浙江大學
ZHEJIANG UNIVERSITY

Undergraduate Report

Course	Application of Java Programming
Academic Advisor	Professor Weiming Lu
Teaching Assistants	
Name	Leming Shen
ID Number	3180103654
College	College of Computer Science and Technology
Major	Software Engineering

December 5, 2020

Contents

1 Introduction to Java Web Crawler & Index

1.1 Background

In the course Java Application Technology, we are asked to write a program that can crawl data from http://dangdang.com. We need to get the title, author's name, classification, publication, picture, brief description of the content, brief introduction to the author, the catalog, and the price of a single book.

After obtaining all the information, we need to store them in a local file and create a searching index on the document. Then we can search for the book in our local document using the index we have just created.

1.2 Goal of Design

1. Crawl some books' information from http://dangdang.com, the information includes:
    - The book's title
    - The author's name
    - The classification of the book
    - The publication
    - The picture of the book (in URL format)
    - A brief description of the book
    - A brief introduction to the author
    - The catalog of the book
    - The price of the book
2. Store the data into a local document
3. Create searching index on the local document
4. Search for the corresponding book's information according to your inputted keyword and attribute(s) name

1.3 Development & Running Configuration Environment

1.3.1 Development Environment

1. Java version "1.8.0_251"
2. Java(TM) SE Runtime Environment (build 1.8.0_251-b08)
3. Java HotSpot(TM) Client VM (build 25.251-b08, mixed mode, sharing)
4. JDK 12.0
5. IDE: Jetbrains IntelliJ IDEA 2020.2.3
6. Chrome Driver
7. Third-party Jar
    - Jsoup
    - IKAnalyzer
    - Lucene
    - Selenium

1.3.2 Running Configuration Environment

1. Before running the program, make sure that you have downloaded the correct version of chromedriver corresponding to your Chrome browser
2. Make sure that you have added the chromedriver to the environment variables
3. Open the whole folder with IntelliJ IDEA

2 Overall Design

2.1 Project Structure

|-homework4

|----.idea

|----database

|----index (contains the created index)

|----output.txt

|----lib

|----Lucene (contains Lucene jar)

|----selenium (contains selenium jar)

|----IKAnalyzer2012_FF.jar

|----jsoup-1.13.1.jar

|----out

|----src

|----Book.java

|----Dangdang_Crawler.java

|----Index.java

|----test.java (contains the main() function)

