Lab6 Report

—Mandy (Andrew id: jiayaoc)

Step I: Keyword Search

---

a. Purpose:

— Implement basic keyword search functionality over DBLP dataset: return the paper titles that contain a set of keywords for either paper titles or authors.
— For multiple word queries, consider the relationship as OR.

b. Implementation:

1) Create a Lucene index on attributes of database tables.

```
public static void SearchKeywords(ArrayList<String> keywords) throws Exception {
    List<Article> articles = DBLP.getDatabase();
    StandardAnalyzer standardAnalyzer = new StandardAnalyzer();
    Directory index = new RAMDirectory();
    IndexWriterConfig config = new IndexWriterConfig(standardAnalyzer);
    IndexWriter w = new IndexWriter(index, config);
```

2) Add documents to the index

```
    // Filter database to create all indexes for "title" and "author"
    for (int i = 0; i < articles.size(); i++) {
        Document doc = new Document();
        doc.add(new TextField("title", articles.get(i).title, Field.Store.YES));
        doc.add(new TextField("author", articles.get(i).author.toString(), Field.Store.YES));
        doc.add(new IntField("year", Integer.parseInt(articles.get(i).year), Field.Store.YES));
        w.addDocument(doc);
    }
```

3) Build queries after parsing by different requirements

```
    // *Situation1: If the keyword is for title*
    // Create a query
    Query qTitle = new QueryParser("title", standardAnalyzer).parse(keyword);


    // *Situation2: If the keyword is for author*
    // Create a query
    Query qAuthor = new QueryParser("author", standardAnalyzer).parse(keyword);
```

4) Create a searcher for each using the query, take ”author” as an example

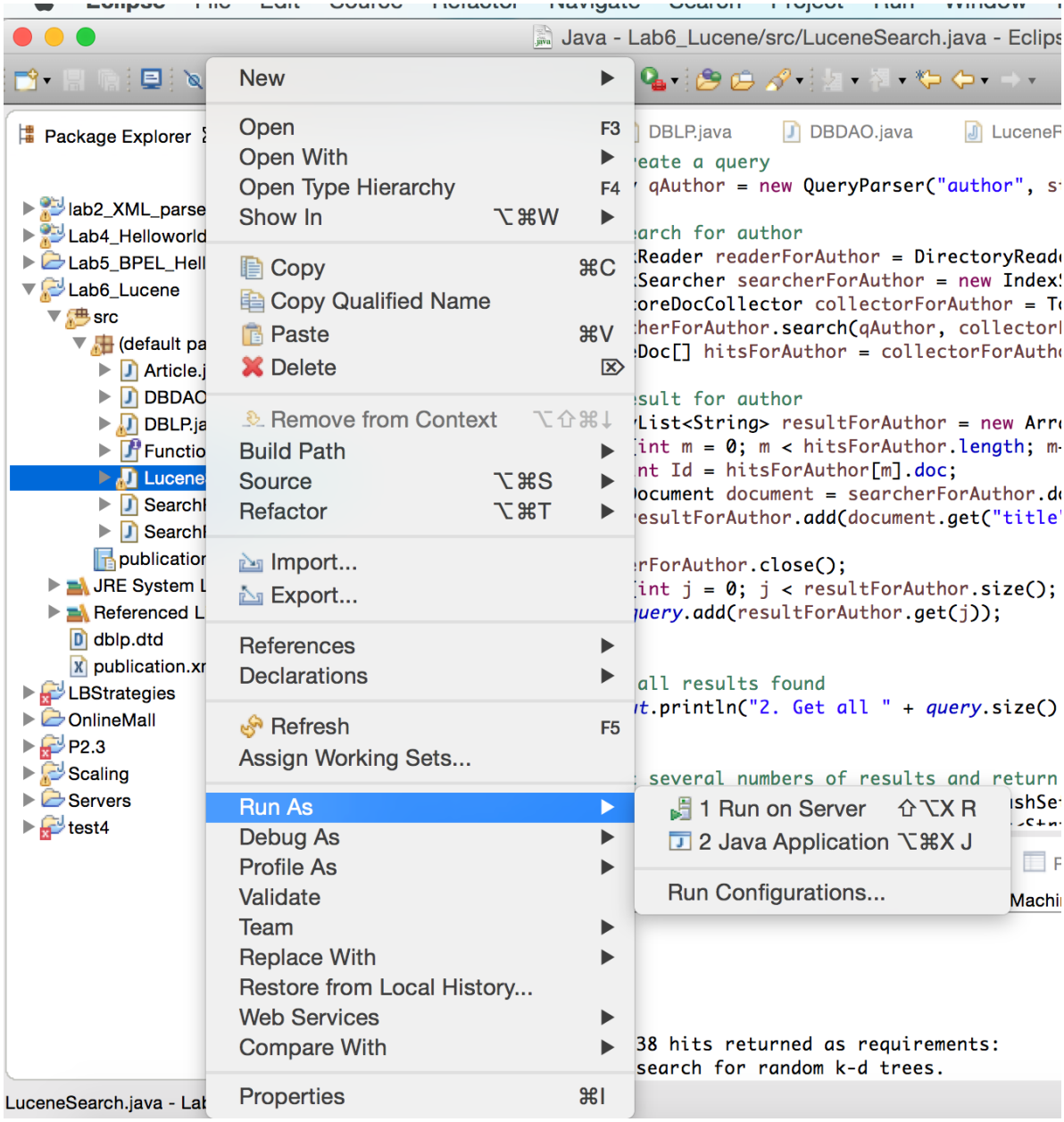
```
    // Search for author
    IndexReader readerForAuthor = DirectoryReader.open(index);
    IndexSearcher searcherForAuthor = new IndexSearcher(readerForAuthor);
    TopScoreDocCollector collectorForAuthor = TopScoreDocCollector.create(hitsPerPage);
    searcherForAuthor.search(qAuthor, collectorForAuthor);
    ScoreDoc[] hitsForAuthor = collectorForAuthor.topDocs().scoreDocs;
```

