Building A Search Engine Using Lucene (Maven and Junit)

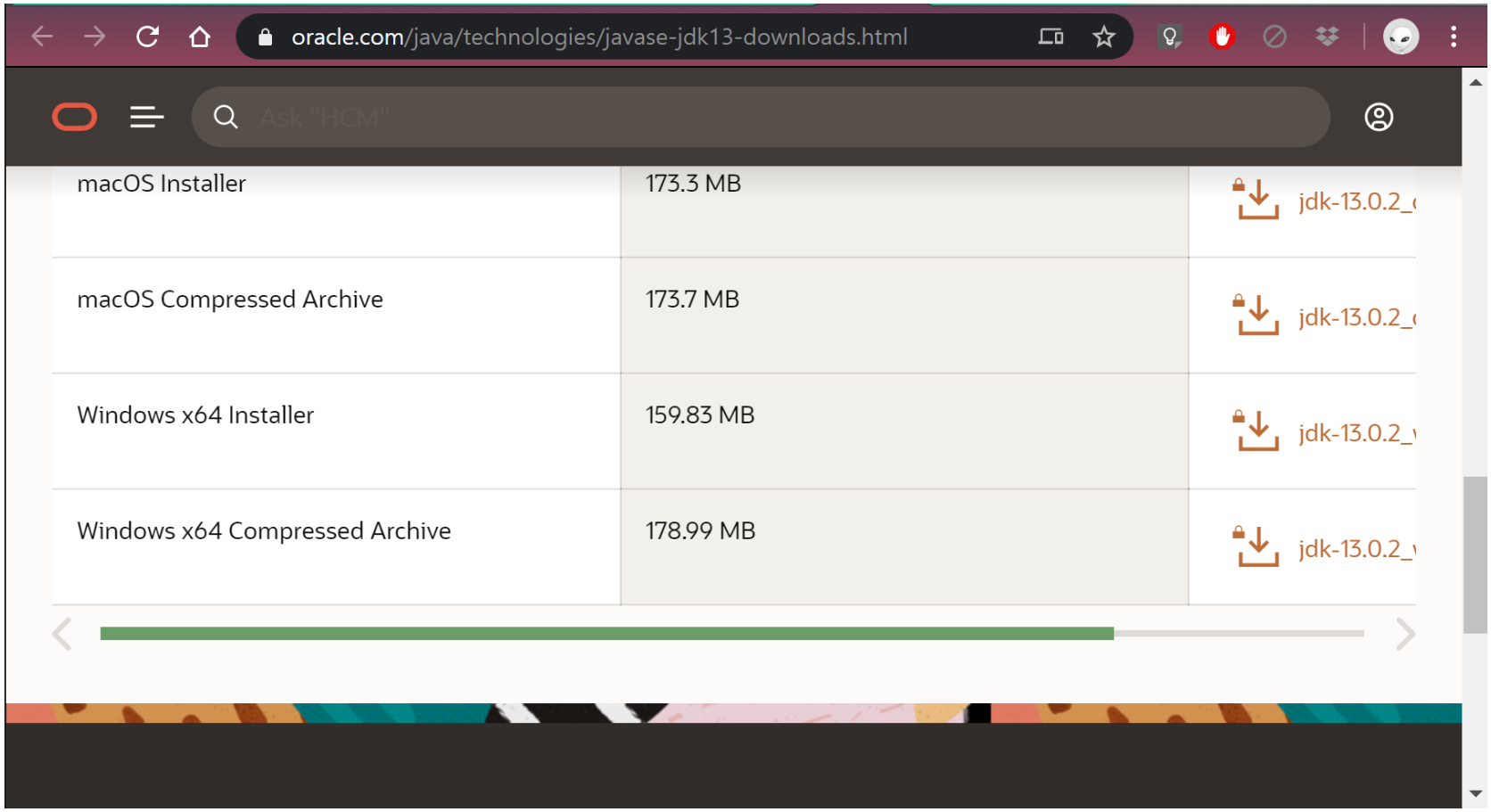
Younghoon Kim
nongaussian@hanyang.ac.kr

Tools

- Installing tools
  - JDK download & installation
  - Eclipse download & installation
- HelloLucene
  - Search engine using Lucene APIs
  - HelloLucene class implemented the SimpleSE interface
  - JUnit test

