production

High level overview

- Lucene is a full-text search library
- Designed to add search to your application
- Maintains a full-text index.
- Searches the index and returns results ranked by either the relevance to the query (or by an arbitrary field such as a document's last modified date.)

Typical architecture of a Lucene search app



Our first HelloWorld app with Lucene

- Create an in-memory index
- Add a few documents
- Construct a query
- Search an index
- Display results

Setting up our first example

Download Lucene sources and binary from:

http://www.apache.org/dist/lucene/java/3.5.0/

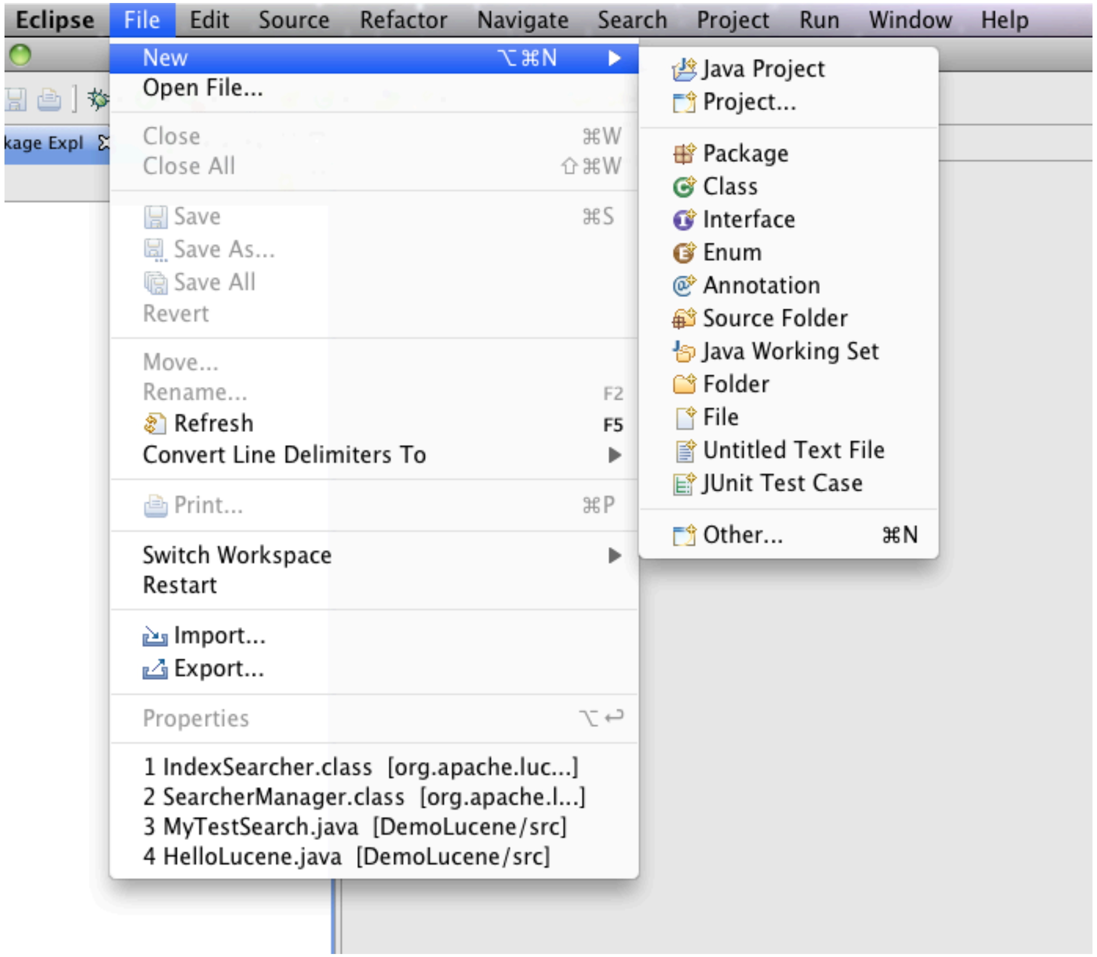
Or download everything from :

https://github.com/aseveryn/NLPIR-2013

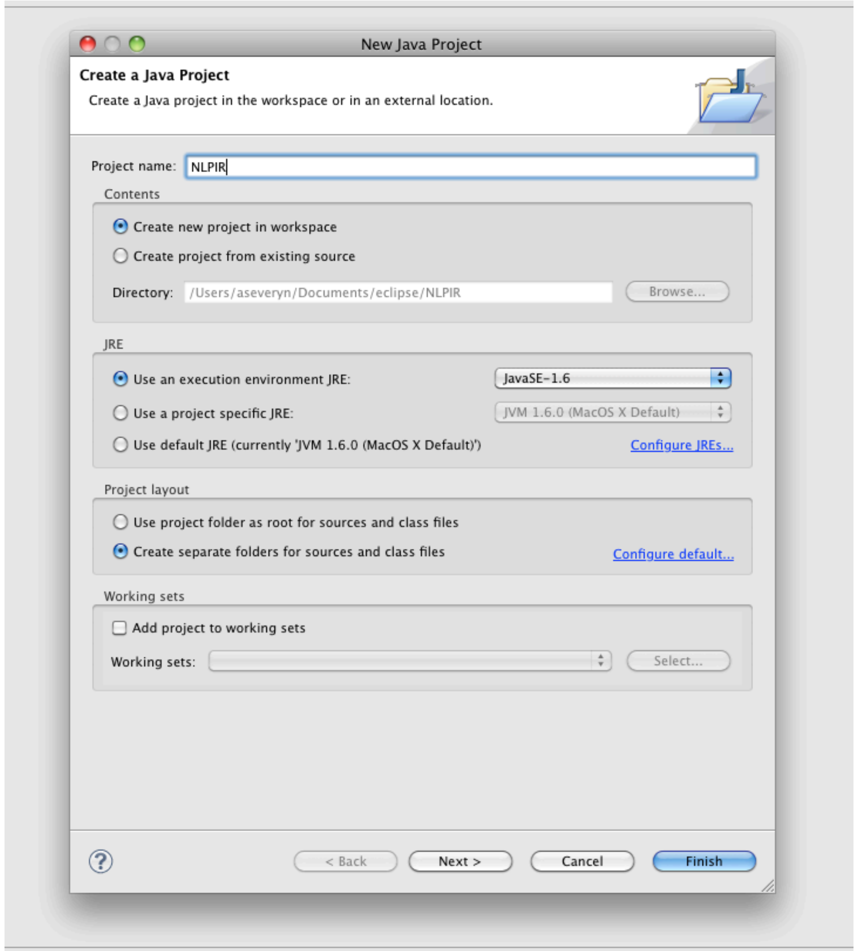
E.g. try the following in your terminal:

```
$ git clone https://github.com/aseveryn/NLPIR-2013.git
```

Create a new project



Name your project



New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: NLPIR

Contents

● Create new project in workspace

○ Create project from existing source

Directory: /Users/aseveryn/Documents/eclipse/NLPIR

JRE

● Use an execution environment JRE: JavaSE-1.6

○ Use a project specific JRE: JVM 1.6.0 (MacOS X Default)

○ Use default JRE (currently 'JVM 1.6.0 (MacOS X Default)')