5) Display results

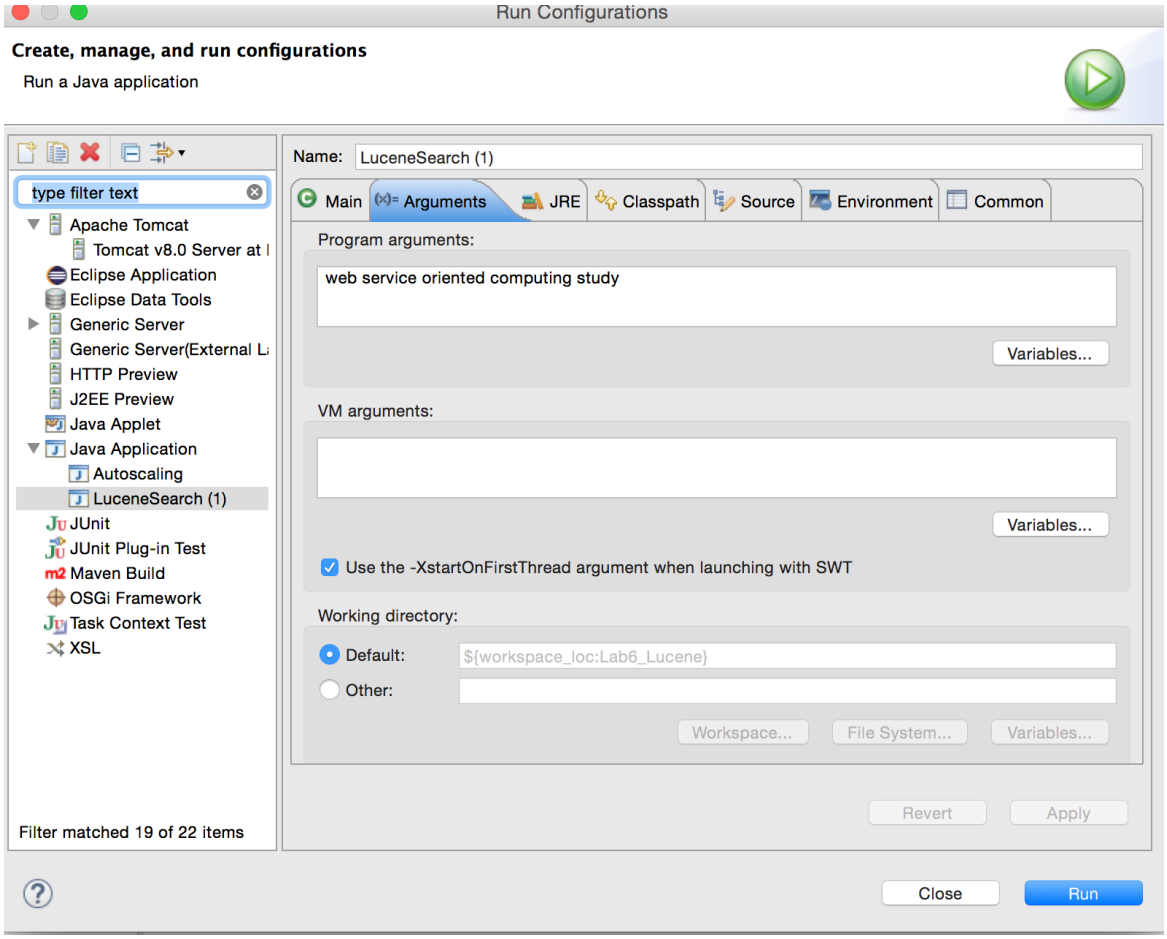
```
        // Result for author
        ArrayList<String> resultForAuthor = new ArrayList<String>();
        for (int m = 0; m < hitsForAuthor.length; m++) {
            int Id = hitsForAuthor[m].doc;
            Document document = searcherForAuthor.doc(Id);
            resultForAuthor.add(document.get("title"));
        }
        readerForAuthor.close();
        for (int j = 0; j < resultForAuthor.size(); j++) {
            query.add(resultForAuthor.get(j));
        }
    }
```