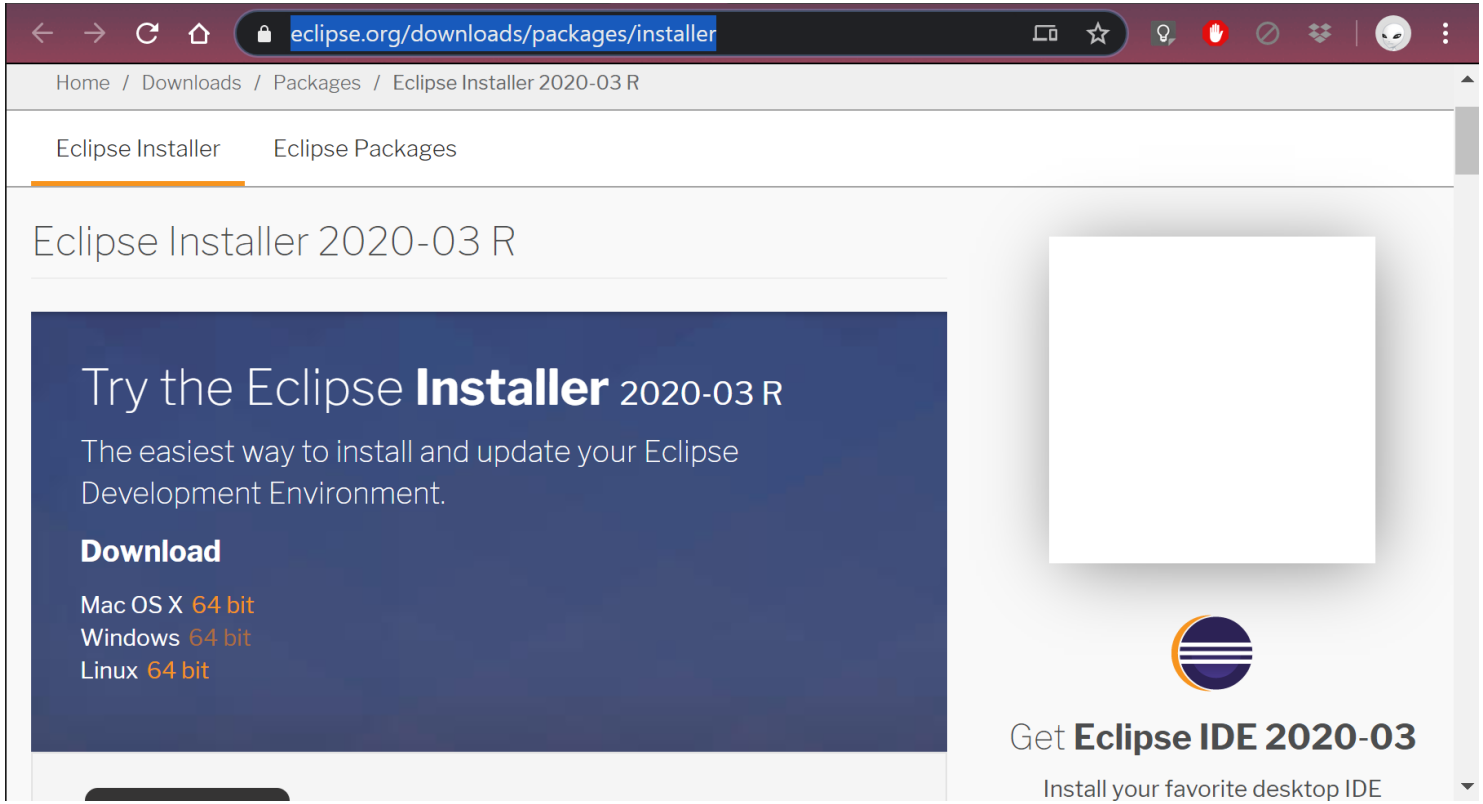
JDK

https://www.oracle.com/java/technologies/downloads/

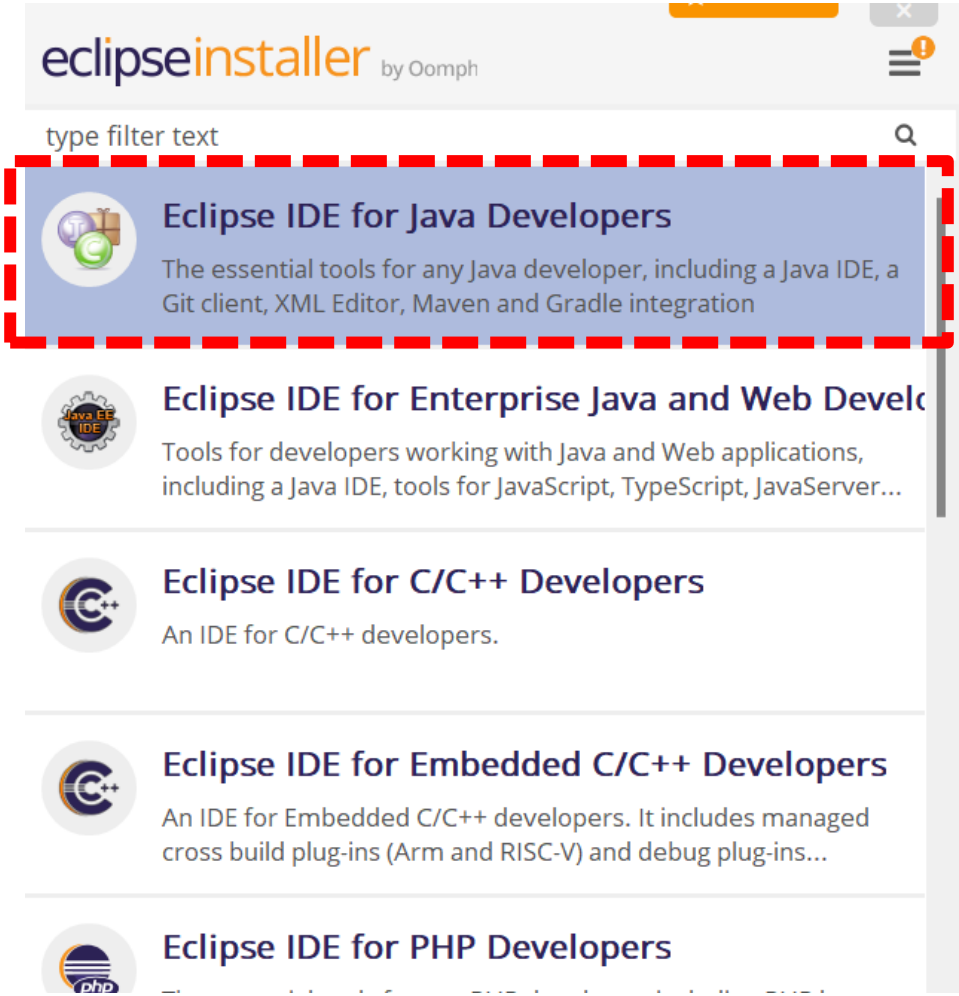


Eclipse

- https://www.eclipse.org/downloads/packages/installer



Eclipse


eclipseinstaller by Oomph
type filter text
Eclipse IDE for Java Developers
The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Maven and Gradle integration
Eclipse IDE for Enterprise Java and Web Develc
Tools for developers working with Java and Web applications, including a Java IDE, tools for JavaScript, TypeScript, JavaServer...
Eclipse IDE for C/C++ Developers
An IDE for C/C++ developers.
Eclipse IDE for Embedded C/C++ Developers
An IDE for Embedded C/C++ developers. It includes managed cross build plug-ins (Arm and RISC-V) and debug plug-ins...
Eclipse IDE for PHP Developers

Maven Plugin for Eclipse

https://www.eclipse.org/m2e/ → https://github.com/eclipse-m2e/m2e-core/blob/master/README.md#-installation

Installation

The recommended way to install Eclipse-m2e is using the Eclipse marketplace. Either click on

MARKETPLACE INSTALL ECLIPSE M2E or MARKETPLACE VIEW ECLIPSE M2E

into your Eclipse-IDE, or use the Eclipse Marketplace Client directly from within the IDE.

⚠️ *Some other entries exist that look like m2e. They're usually outdated or incorrect. Please use the c*

Alternatively, you can install the lastest M2Eclipse release by using the *Install New Software* dialog
repository:
https://download.eclipse.org/technology/m2e/releases/latest/

To use the latest snapshot build, you can use this p2 repository:
https://download.eclipse.org/technology/m2e/snapshots/latest/