|----debug.log

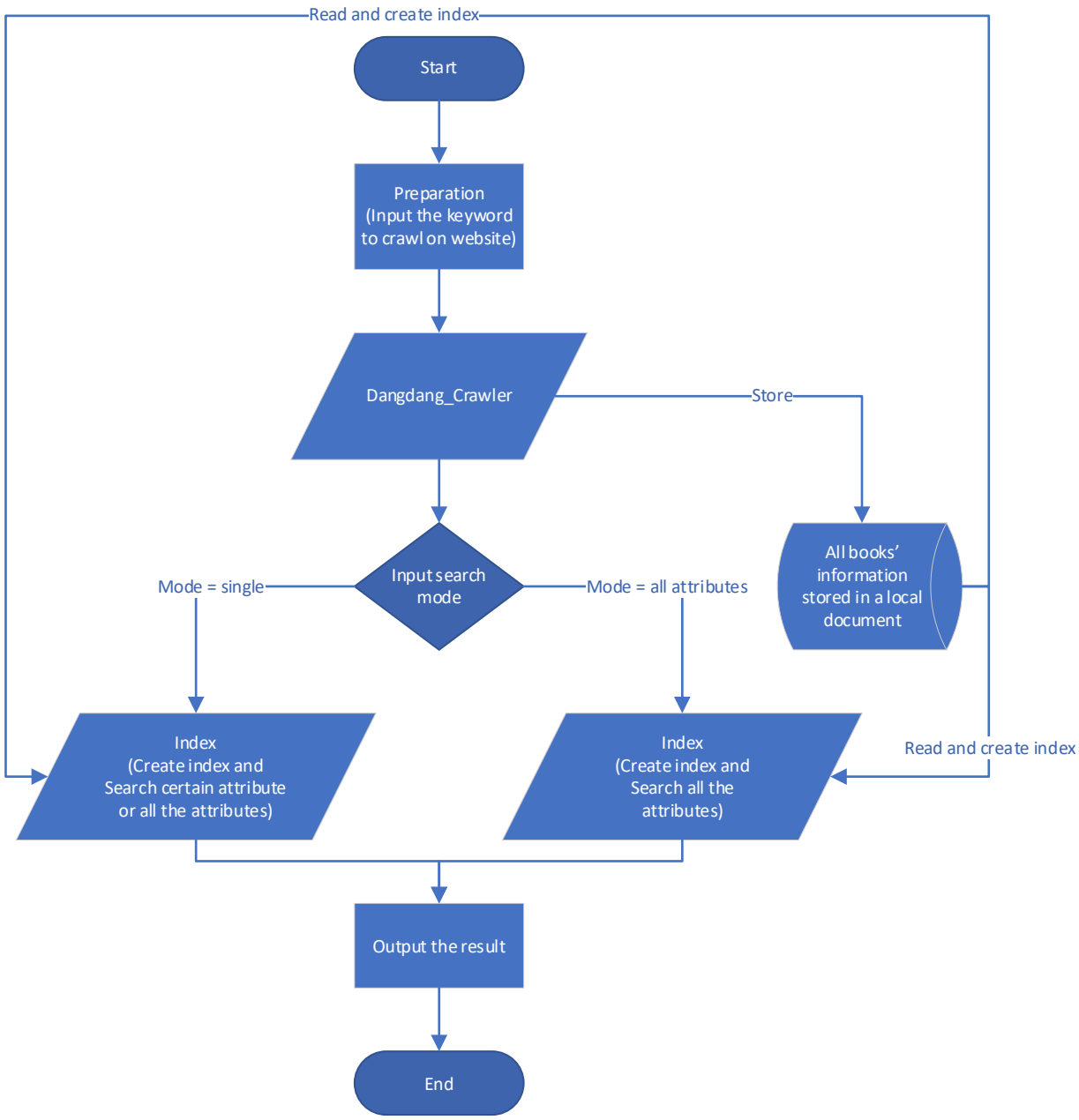
|----homework4.imi

2.2 Class Design

1. Book
   This class encapsulate a single book into a class, which contains a single book's information.
2. Dangdang_Crawler
   This class can crawl data from http://dangdang.com and write the obtained information into a local document located at /database/ named "output.txt".
3. Index

This class can read the information from the local document created by Dangdang_Crawler and create a search index on it. With whatever keyword you input and whatever attributes you want to search for, it can quickly get the result you need.

2.3 Flowchart



3 Detailed Design

3.1 Book

Private attributes (A single book's property):

title, author, classification, publication, picture_link, price, content_description, author_description, catalog

Public Method (Return the book's property as an ArrayList):

public ArrayList<String> get_all_information()

3.2 Dangdang_Crawler

3.2.1 Public ArrayList<Book> Craw()

This method firstly opens *"http://search.dangdang.com/?key="* + ***keyword*** + *"&action=input"* and find the total number of pages of the result.

Then for each page, to prevent the program from being detected by Dangdang, I use a chromedriver to help me. We can use Java code to directly operate the chromedriver and get the website page's source code from it. The URL of each page is formatted as: *"http://search.dangdang.com/?key="* + ***this.keyword*** + *"&action=input&page_index="* + ***Integer.toString(page_index).***

After getting the page source, Jsoup is used to analyze the html code and get all the information from the page.

As soon as we get all properties of a single book, the data is immediately encapsulated in a class named Book and added to the result ArrayList. Because of that, even if our program is detected by Dangdang and is interrupted, the data we already crawled are still in memory.

3.2.2 Public void Write()

This method writes every single book in ArrayList to a local document, formatted as:

Title	XXX
Author	XXX
Classification	XXX
Publication	XXX
Picture_link	XXX
Price	XXX
Content_description	XXX
Author_description	XXX
Catalog	XXX

3.3 Index

3.3.1 Public void create_index()

This method creates index on the array of documents returned from get_document().

3.3.2 Public ArrayList<Document> get_document()

This method firstly read the data from the local file created by Dangdang_Crawler.Write(), then for each book, it creates a document containing 9 TextField corresponding to a single book's 9 properties.

The documents are all added to the result ArrayList and returned.