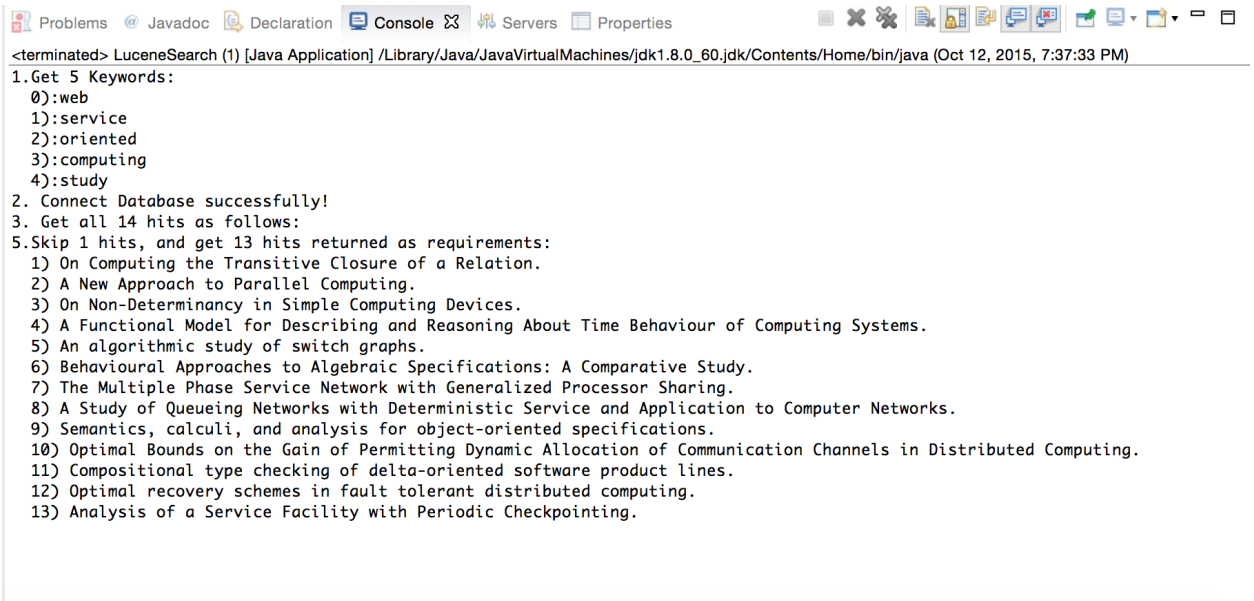
6) Screenshots for different situations & Way to run codes