Project layout

○ Use project folder as root for sources and class files

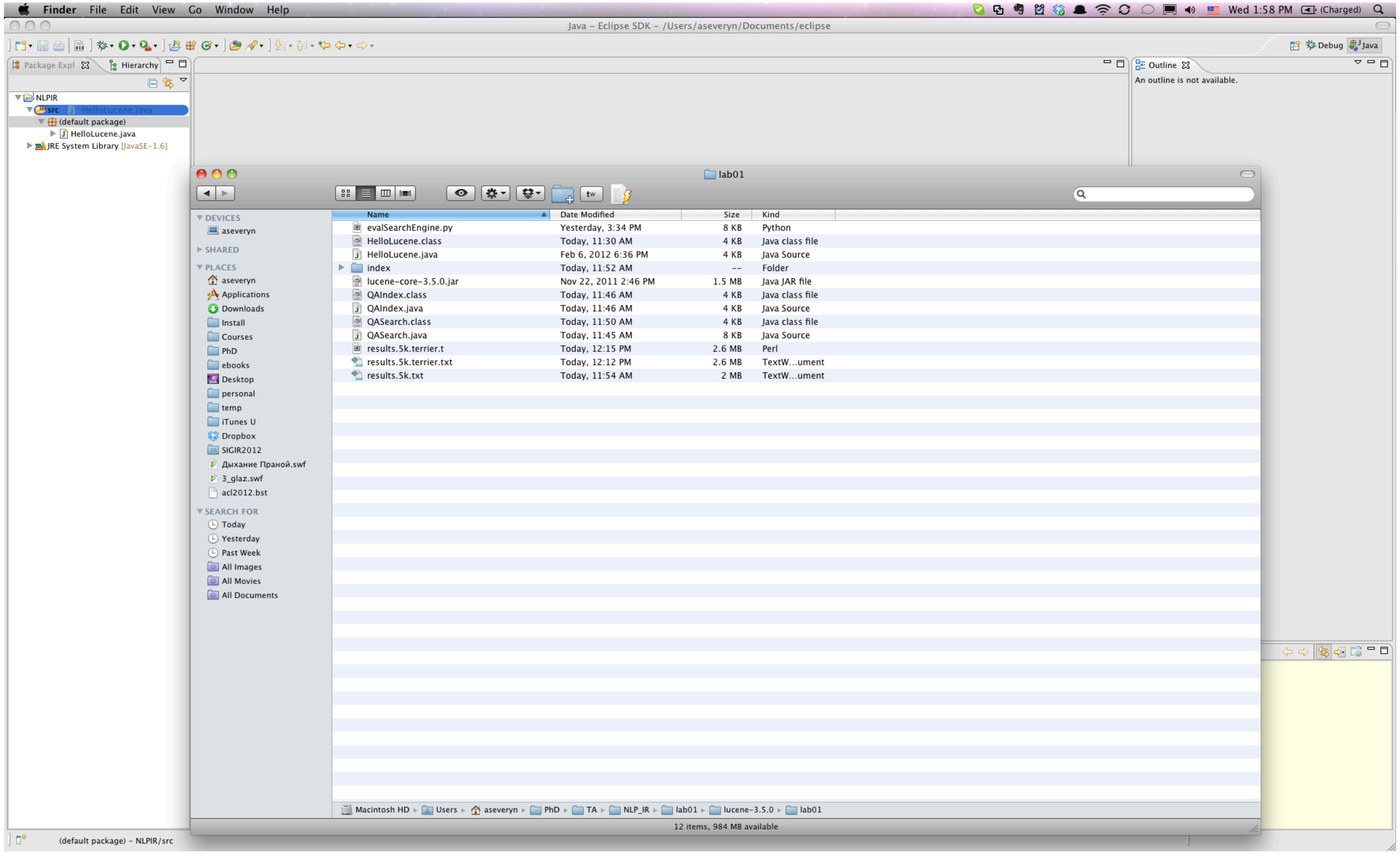
● Create separate folders for sources and class files

Working sets

□ Add project to working sets

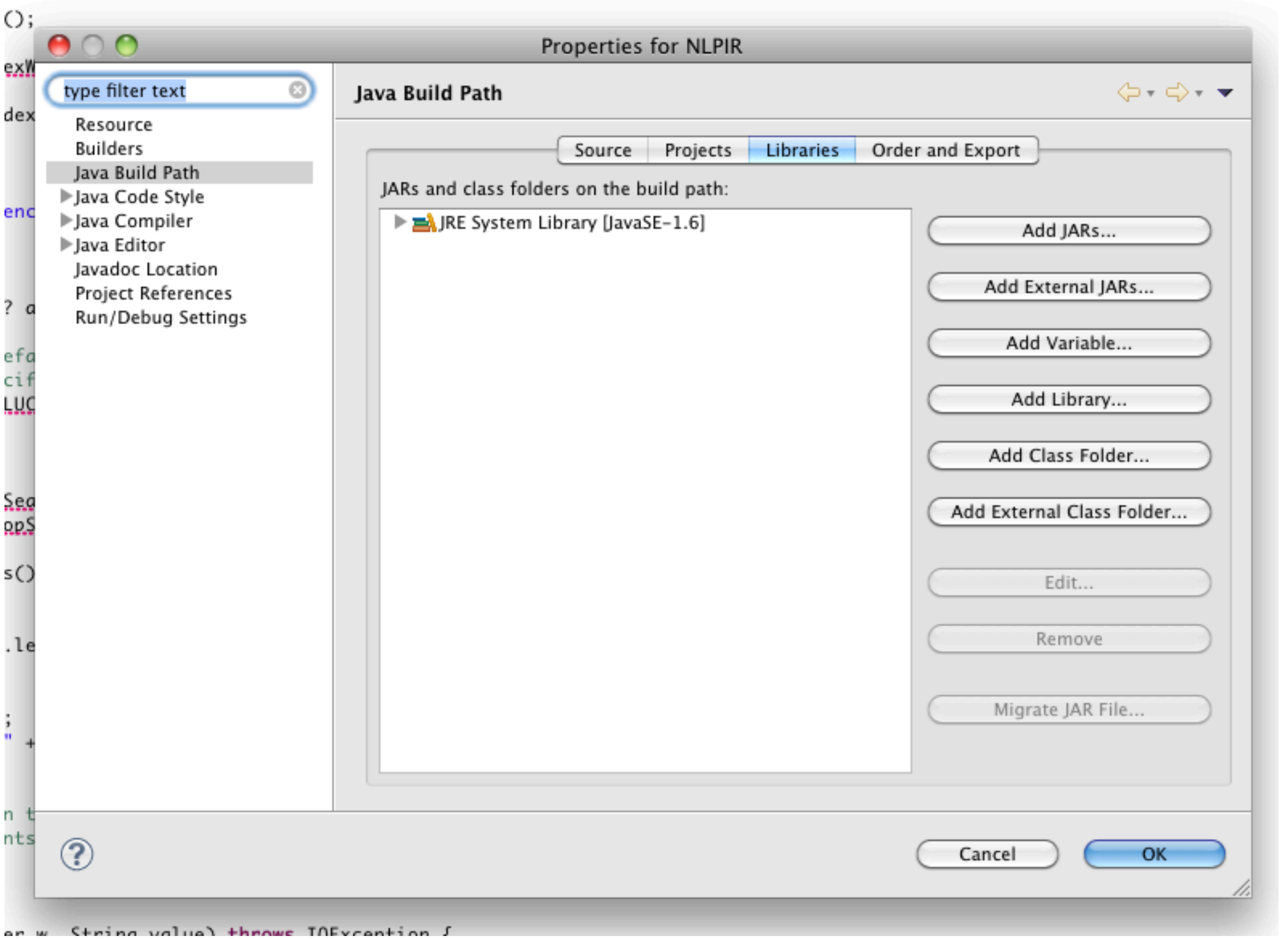
Working sets:

Drag HelloLucene.java to the src folder



Name	Date Modified	Size	Kind
evalSearchEngine.py	Yesterday, 3:34 PM	8 KB	Python
HelloLucene.class	Today, 11:30 AM	4 KB	Java class file
HelloLucene.java	Feb 6, 2012 6:36 PM	4 KB	Java Source
index	Today, 11:52 AM	--	Folder
lucene-core-3.5.0.jar	Nov 22, 2011 2:46 PM	1.5 MB	Java JAR file
QAIndex.class	Today, 11:46 AM	4 KB	Java class file
QAIndex.java	Today, 11:46 AM	4 KB	Java Source
QASearch.class	Today, 11:50 AM	4 KB	Java class file
QASearch.java	Today, 11:45 AM	8 KB	Java Source
results.5k.terrier.t	Today, 12:15 PM	2.6 MB	Perl
results.5k.terrier.txt	Today, 12:12 PM	2.6 MB	TextW...ument
results.5k.txt	Today, 11:54 AM	2 MB	TextW...ument