3.3.3 Public ArrayList<Book> search()

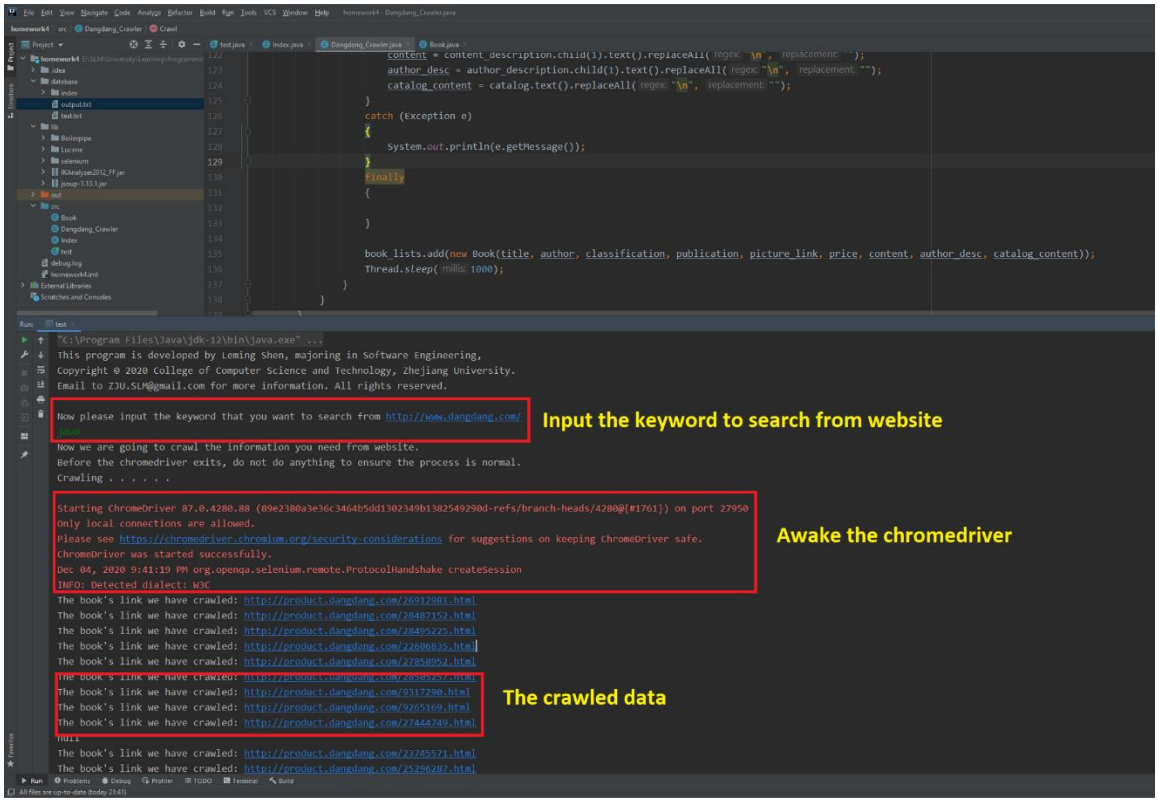
The search() method is a stereotype that search the keyword from the document with the help of index we have just created.

One thing need mentioning is that this method has two searching modes: search by single attribute or search by all attributes. I use a class named MultiFieldQueryParser to implement multi field searching.

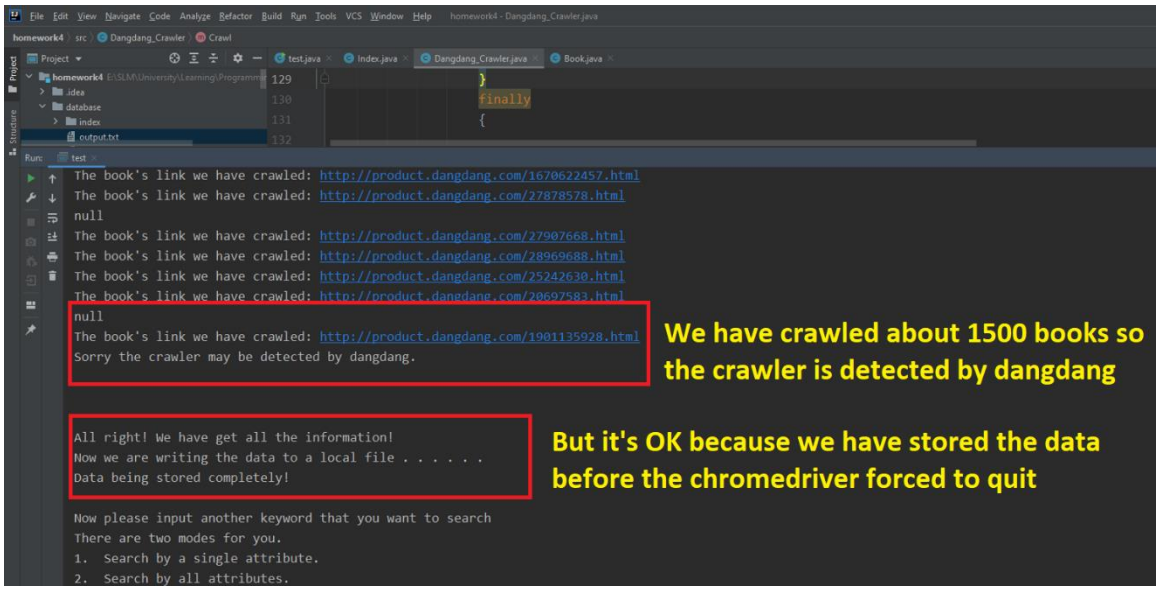
After getting the result, we choice top 10 to output.