- **right click on LuceneSearch.java, choose to run as Java Application at the first time when running, and choose to Run Configurations every time to input keywords:**

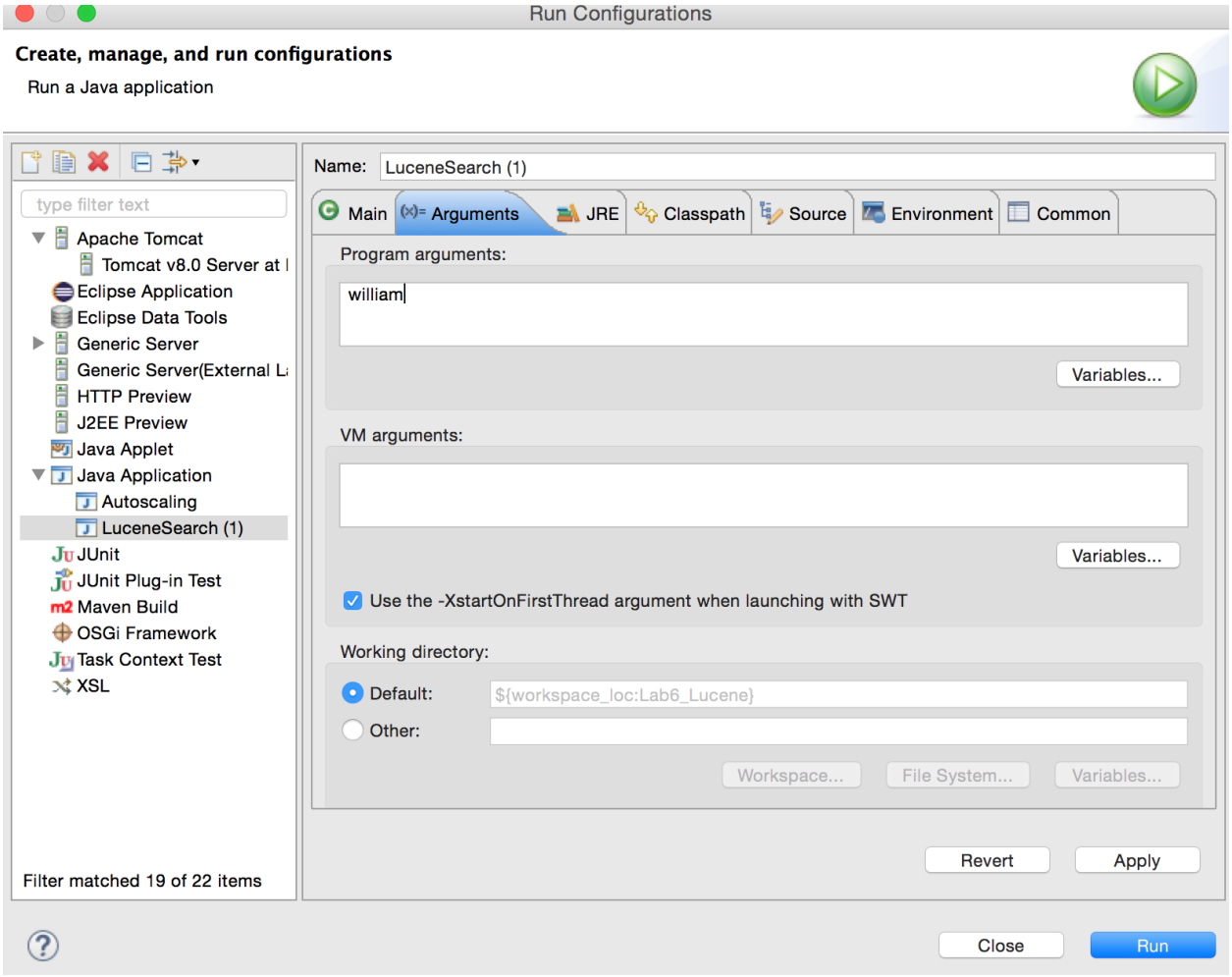


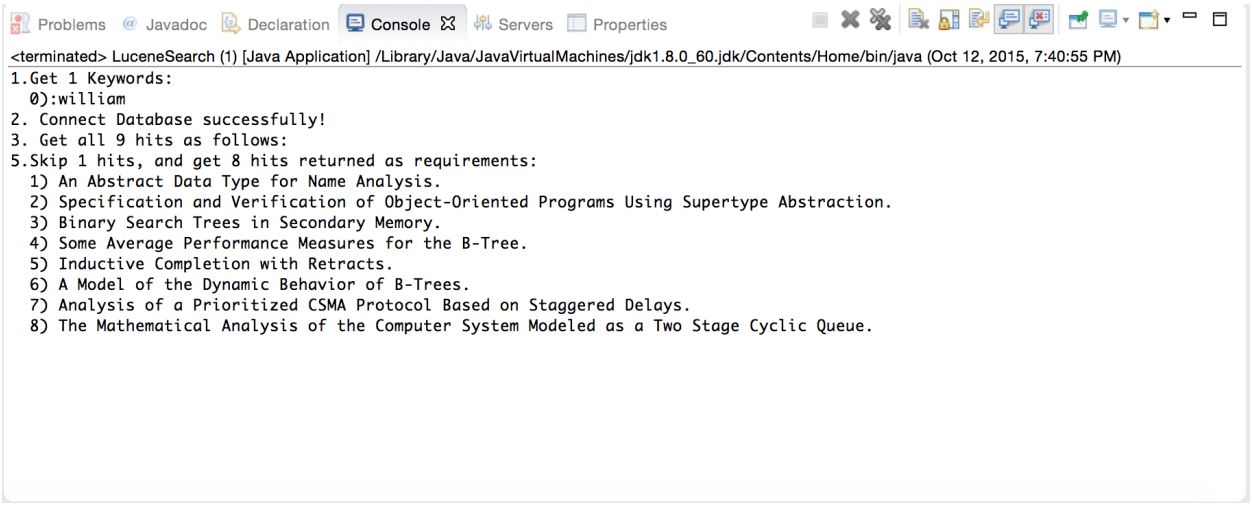
- **multiple title keywords: use Hashset to avoid duplicated results from “title” search and “author ” search**





- **single author keyword**





Step II: Spatial Search

---

a. Purpose:

— Search again by year ranges

b. Implementation:

1) using NumericRangeQuery API from Lucene to filter data according to years conveniently.

```
Query qYear = NumericRangeQuery.newIntRange("year", start, end, true, true);
```

2) Realize implementation as what done with “title” and “author”.
3) Screen shots for results, take the example ”william” used in the Step I BasicSearch as an example:

The result was 9 hits, and it turns to 3 hits after limiting years range from 1985-1990

```
<terminated> LuceneSearch (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Contents/Home/bin/java (Oct 12, 2015, 9:41:19 PM)
1.Get 1 Keywords:
  0):William
2. Get all 9 hits.
3.Skip 1 hits, and get 8 hits returned as requirements:
  1) An Abstract Data Type for Name Analysis.
  2) Specification and Verification of Object-Oriented Programs Using Supertype Abstraction.
  3) Binary Search Trees in Secondary Memory.
  4) Some Average Performance Measures for the B-Tree.
  5) Inductive Completion with Retracts.
  6) A Model of the Dynamic Behavior of B-Trees.
  7) Analysis of a Prioritized CSMA Protocol Based on Staggered Delays.
  8) The Mathematical Analysis of the Computer System Modeled as a Two Stage Cyclic Queue.
4. If range from year 1985 to 1990, get 3 hits as follows:
  0): Some Average Performance Measures for the B-Tree.
  1): Inductive Completion with Retracts.
  2): A Model of the Dynamic Behavior of B-Trees.
```