Add lucene-core-3.5.0.jar



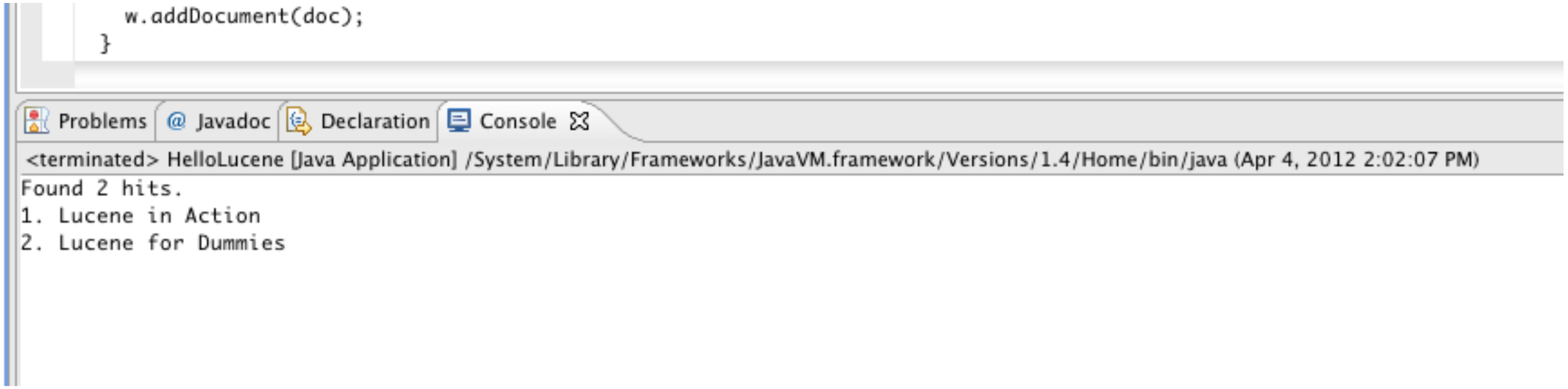
Properties for NLPIR

Java Build Path

JARs and class folders on the build path:

JRE System Library [JavaSE-1.6]

Run your first Lucene app!



```
        w.addDocument(doc);
    }
```

```
<terminated> HelloLucene [Java Application] /System/Library/Frameworks/JavaVM.framework/Versions/1.4/Home/bin/java (Apr 4, 2012 2:02:07 PM)
Found 2 hits.
1. Lucene in Action
2. Lucene for Dummies
```

Adding Lucene documentation to the project

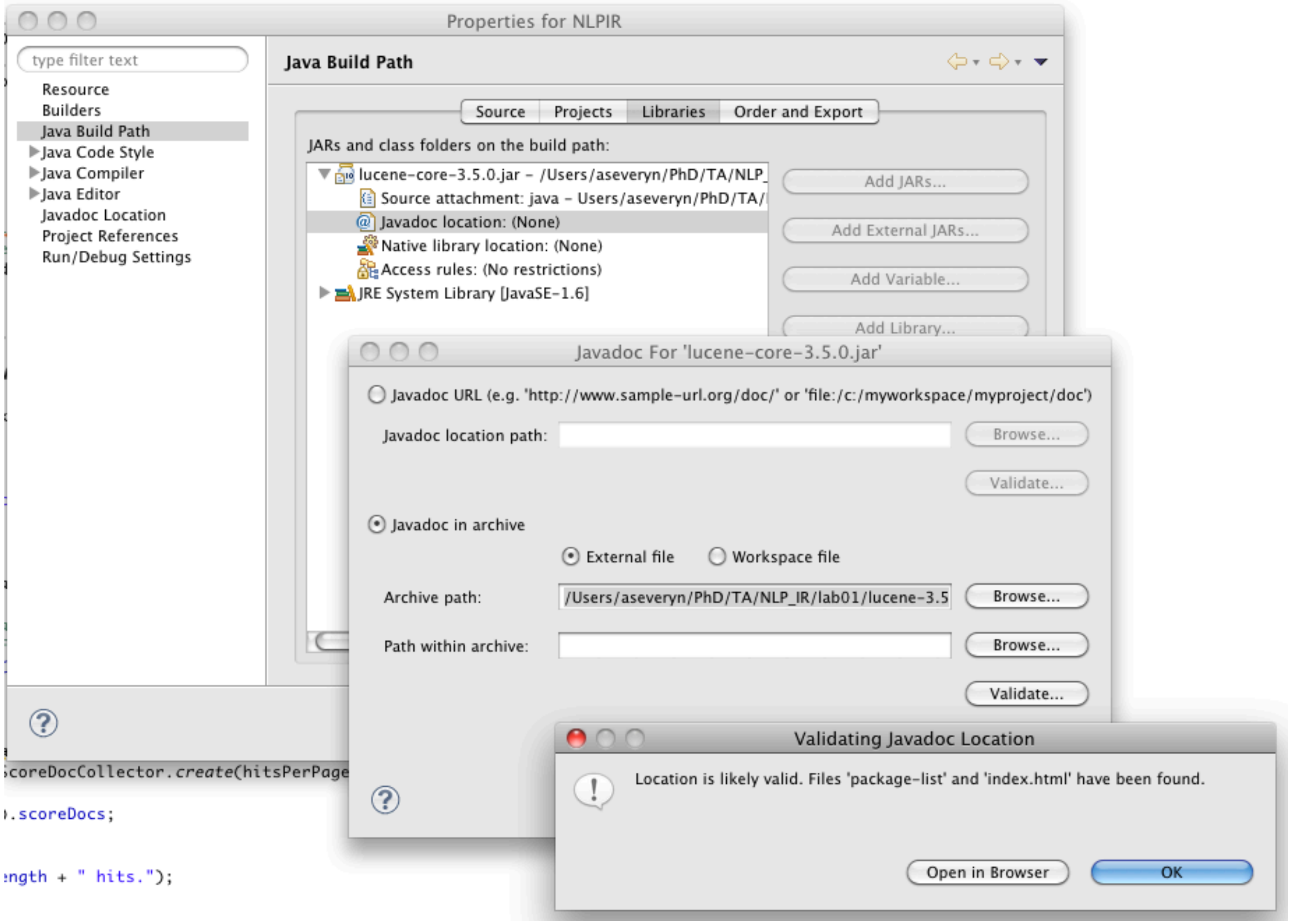
Go to Project properties->Libraries

Select lucene-core-3.5.0.jar

Select javadoc location

Locate lucene-core-3.5.0.jar

Adding javadoc



JARs and class folders on the build path:

- lucene-core-3.5.0.jar – /Users/aseveryn/PhD/TA/NLP_
  - Source attachment: java – Users/aseveryn/PhD/TA/
  - Javadoc location: (None)
  - Native library location: (None)
  - Access rules: (No restrictions)
- JRE System Library [JavaSE-1.6]

○ Javadoc URL (e.g. 'http://www.sample-url.org/doc/' or 'file:/c:/myworkspace/myproject/doc')

Javadoc location path:

● Javadoc in archive

● External file ○ Workspace file

Archive path: /Users/aseveryn/PhD/TA/NLP_IR/lab01/lucene-3.5

Path within archive:

Location is likely valid. Files 'package-list' and 'index.html' have been found.

```
coreDocCollector.create(hitsPerPage
.scoreDocs;
ngth + " hits.");
```

Setting the working environment

To be able to look at the Lucene internals:

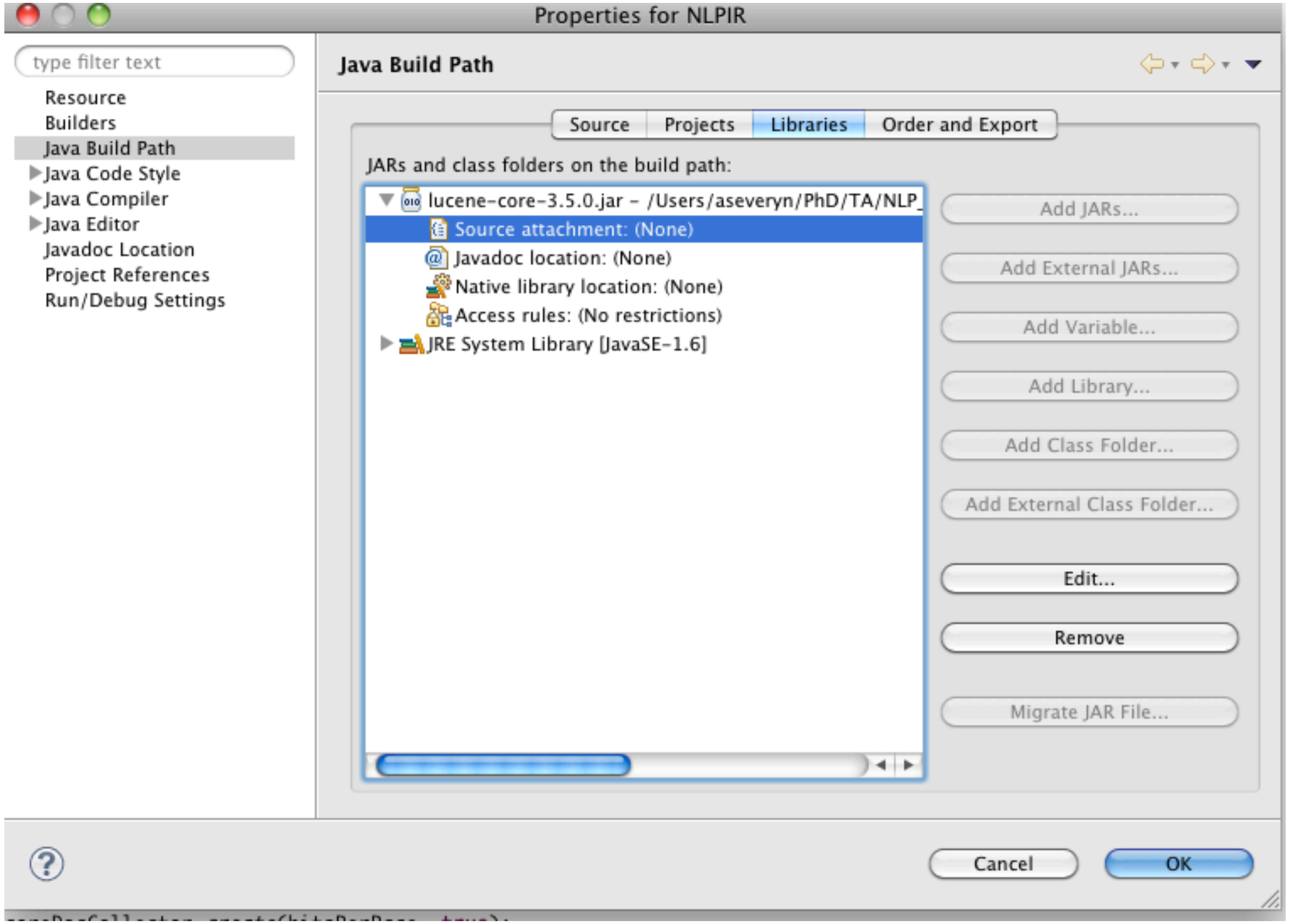
Go to Project properties->Libraries

Select lucene-core-3.5.0.jar

Select source attachment

Locate src folder

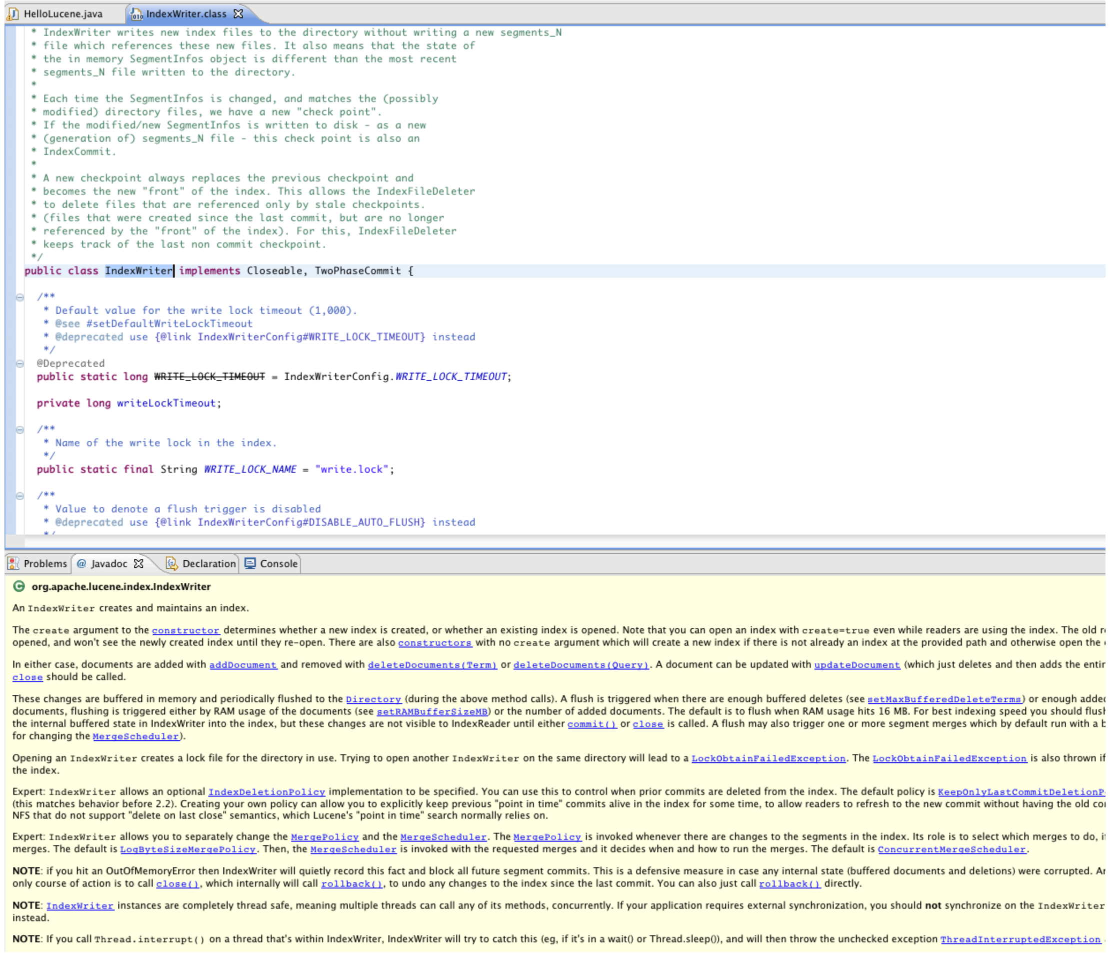
Adding sources to the JAR



JARs and class folders on the build path:

- lucene-core-3.5.0.jar - /Users/aseveryn/PhD/TA/NLP_
  - Source attachment: (None)
  - Javadoc location: (None)
  - Native library location: (None)
  - Access rules: (No restrictions)
- JRE System Library [JavaSE-1.6]