BUILDING A SMALL SEARCH ENGINE USING LUCENE

Apache Lucene Project

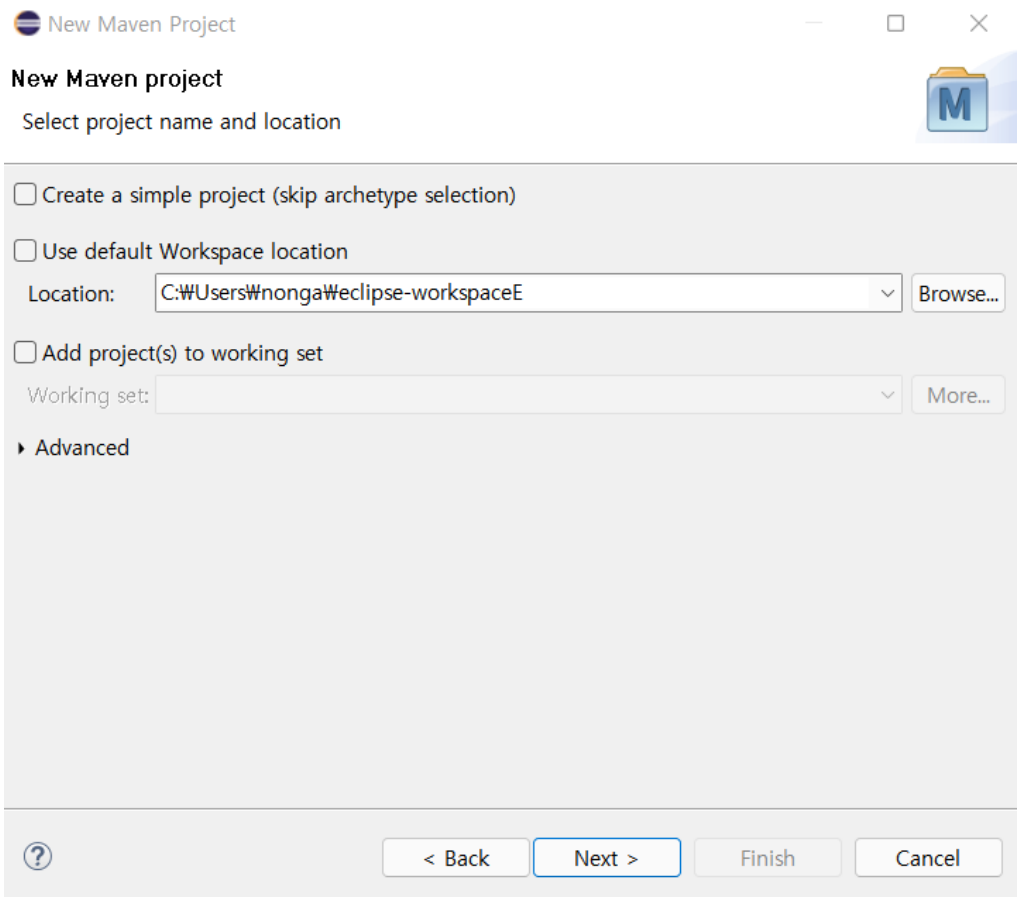
- Apache Lucene
  - A full-text search engine which can be used from various programming languages
  - A free and open-source search engine software library, originally written in Java
- Elasticsearch
  - A search engine based on the Lucene library

Apache Maven Project

- Maven is
  - A build automation tool used primarily for Java projects
- Maven deals with several areas of concern:
  - Making the build process easy
  - Providing a uniform build system
  - Providing quality project information
  - Encouraging better development practices