4 Testing & Running

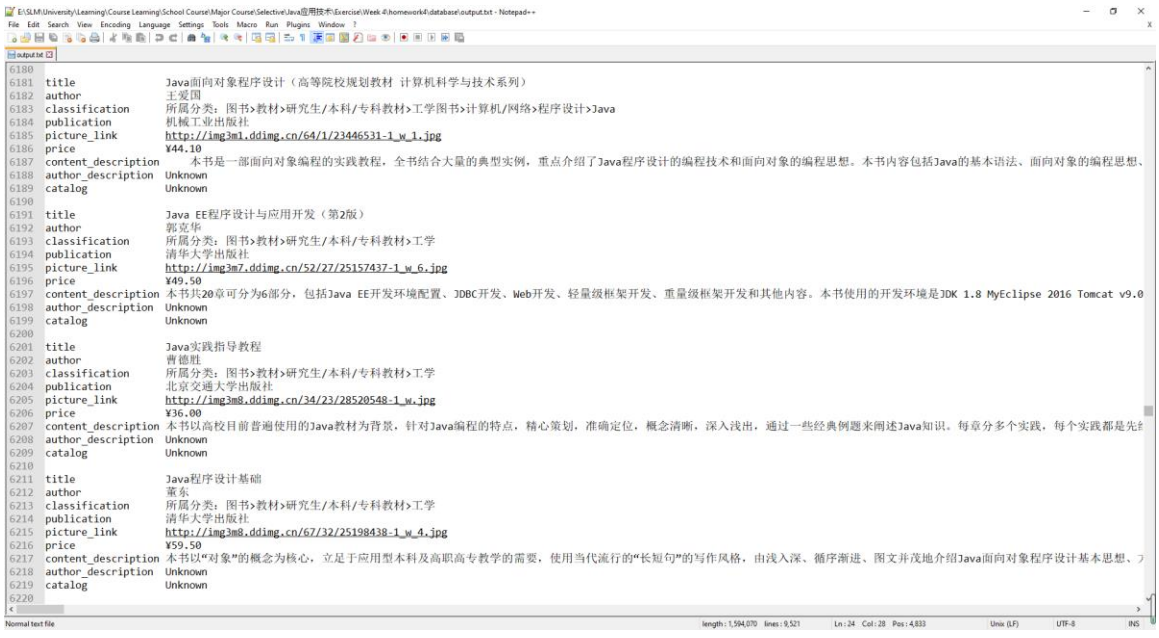
4.1 Input the keyword and search it from website meanwhile crawling the information of the book