Now we can examine the sources and documentation



```
 * IndexWriter writes new index files to the directory without writing a new segments_N
 * file which references these new files. It also means that the state of
 * the in memory SegmentInfos object is different than the most recent
 * segments_N file written to the directory.
 *
 * Each time the SegmentInfos is changed, and matches the (possibly
 * modified) directory files, we have a new "check point".
 * If the modified/new SegmentInfos is written to disk - as a new
 * (generation of) segments_N file - this check point is also an
 * IndexCommit.
 *
 * A new checkpoint always replaces the previous checkpoint and
 * becomes the new "front" of the index. This allows the IndexFileDeleter
 * to delete files that are referenced only by stale checkpoints.
 * (files that were created since the last commit, but are no longer
 * referenced by the "front" of the index). For this, IndexFileDeleter
 * keeps track of the last non commit checkpoint.
 */
public class IndexWriter implements Closeable, TwoPhaseCommit {

  /**
   * Default value for the write lock timeout (1,000).
   * @see #setDefaultWriteLockTimeout
   * @deprecated use {@link IndexWriterConfig#WRITE_LOCK_TIMEOUT} instead
   */
  @Deprecated
  public static long ~~WRITE_LOCK_TIMEOUT~~ = IndexWriterConfig.WRITE_LOCK_TIMEOUT;

  private long writeLockTimeout;

  /**
   * Name of the write lock in the index.
   */
  public static final String WRITE_LOCK_NAME = "write.lock";

  /**
   * Value to denote a flush trigger is disabled
   * @deprecated use {@link IndexWriterConfig#DISABLE_AUTO_FLUSH} instead
```

org.apache.lucene.index.IndexWriter

An IndexWriter creates and maintains an index.

The create argument to the constructor determines whether a new index is created, or whether an existing index is opened. Note that you can open an index with create=true even while readers are using the index. The old r
opened, and won't see the newly created index until they re-open. There are also constructors with no create argument which will create a new index if there is not already an index at the provided path and otherwise open the

In either case, documents are added with addDocument and removed with deleteDocuments(Term) or deleteDocuments(Query). A document can be updated with updateDocument (which just deletes and then adds the entir
close should be called.

These changes are buffered in memory and periodically flushed to the Directory (during the above method calls). A flush is triggered when there are enough buffered deletes (see setMaxBufferedDeleteTerms) or enough added
documents, flushing is triggered either by RAM usage of the documents (see setRAMBufferSizeMB) or the number of added documents. The default is to flush when RAM usage hits 16 MB. For best indexing speed you should flush
the internal buffered state in IndexWriter into the index, but these changes are not visible to IndexReader until either commit() or close is called. A flush may also trigger one or more segment merges which by default run with a b
for changing the MergeScheduler).

Opening an IndexWriter creates a lock file for the directory in use. Trying to open another IndexWriter on the same directory will lead to a LockObtainFailedException. The LockObtainFailedException is also thrown if
the index.

Expert: IndexWriter allows an optional IndexDeletionPolicy implementation to be specified. You can use this to control when prior commits are deleted from the index. The default policy is KeepOnlyLastCommitDeletionP
(this matches behavior before 2.2). Creating your own policy can allow you to explicitly keep previous "point in time" commits alive in the index for some time, to allow readers to refresh to the new commit without having the old co
NFS that do not support "delete on last close" semantics, which Lucene's "point in time" search normally relies on.

Expert: IndexWriter allows you to separately change the MergePolicy and the MergeScheduler. The MergePolicy is invoked whenever there are changes to the segments in the index. Its role is to select which merges to do, i
merges. The default is LogByteSizeMergePolicy. Then, the MergeScheduler is invoked with the requested merges and it decides when and how to run the merges. The default is ConcurrentMergeScheduler.

NOTE: if you hit an OutOfMemoryError then IndexWriter will quietly record this fact and block all future segment commits. This is a defensive measure in case any internal state (buffered documents and deletions) were corrupted. A
only course of action is to call close(), which internally will call rollback(), to undo any changes to the index since the last commit. You can also just call rollback() directly.

NOTE: IndexWriter instances are completely thread safe, meaning multiple threads can call any of its methods, concurrently. If your application requires external synchronization, you should **not** synchronize on the IndexWriter
instead.

NOTE: If you call Thread.interrupt() on a thread that's within IndexWriter, IndexWriter will try to catch this (eg, if it's in a wait() or Thread.sleep()), and will then throw the unchecked exception ThreadInterruptedException

Basic concepts in Lucene Search Engine

- Indexing
- Documents
- Fields
- Searching
- Queries

Indexing

- Instead of searching the text directly it searches the index
- Uses **inverted index** - inverts a document-centric data structure (document->words) to a keyword-centric data structure (word->documents)

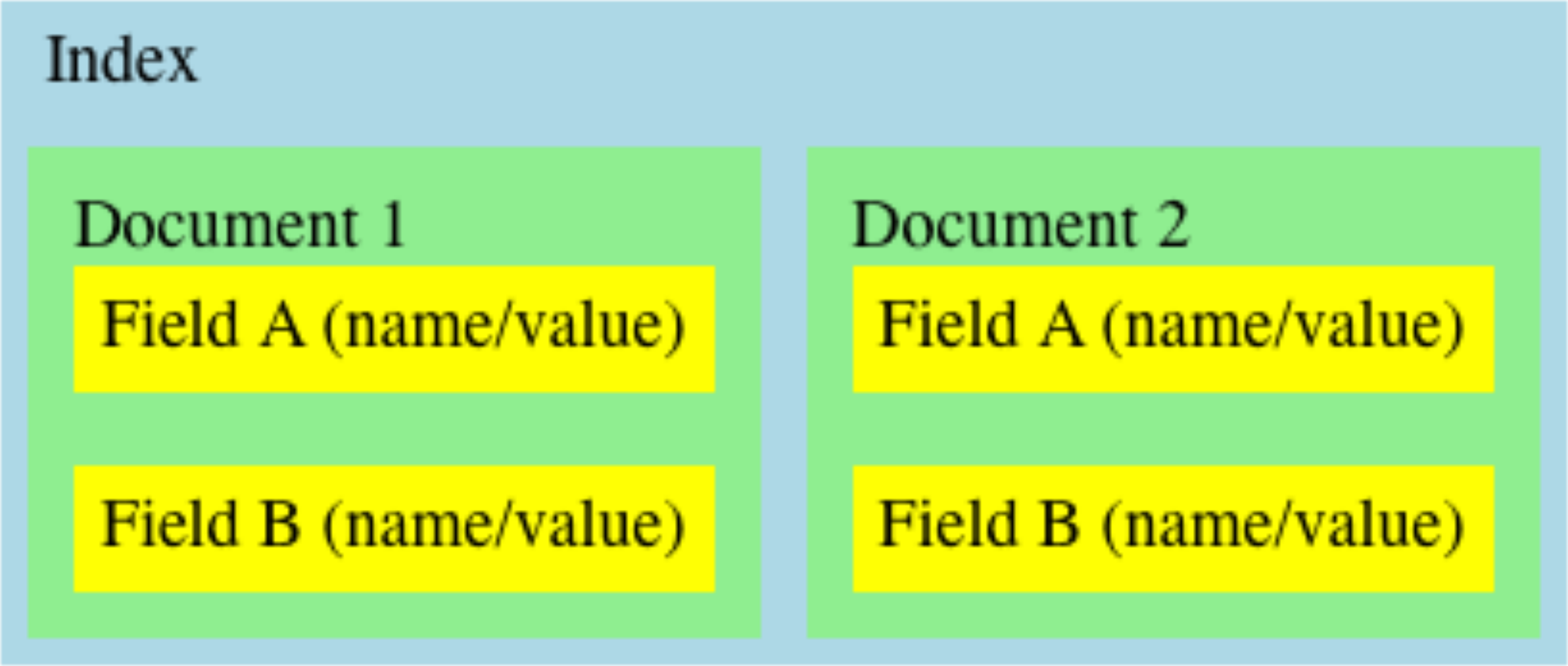
Documents

- In Lucene, a **Document** is the unit of search and index.
- An index consists of one or more Documents.
- **Indexing –** adding Documents to an IndexWriter
- **Searching -** retrieving Documents from an index via an IndexSearcher.