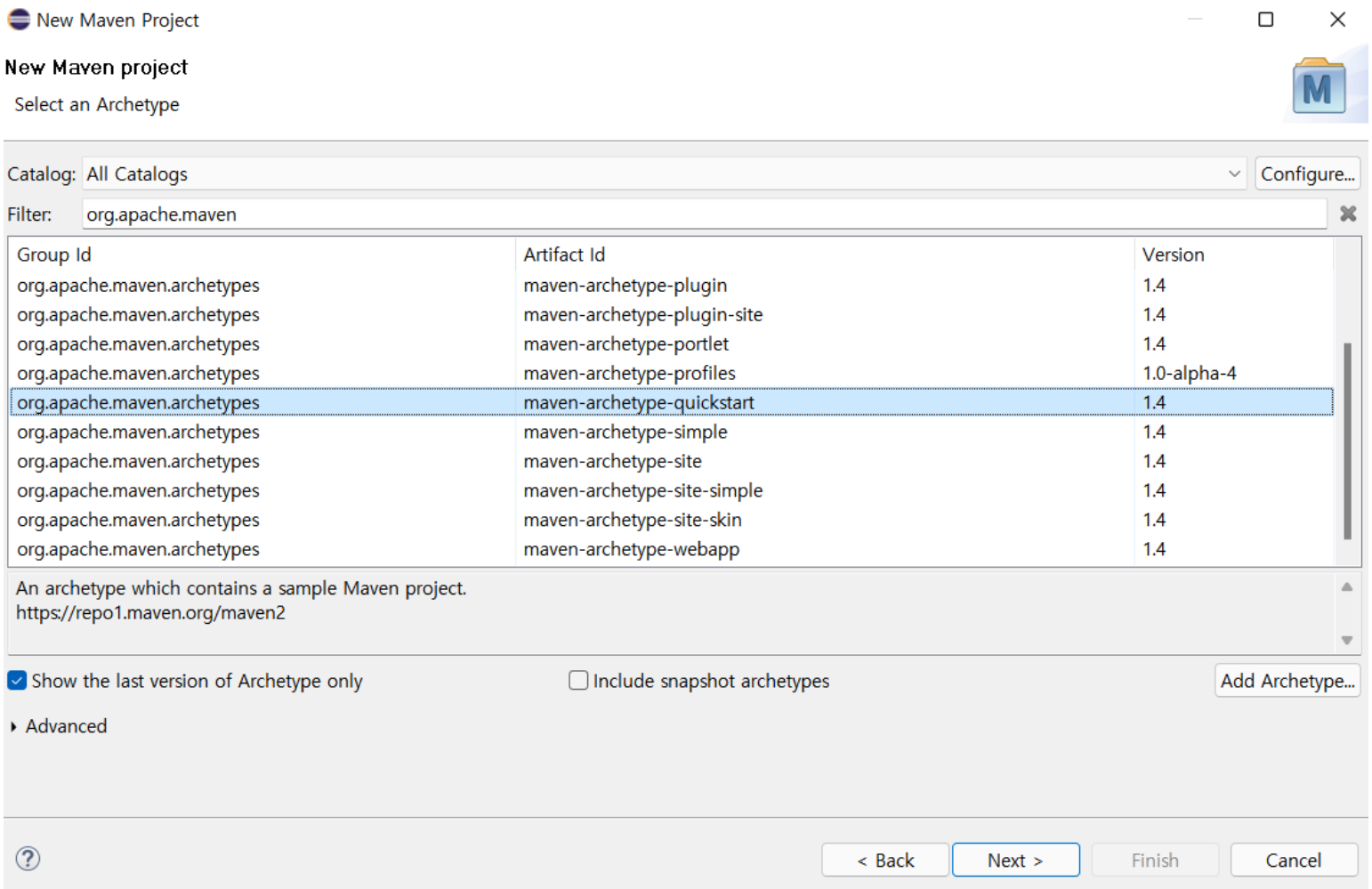
Create A New Maven Project in Eclipse

- File > New > Other
- Maven > Maven Project



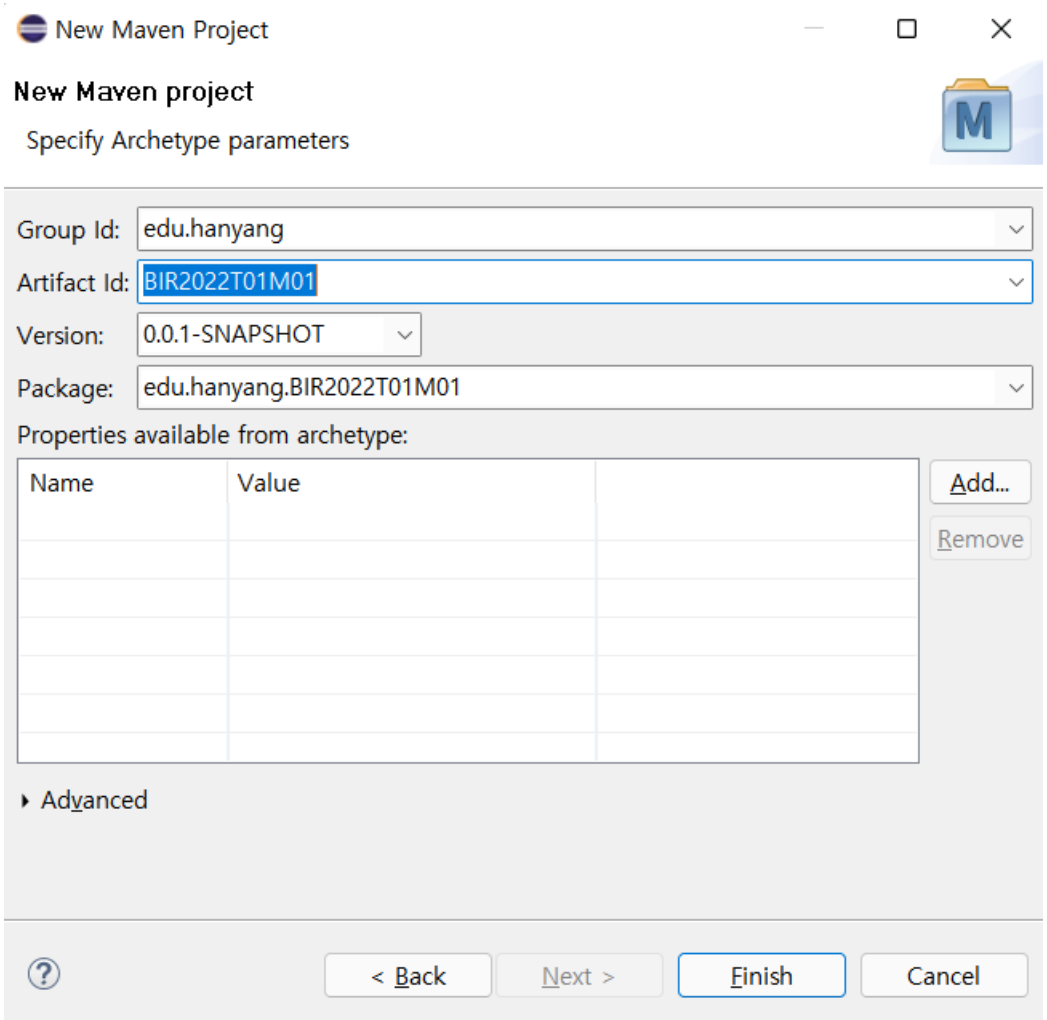
Create A New Maven Project

- Filter: org.apache.maven



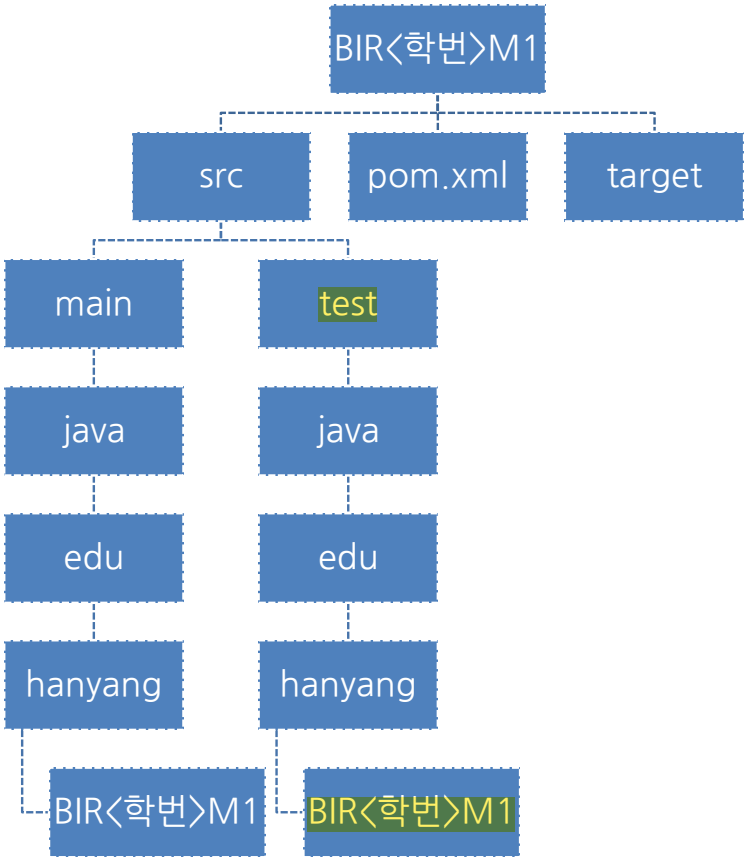
Create A New Maven Project

- Group ID:
  - edu.hanyang
- Artifact ID:
  - BIR<학번>M1
  - E.g., BIR202212345M1



Directory Structure (Mandatory)

- Unzip should output a directory named by the artifact ID (i.e., BIR<학번>M1)
- Directory structure:



Add Dependencies on Lucene in pom.xml

```
20      <dependencies>
21          <dependency>
22              <groupId>junit</groupId>
23              <artifactId>junit</artifactId>
24              <version>4.11</version>
25              <scope>test</scope>
26          </dependency>
27
28          <dependency>
29              <groupId>org.apache.lucene</groupId>
30              <artifactId>lucene-core</artifactId>
31              <version>7.1.0</version>
32          </dependency>
33
34          <dependency>
35              <groupId>org.apache.lucene</groupId>
36              <artifactId>lucene-queryparser</artifactId>
37              <version>7.1.0</version>
38          </dependency>
39
40      </dependencies>
```