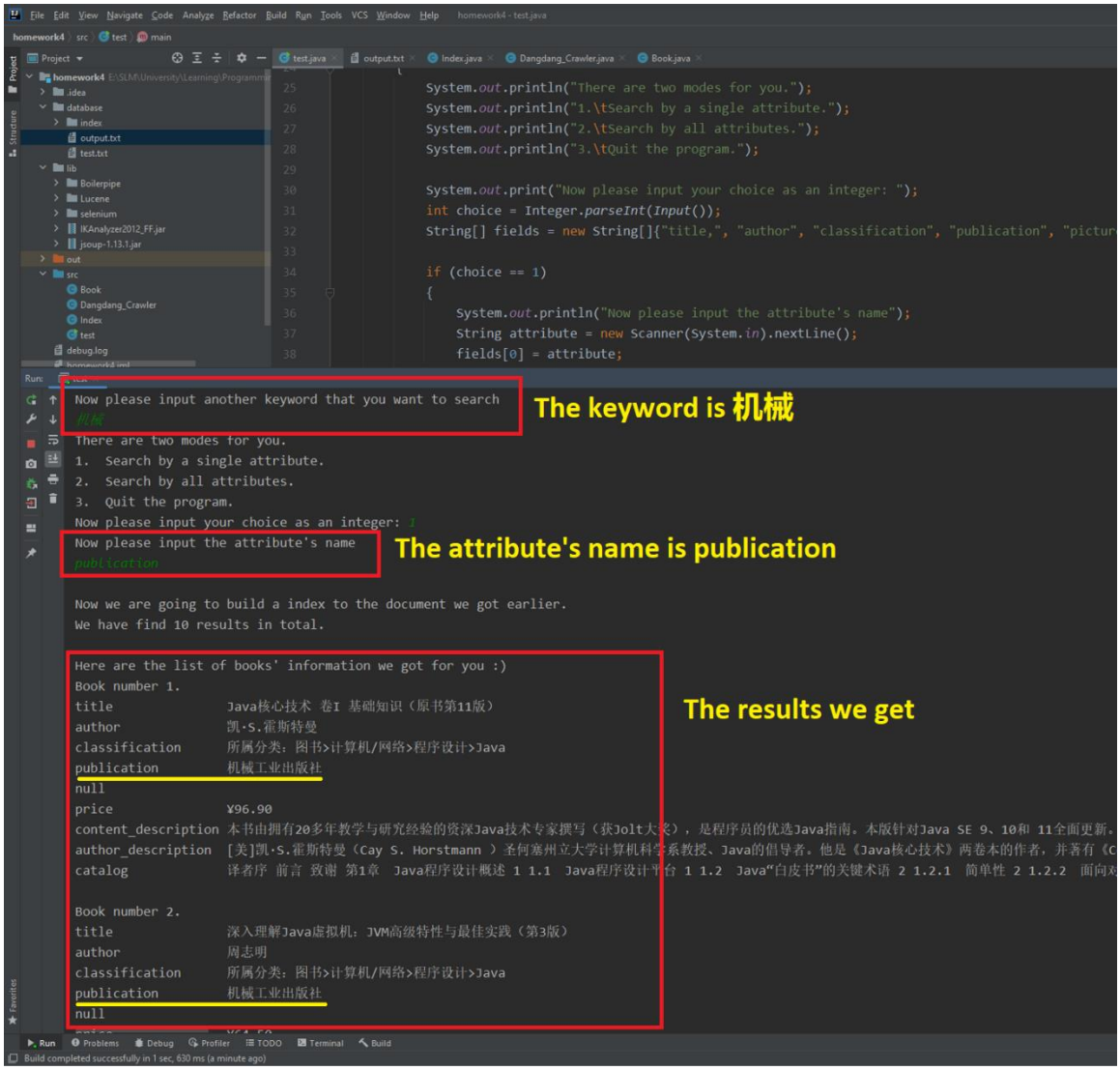
4.2 The crawler is detected by Dangdang because we have crawled so much data