Fields

- A Document consists of one or more Fields.
- A Field is simply a name-value pair.
- For example, a Field commonly found in applications is *title*.
- Indexing in Lucene thus involves creating Documents of one or more Fields, and adding these Documents to an IndexWriter.

Documents and fields in Lucene



Index

Document 1

Field A (name/value)

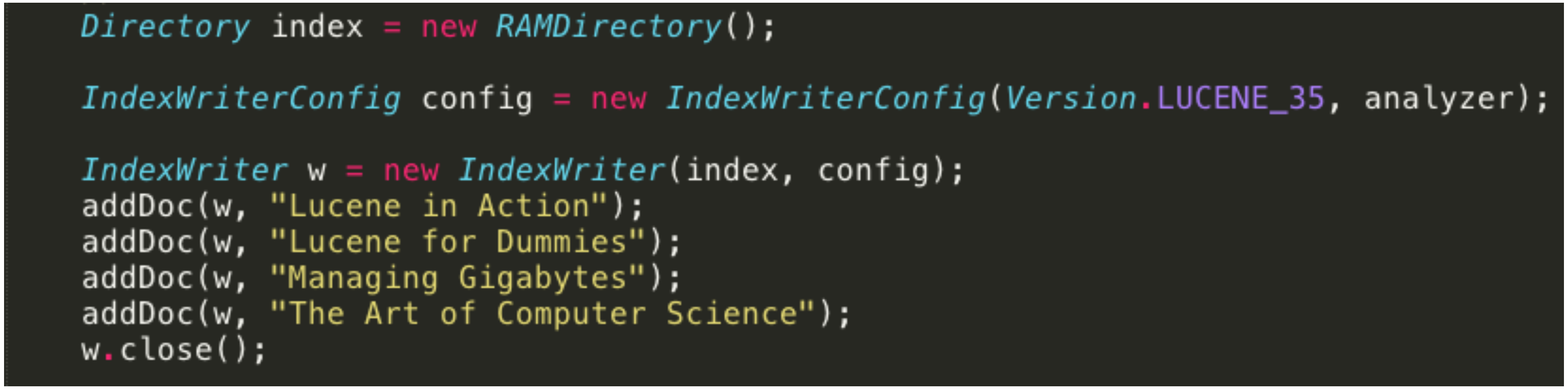
Field B (name/value)

Document 2

Field A (name/value)

Field B (name/value)

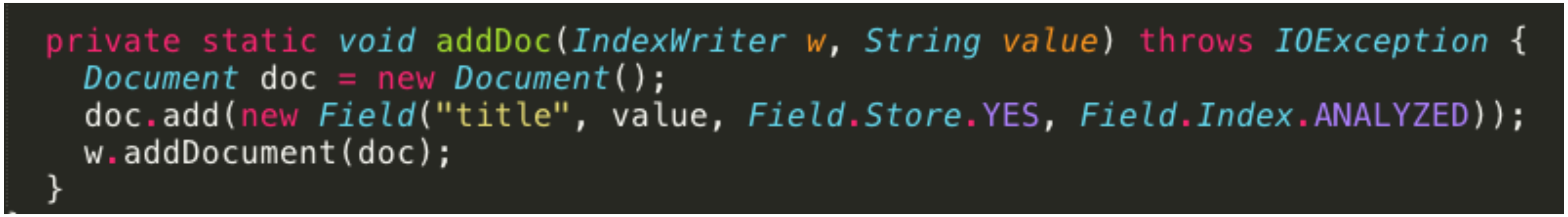
Adding documents to the index



```java
Directory index = new RAMDirectory();

IndexWriterConfig config = new IndexWriterConfig(Version.LUCENE_35, analyzer);

IndexWriter w = new IndexWriter(index, config);
addDoc(w, "Lucene in Action");
addDoc(w, "Lucene for Dummies");
addDoc(w, "Managing Gigabytes");
addDoc(w, "The Art of Computer Science");
w.close();
```



```java
private static void addDoc(IndexWriter w, String value) throws IOException {
  Document doc = new Document();
  doc.add(new Field("title", value, Field.Store.YES, Field.Index.ANALYZED));
  w.addDocument(doc);
}
```

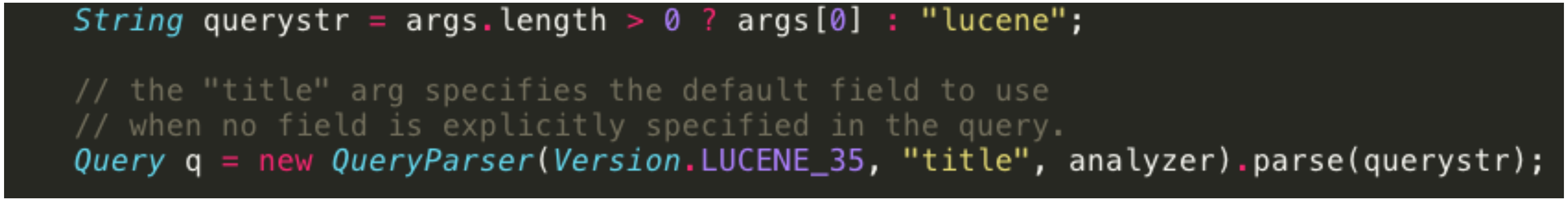
Queries

Lucene has its own mini-language for performing searches.

Allows the user to specify which field(s) to search on, which fields to give more weight to (boosting), the ability to perform boolean queries (AND, OR, NOT) and other functionality.

Query

We read the query from stdin, parse it and build a lucene Query out of it.



```
    String querystr = args.length > 0 ? args[0] : "lucene";

    // the "title" arg specifies the default field to use
    // when no field is explicitly specified in the query.
    Query q = new QueryParser(Version.LUCENE_35, "title", analyzer).parse(querystr);
```

Searching

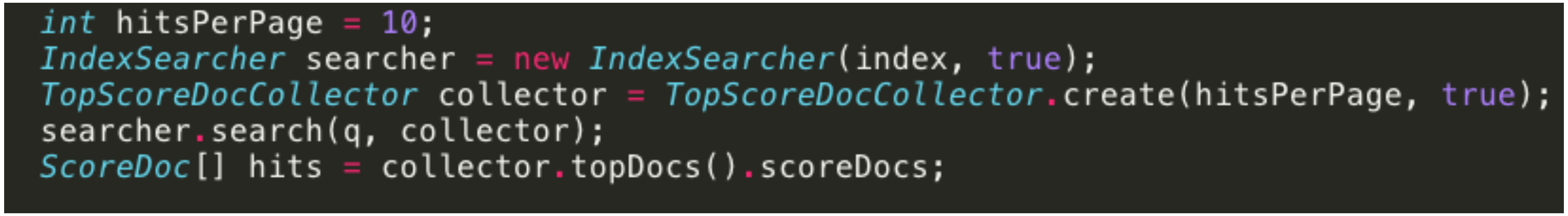
Searching requires an index to have already been built.