Dependencies on Lucene

https://mvnrepository.com/artifact/org.apache.lucene/lucene-core/7.1.0

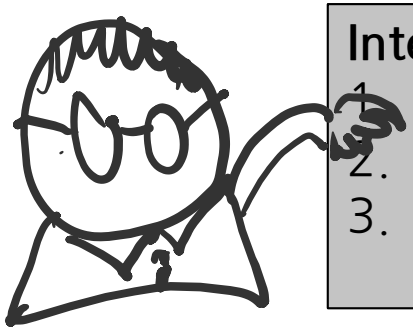
```
<dependency>
    <groupId>org.apache.lucene</groupId>
    <artifactId>lucene-core</artifactId>
    <version>7.1.0</version>
</dependency>

<dependency>
    <groupId>org.apache.lucene</groupId>
    <artifactId>lucene-queryparser</artifactId>
    <version>7.1.0</version>
</dependency>
```

References

- Sample code of using Lucene
  - http://www.lucenetutorial.com/lucene-in-5-minutes.html
  - https://www.baeldung.com/lucene

Modular Programming & Unit Testing

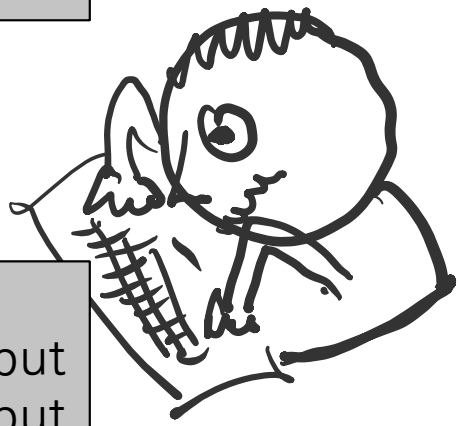
coordinator



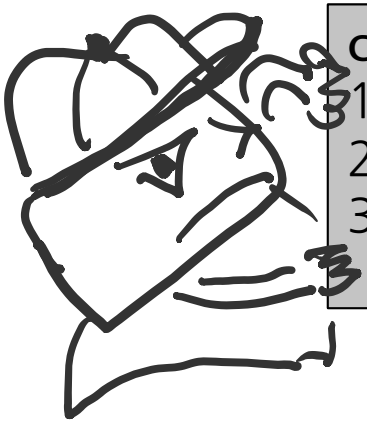
Interface A

1. Function1: input, output
2. Function2: input, output
3. …

Developer



Junit Tester

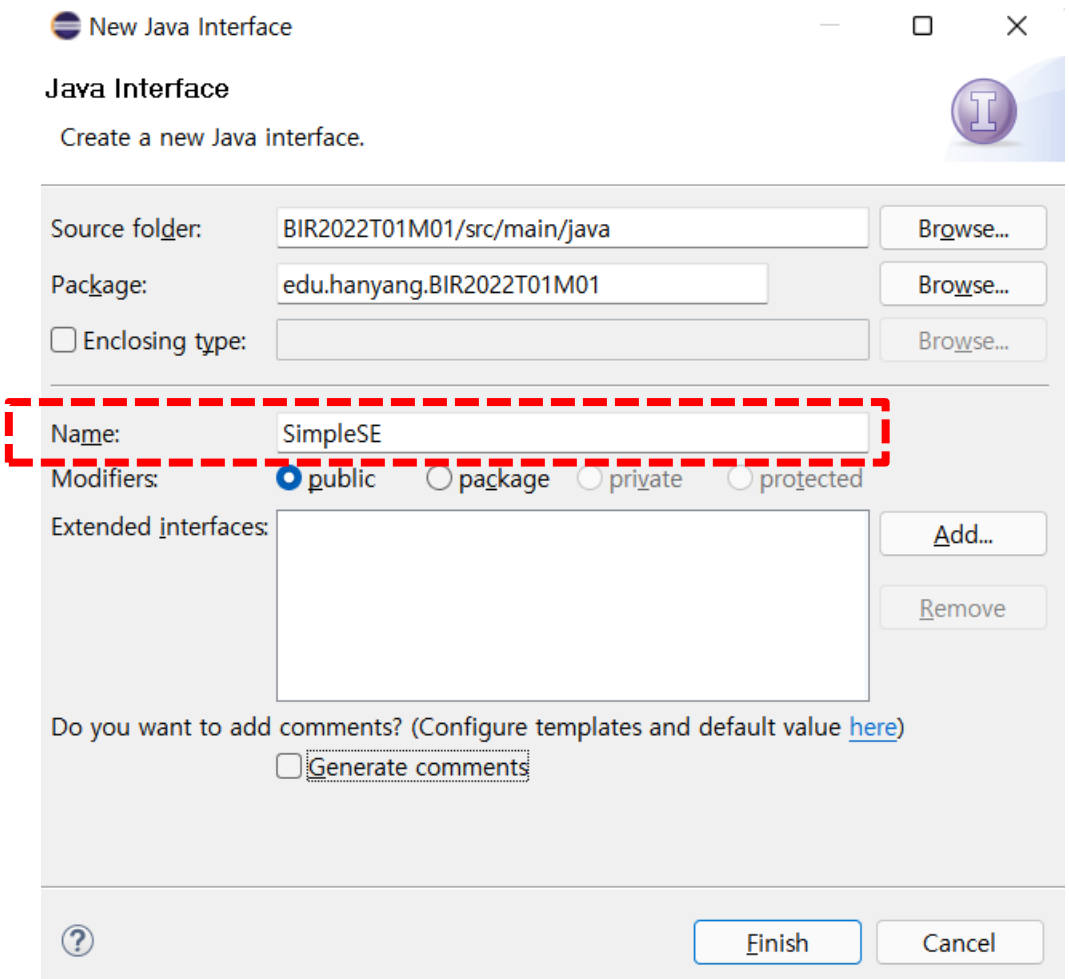


class B implemented A

1. Function1: input, output
2. Function2: input, output
3. …

Interface for Simple Search Engine

- File > New > Interface



Interface *SimpleSE*

Build index with *docs*
Keywords are generated using *analyzer*
Return a *Directory* handler

```
public interface SimpleSE {
    Directory createIndex(String[][] docs, Analyzer analyzer)
throws IOException;
```

Search for *querystr* from index
querystr is tokenized using *analyzer*
Returns an array of title and isbn as double array String

```
    String[][] search(Directory index, String querystr,
Analyzer analyzer) throws ParseException, IOException;
}
```

HELLOLUCENE

Sample Code of Index & Search Using An SimpleSE Instance

```
public static void main( String[] args ) throws IOException, ParseException {
    String docs[][] = {
      {"Lucene in Action", "193398817"},
      {"Lucene for Dummies", "55320055Z"},
      {"Managing Gigabytes", "55063552A"},
      {"The Art of Computer Science", "9900333X"}
    };

    HelloLucene se = new HelloLucene();

    // 1. create index & add docs
    Analyzer analyzer = new StandardAnalyzer();
    Directory index = se.createIndex(docs, analyzer);

    // 2. query
    String querystr = args.length > 0 ? args[0] : "lucene";

    // 3. search
    String[][] hits = se.search(index, querystr, analyzer);

    // 4. display results
    System.out.println("Found " + hits.length + " hits.");
    for(int i=0;i<hits.length;++i) {
        System.out.println((i + 1) + ". " + hits[i][0] + "\t" + hits[i][1]);
    }
}
```

Keyword tokenizer

The same tokenizer

Recall the Indexing Process

documents

Tokenization & Normalization

DocID=1:

build
your
own
search
engine
...