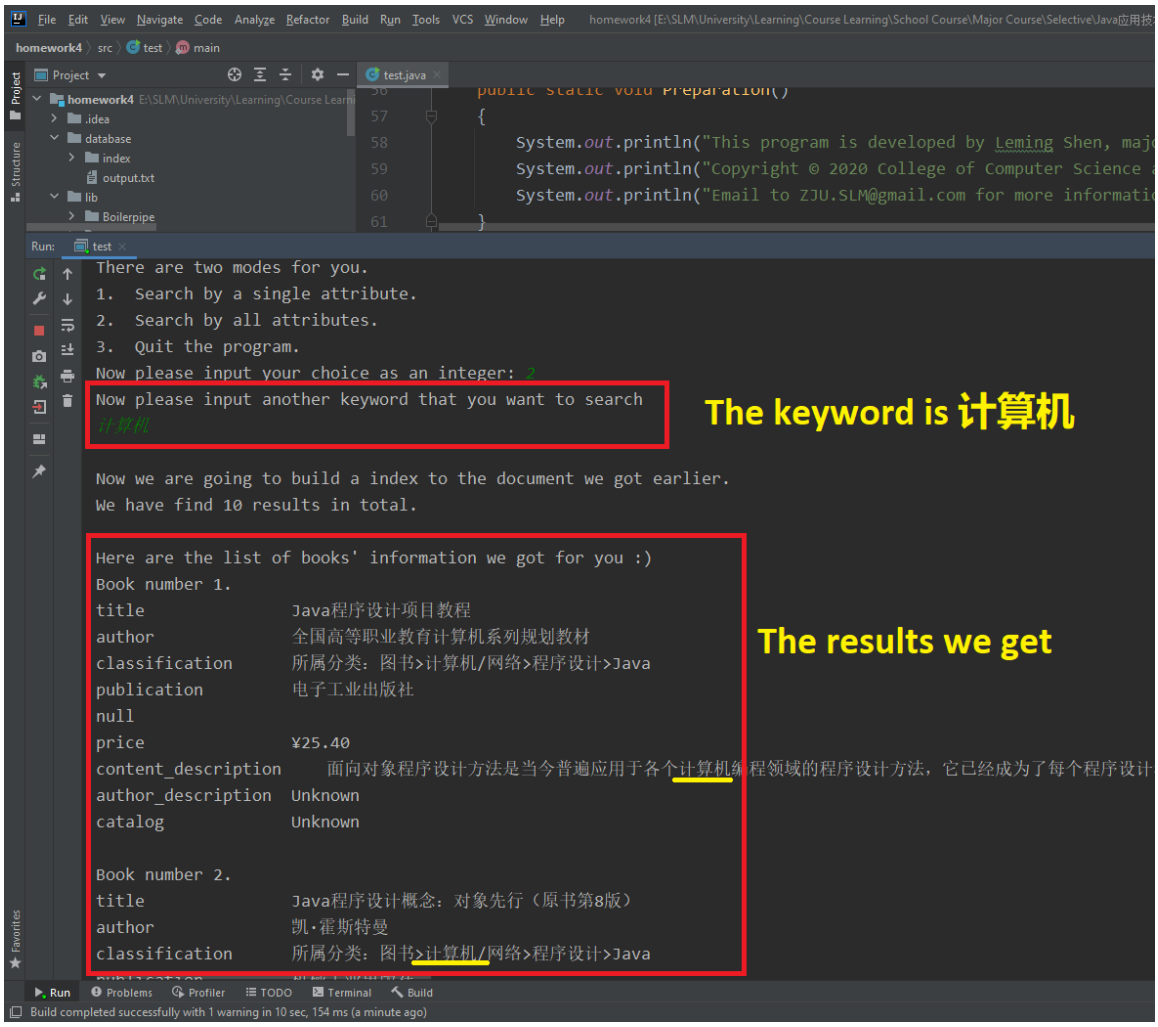
4.3 The crawled data stored in output.txt



4.4 Search the keyword by a single attribute



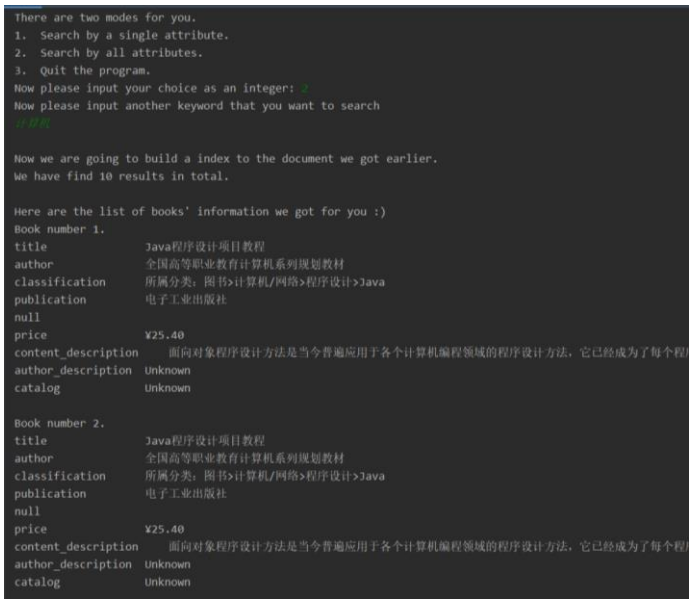
4.5 Search the keyword by all attributes



5 Summary

The project is a kind of easy and the crawler is easy to operate.