Very simple process:

- Create a **Query** (usually via a QueryParser)
- Handle this Query to an **IndexSearcher**
- Process a list of results

Searching

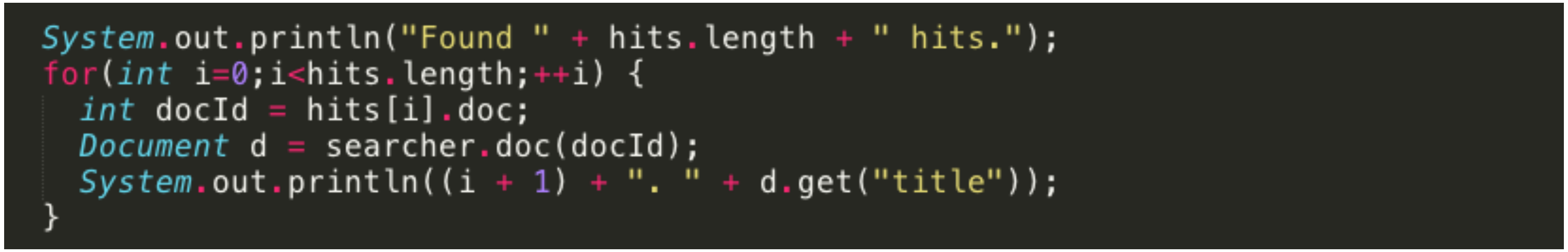
- Using the Query we create a Searcher to search the index.
- Then instantiate a TopScoreDocCollector to collect the top 10 scoring hits.



```
int hitsPerPage = 10;
IndexSearcher searcher = new IndexSearcher(index, true);
TopScoreDocCollector collector = TopScoreDocCollector.create(hitsPerPage, true);
searcher.search(q, collector);
ScoreDoc[] hits = collector.topDocs().scoreDocs;
```

Display of results

Now that we have results from our search, we display the results to the user.



```java
System.out.println("Found " + hits.length + " hits.");
for(int i=0;i<hits.length;++i) {
  int docId = hits[i].doc;
  Document d = searcher.doc(docId);
  System.out.println((i + 1) + ". " + d.get("title"));
}
```

Let’s get more practical

Build a Search Engine for answer passage retrieval in the Question Answering system

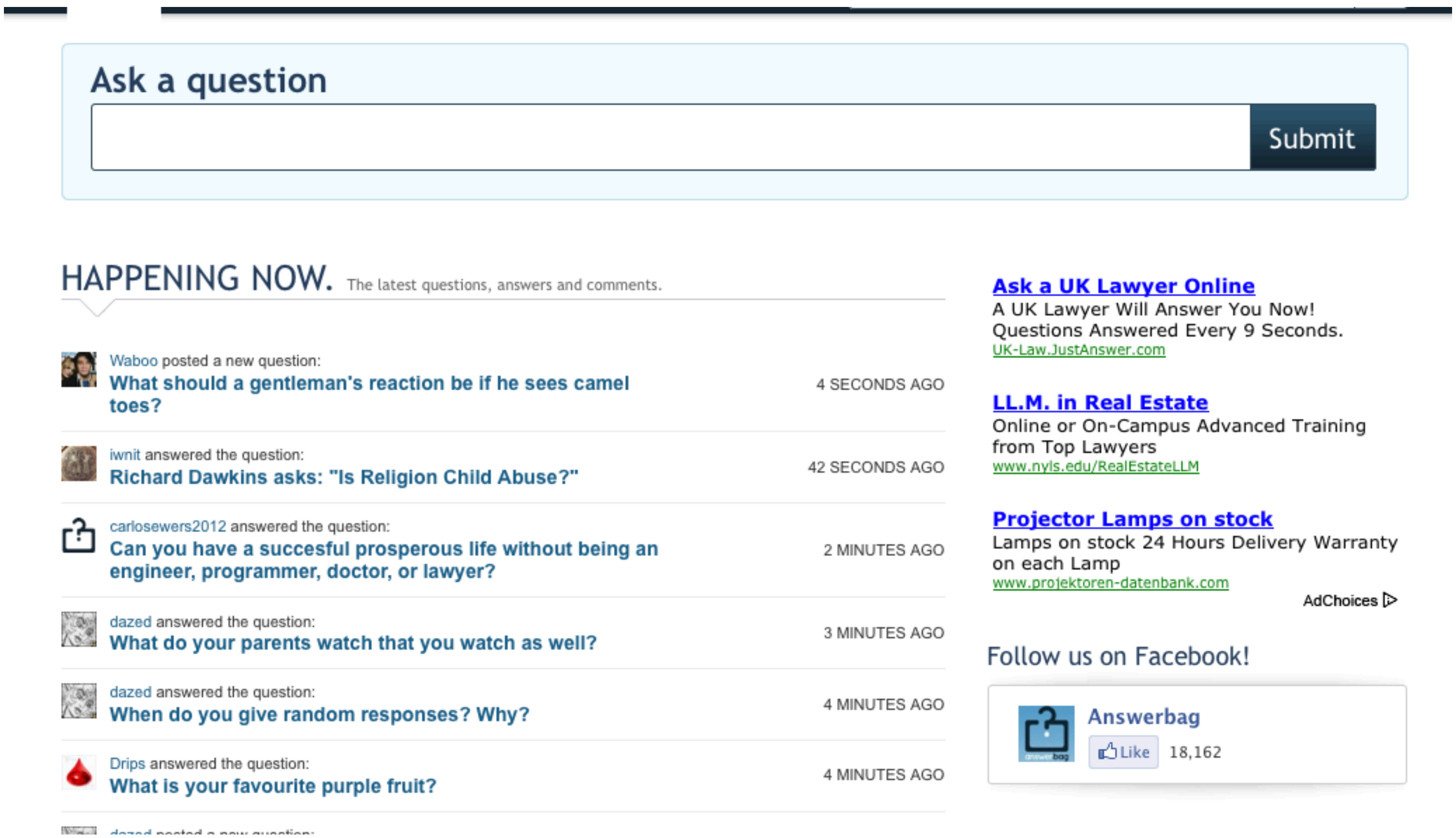
Use community QA site: Answerbag*

Use ~180k of automatically scraped question/answer pairs from over 20 categories

To reduce the amount of junk content focus only on professionally answered questions

http://www.answerbag.com/

QA system with AnswerBag data



Ask a question

HAPPENING NOW. The latest questions, answers and comments.

Waboo posted a new question:
What should a gentleman's reaction be if he sees camel toes? 4 SECONDS AGO

iwnit answered the question:
Richard Dawkins asks: "Is Religion Child Abuse?" 42 SECONDS AGO

carlosewers2012 answered the question:
Can you have a succesful prosperous life without being an engineer, programmer, doctor, or lawyer? 2 MINUTES AGO

dazed answered the question:
What do your parents watch that you watch as well? 3 MINUTES AGO

dazed answered the question:
When do you give random responses? Why? 4 MINUTES AGO

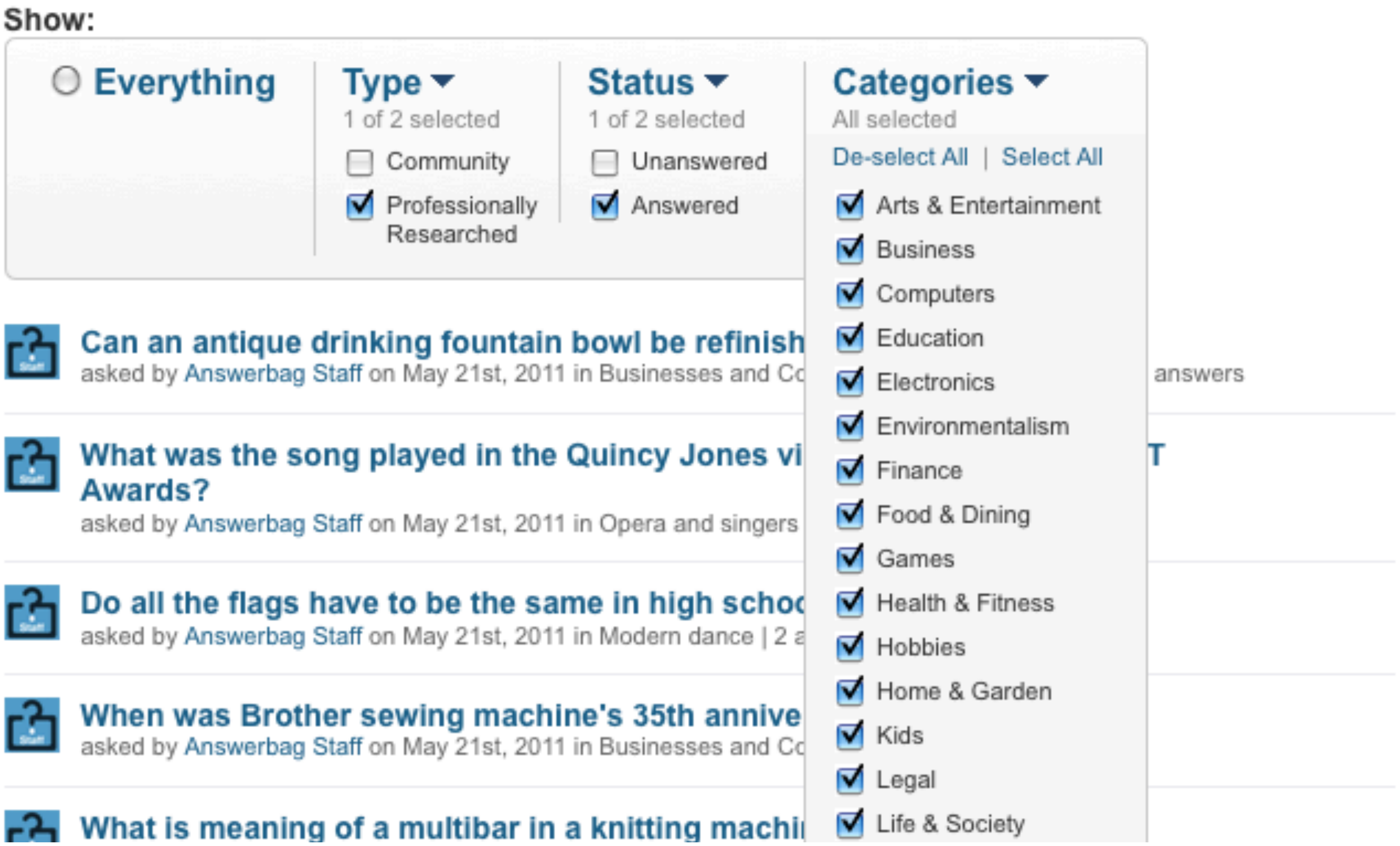
Drips answered the question:
What is your favourite purple fruit? 4 MINUTES AGO