DocID=2:
...

collect the triples of terms and their positions

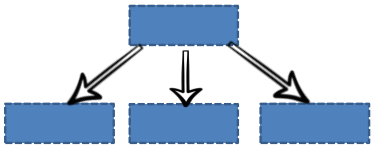
<build, 1, 1>
<your, 1, 2>
<own, 1, 3>
<search, 1, 4>
<engine, 1, 5>
...
<want, 1023, 7>
<to, 1023, 8>
<search, 1023, 9>
<for, 1023, 10>
...

External sort by term, DocId, position

<a, 2, 4>, <a, 3, 1>, ···,
<build, 1, 1>, <build, 49, 2>, <build, 49, 10>, ···
<search, 1, 4>, <search, 1023, 9>, ···

Build dictionary & posting lists

B+ tree



...

<a,0>, ···, <build, 40301>, ···

<2, 4>, <3, 1>, ···,
<1, 1>, <49, 2>, <49, 10>, ···
<1, 4>, <1023, 9>, ···

Interface: createIndex

Keyword tokenizer

```
 private static Directory createIndex(String[][] docs, Analyzer
analyzer) throws IOException {
    // 1. Index
    Directory index = new RAMDirectory();

    IndexWriterConfig config = new IndexWriterConfig(analyzer);
    IndexWriter w = new IndexWriter(index, config);

    for (String[] doc: docs) {
        addDoc(w, doc[0], doc[1]);
    }

    w.close();

    return index;
}
```

Handler for Index data structure

Building inverted index

addDoc

- Define a function ‘addDoc’

```
public void addDoc(IndexWriter w, String title, String isbn) throws
IOException {
    Document doc = new Document();

    doc.add(new TextField("title", title, Field.Store.YES));
    doc.add(new StringField("isbn", isbn, Field.Store.YES));

    w.addDocument(doc);
}
```

public TextField(String name, String value, Field.Store store)

Creates a new TextField with String value.

Parameters:

- name - *field name*
- value - string value
- store - Store.YES if the content should also be stored

Recall Query Processing

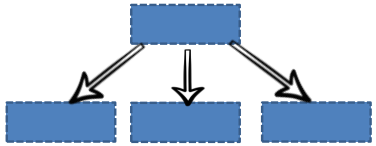
Retrieve posting lists

Query:
build AND engine

build: <1, 1>, <49, 2>, <49, 10>, <83, 149>, ···

engine: <1, 5>, <51, 2>, <58, 55>, <83, 4>, ···

B+ tree



···

<a, 0>, ···, <build, 40301>, ···

<2, 4>, <3, 1>, ···,
<1, 1>, <49, 2>, <49, 10>, ···
<1, 4>, <1023, 9>, ···

Query processing

Result: 1, 83, ···

Module 4

Rank by relevance

1, 83, ···

Interface: search

```
private static String[][] search(Directory index, String querystr, Analyzer
analyzer) throws ParseException, IOException {
    Query q = new QueryParser("title", analyzer).parse(querystr);

    int hitsPerPage = 10;
    IndexReader reader = DirectoryReader.open(index);
    IndexSearcher searcher = new IndexSearcher(reader);

    TopScoreDocCollector collector = TopScoreDocCollector.create(hitsPerPage);
    searcher.search(q, collector);
    ScoreDoc[] hits = collector.topDocs().scoreDocs;

    String[][] result = new String[hits.length][2];
    for(int i=0; i<hits.length; i++) {
        int docId = hits[i].doc;
        Document d = searcher.doc(docId);
        result[i][0] = d.get("title");
        result[i][1] = d.get("isbn");
    }

    reader.close();

    return result;
}
```

Parsing query with a tokenizer

Fetching results

Output Result

```
public static void main( String[] args ) throws IOException, ParseException {
    …

    // 4. display results
    System.out.println("Found " + hits.length + " hits.");
    for(int i=0;i<hits.length;++i) {
        System.out.println((i + 1) + ". " + hits[i][0] + "\t" + hits[i][1]);
    }
}
```

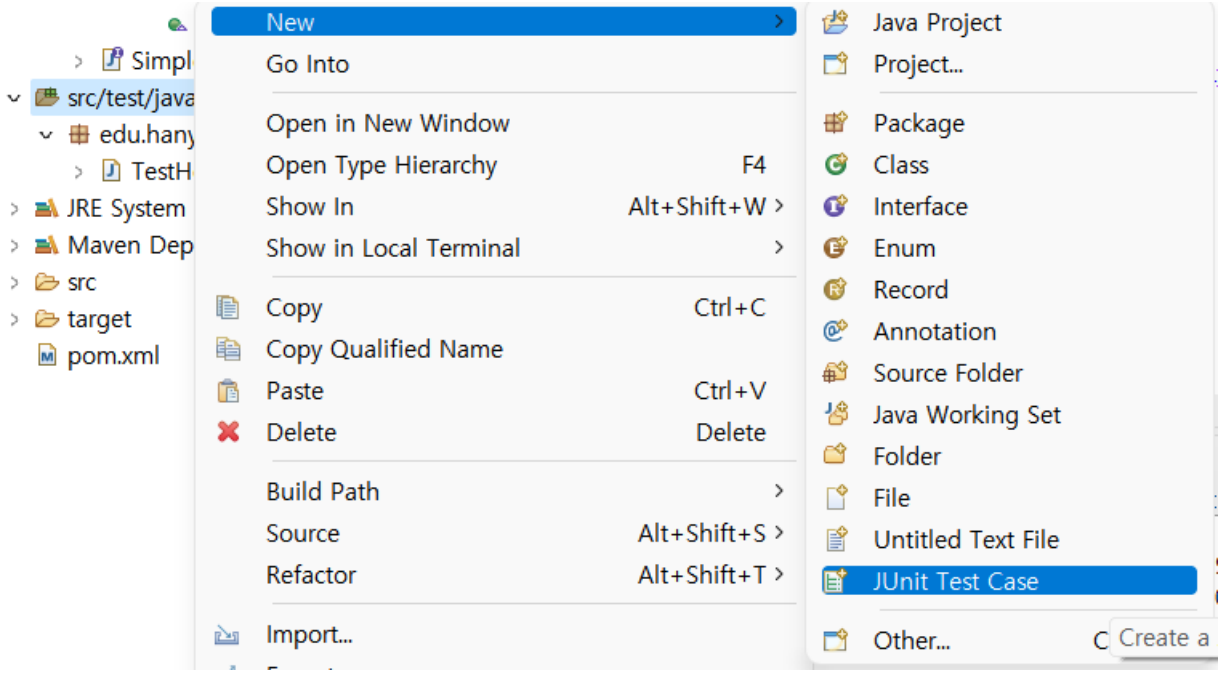
```
Found 2 hits.
1. Lucene in Action 193398817
2. Lucene for Dummies 55320055Z
```

```
package edu.hanyang.BIR202212345M1;

import …


public class HelloLucene implements SimpleSE
{

    public static void main( String[] args ) throws IOException, ParseException
    {
                    …
    }

    public Directory createIndex(String[][] docs, Analyzer analyzer) throws IOException {
       Directory index = new RAMDirectory();

                    …
      return index;
    }

    public String[][] search(Directory index, String querystr, Analyzer analyzer) throws ParseException, IOException {
      Query q = new QueryParser("title", analyzer).parse(querystr);
                        …


      return result;
    }

    public void addDoc(IndexWriter w, String title, String isbn) throws IOException {
                                        …
    }
}
```