The only problem I get is that every time the program is runed, a new index will be added to the directory instead of replacing the original indexes created previously. Thus, the second time we run the program, we have got two duplicate documents, causing two duplicate result. →



6 References

1. *Introduction to JAVA Programming*, comprehensive version, 10th edition, Y. Daniel Liang.
2. *JAVA Concepts Early Objects*, 7th edition, Cay Horstmann, San Jose State University.
3. *Java Software Solutions*, Foundations of Program Design, Global Edition, John Lewis, William Loftus.
4. *Starting out with JAVA Early Objects*, 5th edition, Tony Daddis.

7 Source Code

1. Book.java

```
1.  import java.util.ArrayList;
2.
3.  public class Book
4.  {
5.      private final String title;
6.      private final String author;
7.      private final String classification;
8.      private final String publication;
9.      private final String picture_link;
10.     private final String price;
11.     private final String content_description;
12.     private final String author_description;
13.     private final String catalog;
14.
15.     public Book(
16.             String title,
17.             String author,
18.             String classification,
19.             String publication,
20.             String picture_link,
21.             String price,
22.             String content_description,
23.             String author_description,
24.             String catalog
25.     )
26.     {
27.         this.title = title;
28.         this.author = author;
29.         this.classification = classification;
30.         this.publication = publication;
31.         this.picture_link = picture_link;
32.         this.price = price;
33.         this.content_description = content_description;
34.         this.author_description = author_description;
35.         this.catalog = catalog;
36.     }
37.
38.     public ArrayList<String> get_all_information()
39.     {
40.         ArrayList<String> result = new ArrayList<String>();
41.
42.         result.add(this.title);
43.         result.add(this.author);
44.         result.add(this.classification);
45.         result.add(this.publication);
46.         result.add(this.picture_link);
```

```
        result.add(this.price);
        result.add(this.content_description);
        result.add(this.author_description);
        result.add(this.catalog);

        return result;
    }
}
```