dazed posted a new question:

Follow us on Facebook!

Answerbag
Like 18,162

Build high-quality QA corpus



Show:

○ Everything

Type ▾
1 of 2 selected
□ Community
☑ Professionally Researched

Status ▾
1 of 2 selected
□ Unanswered
☑ Answered

Categories ▾
All selected
De-select All | Select All
☑ Arts & Entertainment
☑ Business
☑ Computers
☑ Education
☑ Electronics
☑ Environmentalism
☑ Finance
☑ Food & Dining
☑ Games
☑ Health & Fitness
☑ Hobbies
☑ Home & Garden
☑ Kids
☑ Legal
☑ Life & Society

Can an antique drinking fountain bowl be refinish
asked by Answerbag Staff on May 21st, 2011 in Businesses and Co answers

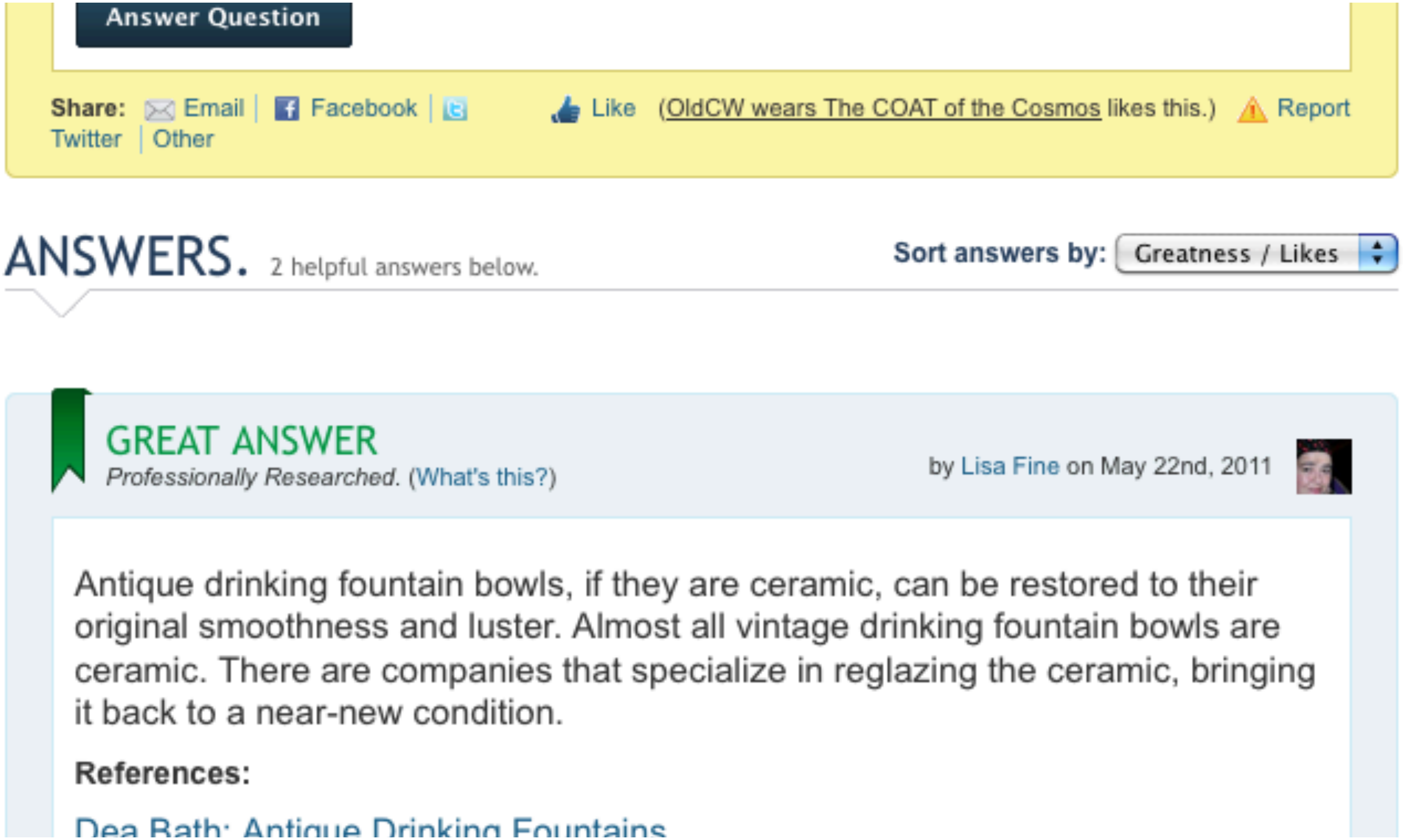
What was the song played in the Quincy Jones vi T
Awards?
asked by Answerbag Staff on May 21st, 2011 in Opera and singers

Do all the flags have to be the same in high schoo
asked by Answerbag Staff on May 21st, 2011 in Modern dance | 2 a

When was Brother sewing machine's 35th annive
asked by Answerbag Staff on May 21st, 2011 in Businesses and Co

What is meaning of a multibar in a knitting machi

Professionaly researched answers



Answer Question
Share: Email | Facebook | Twitter | Other
Like
Report
Sort answers by: Greatness / Likes

(OldCW wears The COAT of the Cosmos likes this.)

ANSWERS. 2 helpful answers below.

GREAT ANSWER

Professionally Researched. (What's this?)

by Lisa Fine on May 22nd, 2011

Antique drinking fountain bowls, if they are ceramic, can be restored to their original smoothness and luster. Almost all vintage drinking fountain bowls are ceramic. There are companies that specialize in reglazing the ceramic, bringing it back to a near-new condition.

References:

Dea Bath: Antique Drinking Fountains

Setting up QA example

Index:

```
$ sh run_qa_index.sh indexdir data/answers.50k.txt
```

Search:

```
$ sh run_qa_interactive.sh index 10
```

Exercise

- Try out constructing boolean queries
- Experiment with a more involved StemAnalyzer for indexing and searching
- Answer: How old is too old for tires?