JUNIT TEST

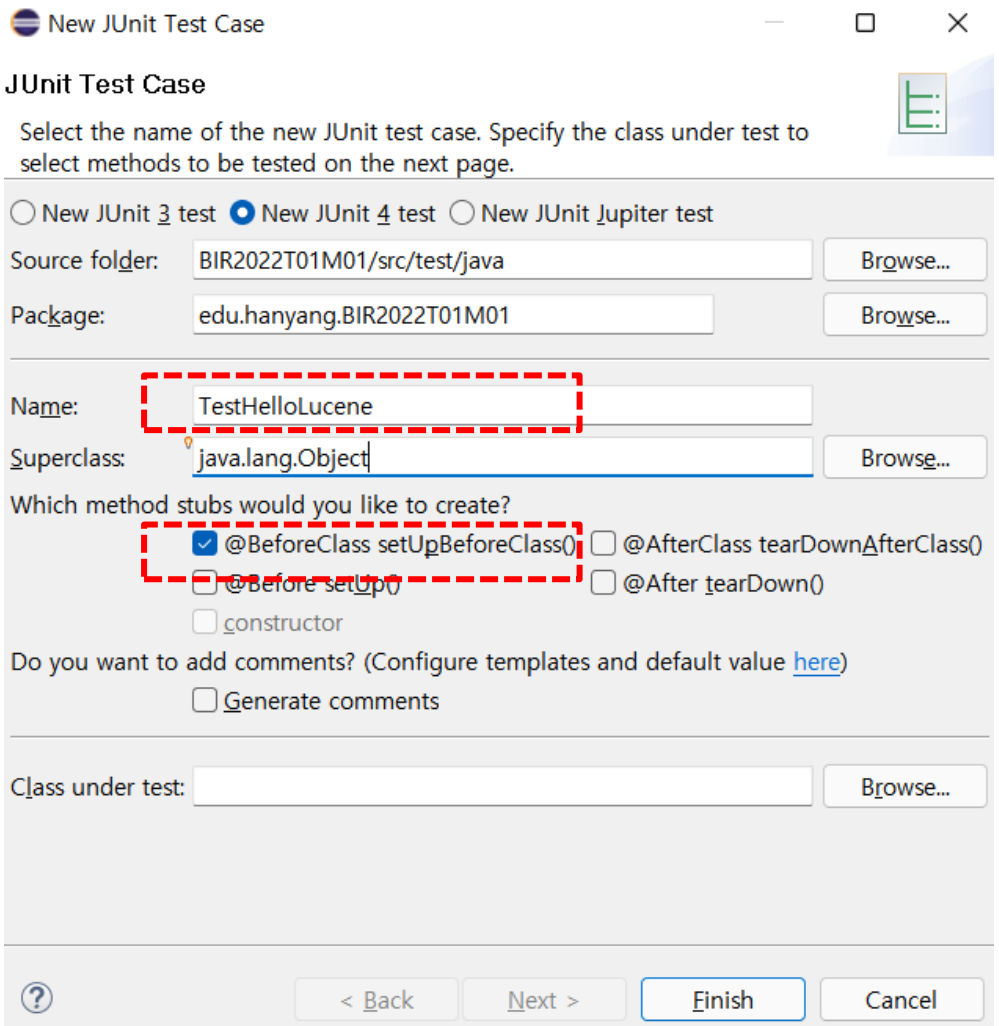
Add Junit Test Class

- Right click on ‘src/test/java’ > New > JUnit Test Case



Add Junit Test Class

- Right click on ‘src/test/java’ > New > JUnit Test Case



TestHelloLucene.java

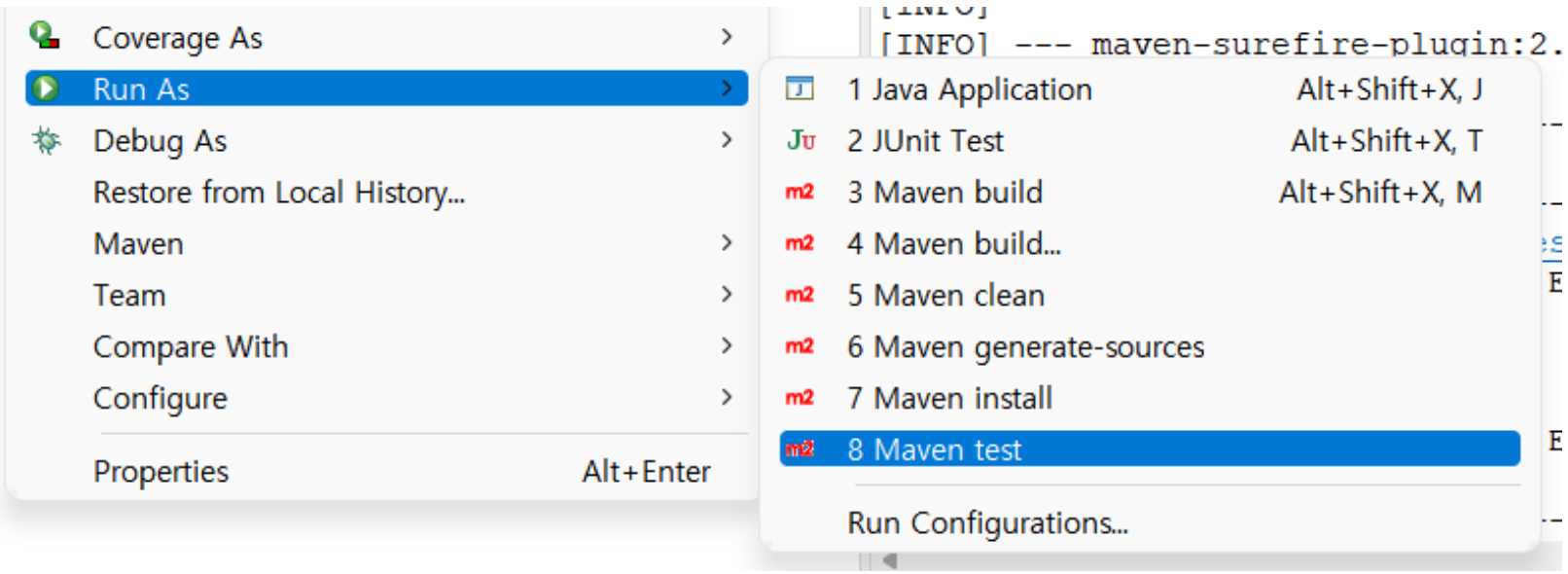
```
private static HelloLucene se = null;
private static Directory index = null;
private static Analyzer analyzer = null;


@BeforeClass
public static void setUpBeforeClass() throws Exception {
    String docs[][] = {
    {"Lucene in Action", "193398817"},
    {"Lucene for Dummies", "55320055Z"},
    {"Managing Gigabytes", "55063552A"},
    {"The Art of Computer Science", "9900333X"}
    };

    se = new HelloLucene();

    // 1. create index & add docs
    analyzer = new StandardAnalyzer();
    index = se.createIndex(docs, analyzer);
}
```

TestHelloLucene.java

```
@Test
public void test1() throws ParseException, IOException {
  String[][] hits = se.search(index, "lucene", analyzer);
  assertEquals(hits.length, 2);
  assertEquals(hits[0][1], "193398817");
  assertEquals(hits[1][1], "55320055Z");
}

@Test
public void test2() throws ParseException, IOException {
  String[][] hits = se.search(index, "action", analyzer);
  assertEquals(hits.length, 1);
  assertEquals(hits[0][1], "193398817");
}

@Test
public void test3() throws ParseException, IOException {
  String[][] hits = se.search(index, "computer", analyzer);
  assertEquals(hits.length, 1);
  assertEquals(hits[0][1], "9900333X");
}
```

Run Maven Test

- Right click the project root node > Run as > Maven Test



Test Result

```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running edu.hanyang.test.TestHelloLucene
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.227 s - in edu
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.867 s
[INFO] Finished at: 2022-02-25T00:46:26+09:00
[INFO] ------------------------------------------------------------------------
```

Homework

- Goal
  - Learn how to build a search engine using Lucene's APIs
  - Use Maven to test and manage packages
  - Understand what JUnit does
- Problem
  - Implement *SimpleSE* interface
  - The class name **MUST BE** HelloHolmes
  - Index
    - A Study In Scarlet, by Arthur Conan Doyle
    - A document = (a line number, string in the line)
    - Uploaded on our LMS (게시판에 업로드되어 있는 244-8.txt파일 사용)
  - Returns the line number where a given query string appears in the line
  - Your submission must pass the unit test with TestHelloHolmes

Homework

- Constraints
  - Return of the ‘search’ function is a double array of String type
    - String[][]
      - String[][0] = a line number in the string type (e.g., 0, 1, 2, 3, ⋯)
      - String[][1] = null
  - Example
    - Returns String[][] = {
      {“0”, null },
      {“1”, null }
      }

How to read a file line by line

```
private static String[][] read_docs () throws IOException {
    ClassLoader classLoader = TestHelloHolmes.class.getClassLoader();
    File path = new File(classLoader.getResource("244-8.txt").getFile());

    List<String> linelist = Files.readAllLines(path.toPath(),
    StandardCharsets.ISO_8859_1);
    String[][] lines = new String[linelist.size()][2];

    int idx = 0;
    for (String line: linelist) {
        lines[idx][0] = Integer.toString(idx);
        lines[idx][1] = line;
        idx++;
    }

    return lines;
}
```

JUnitTest

- Download from LMS

```
package edu.hanyang.BIR202212345M1;

import static org.junit.Assert.*;

import java.io.File;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.util.List;

import org.apache.lucene.analysis.Analyzer;
import org.apache.lucene.analysis.standard.StandardAnalyzer;
import org.apache.lucene.queryparser.classic.ParseException;
import org.apache.lucene.store.Directory;
import org.junit.BeforeClass;
import org.junit.Test;

public class TestHelloHolmes {

		private static HelloHolmes se = null;
		private static Directory index = null;
```

```
private static Analyzer analyzer = null;

private static String[][] read_docs () throws IOException {
ClassLoader classLoader = TestHelloHolmes.class.getClassLoader();
File path = new File(classLoader.getResource("244-8.txt").getFile());

        List<String> linelist = Files.readAllLines(path.toPath(), StandardCharsets.ISO_8859_1);
        String[][] lines = new String[linelist.size()][2];

        int idx = 0;
        for (String line: linelist) {
                lines[idx][0] = Integer.toString(idx);
                lines[idx][1] = line;
                idx++;
        }

        return lines;
}

@BeforeClass
public static void setUpBeforeClass() throws Exception {
        String docs[][] = read_docs();

        se = new HelloHolmes();

        // 1. create index & add docs
        analyzer = new StandardAnalyzer();
        index = se.createIndex(docs, analyzer);
}

@Test
public void test1() throws ParseException, IOException {
```

```
@BeforeClass
public static void setUpBeforeClass() throws Exception {
		String docs[][] = read_docs();

		se = new HelloHolmes();

		// 1. create index & add docs
		analyzer = new StandardAnalyzer();
		index = se.createIndex(docs, analyzer);
}

@Test
public void test1() throws ParseException, IOException {
		String[][] hits = se.search(index, "holmes", analyzer);

		assertEquals(hits[0][0], "2226");
		assertEquals(hits[1][0], "1708");
}

@Test
public void test2() throws ParseException, IOException {
		String[][] hits = se.search(index, "watson", analyzer);

		assertEquals(hits[0][0], "4090");
		assertEquals(hits[1][0], "138");
}

@Test
public void test3() throws ParseException, IOException {
		String[][] hits = se.search(index, "murder", analyzer);
```

```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running edu.hanyang.BIR202212345M1.TestHelloHolmes
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.449 s -
[INFO] Running edu.hanyang.BIR202212345M1.TestHelloLucene
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.001 s -
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.199 s
[INFO] Finished at: 2022-03-16T16:11:20+09:00
[INFO] ------------------------------------------------------------------------
```

Submission

- 1) Zip the directory
  (for example, BIR202212345M1.zip)
- 2) Upload the zip file on LMS