2. Dangdang_Crawler.java

```
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.util.HashMap;
import java.util.Map;
import java.util.ArrayList;
import java.util.concurrent.TimeUnit;

public class Dangdang_Crawler
{
    private Document document;
    private String keyword = "";

    public Dangdang_Crawler(String keyword) throws IOException
    {
        String target = "http://search.dangdang.com/?key=" + keyword + "&action=inp
ut";
        this.document = Jsoup.connect(target).get();
        this.keyword = keyword;
    }

    public void Write(ArrayList<Book> book_list) throws IOException
    {
        FileOutputStream output = new FileOutputStream("database/output.txt");

        for (Book single_book : book_list)
        {
            ArrayList<String> result = single_book.get_all_information();
            output.write("title\t\t\t\t".getBytes(StandardCharsets.UTF_8));
            output.write(result.get(0).getBytes(StandardCharsets.UTF_8));
            output.write('\n');

            output.write("author\t\t\t\t".getBytes(StandardCharsets.UTF_8));
            output.write(result.get(1).getBytes(StandardCharsets.UTF_8));
            output.write('\n');

            output.write("classification\t\t".getBytes(StandardCharsets.UTF_8));
            output.write(result.get(2).getBytes(StandardCharsets.UTF_8));
            output.write('\n');

            output.write("publication\t\t\t".getBytes(StandardCharsets.UTF_8));
```

```
49.             output.write(result.get(3).getBytes(StandardCharsets.UTF_8));
50.             output.write('\n');
51.
52.             output.write("picture_link\t\t".getBytes(StandardCharsets.UTF_8));
53.             output.write(result.get(4).getBytes(StandardCharsets.UTF_8));
54.             output.write('\n');
55.
56.             output.write("price\t\t\t\t".getBytes(StandardCharsets.UTF_8));
57.             output.write(result.get(5).getBytes(StandardCharsets.UTF_8));
58.             output.write('\n');
59.
60.             output.write("content_description\t".getBytes(StandardCharsets.UTF_8));
61.             output.write(result.get(6).getBytes(StandardCharsets.UTF_8));
62.             output.write('\n');
63.
64.             output.write("author_description\t".getBytes(StandardCharsets.UTF_8));
65.             output.write(result.get(7).getBytes(StandardCharsets.UTF_8));
66.             output.write('\n');
67.
68.             output.write("catalog\t\t\t\t".getBytes(StandardCharsets.UTF_8));
69.             output.write(result.get(8).getBytes(StandardCharsets.UTF_8));
70.             output.write('\n');
71.             output.write('\n');
72.         }
73.     }
74.
75.     public ArrayList<Book> Crawl() throws IOException, InterruptedException
76.     {
77.         ArrayList<Book> book_lists = new ArrayList<Book>();
78.         Elements page_element = this.document.getElementsByClass("null");
79.         int page_number = Integer.parseInt(page_element.last().text());
80.
81.         System.setProperty("webdriver.chrome.driver", "C:/Program Files (x86)/Google/Chrome/Application/chromedriver.exe");
82.         WebDriver driver = new ChromeDriver();
83.
84.         try
85.         {
86.             for (int page_index = 1; page_index <= page_number; page_index++)
87.             {
88.                 String current_page_url = "http://search.dangdang.com/?key=" + this.keyword + "&action=input&page_index=" + Integer.toString(page_index);
89.                 driver.get(current_page_url);
90.
91.                 Thread.sleep(page_index == 1 ? 10000 : 2000);
92.
93.                 this.document = Jsoup.parse(driver.getPageSource());
94.                 Elements book_list = this.document.getElementById("search_nature_rg").getAllElements().first().getElementsByTag("li");
95.
96.                 for (Element book_element : book_list)
97.                 {
98.                     HashMap<String, String> single_book = new HashMap<String, String>();
99.                     String book_link = book_element.child(0).attr("href");
100.                        System.out.println("The book's link we have crawled: " + book_link);
101.                        driver.get(book_link);
102.                        Document document = Jsoup.parse(driver.getPageSource());
```

```
103.
104.
105.                         Element main_information = document.getElementsByClass("prod
    uct_main clearfix").first();
106.                         Element picture_information = main_information.getElementByI
    d("largePicDiv");
107.                         Element product_information = main_information.getElementByI
    d("product_info");
108.                         Element author_description = document.getElementById("author
    Introduction");
109.                         Element content_description = document.getElementById("conte
    nt");
110.                         Element catalog = document.getElementById("catalog-show");
111.
112.                         String title = "Unknown", author = "Unknown", classification
     = "Unknown", publication = "Unknown", picture_link = "Unknown", price = "Unknown",
     content = "Unknown", author_desc = "Unknown", catalog_content = "Unknown";
113.                         try
114.                         {
115.                             title = product_information.child(0).child(0).attr("titl
    e");
116.                             author = product_information.child(1).child(0).child(0).
    text();
117.                             classification = document.getElementById("detail-
    category-path").text();
118.                             publication = product_information.child(1).child(1).chil
    d(0).text();
119.                             picture_link = picture_information.child(0).child(0).att
    r("src");
120.                             price = product_information.getElementById("dd-
    price").text();
121.                             content = content_description.child(1).text().replaceAll
    ("\n", "");
122.                             author_desc = author_description.child(1).text().replace
    All("\n", "");
123.                             catalog_content = catalog.text().replaceAll("\n", "");
124.                         }
125.                         catch (Exception e)
126.                         {
127.                             System.out.println(e.getMessage());
128.                         }
129.                         finally
130.                         {
131.
132.                         }
133.
134.                         book_lists.add(new Book(title, author, classification, publi
    cation, picture_link, price, content, author_desc, catalog_content));
135.                         Thread.sleep(1000);
136.                     }
137.                 }
138.             }
139.             catch (Exception e)
140.             {
141.                 System.out.println("Sorry the crawler may be detected by dangdang.\n
    \n");
142.             }
143.
144.             driver.quit();
145.
146.             return book_lists;
```

```
        }
    }
```

3. Index.java

```
import java.io.*;
import java.util.ArrayList;

import org.apache.lucene.analysis.Analyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.document.Field;
import org.apache.lucene.document.TextField;
import org.apache.lucene.index.DirectoryReader;
import org.apache.lucene.index.Fields;
import org.apache.lucene.index.IndexWriter;
import org.apache.lucene.index.IndexWriterConfig;
import org.apache.lucene.queryparser.classic.MultiFieldQueryParser;
import org.apache.lucene.queryparser.classic.ParseException;
import org.apache.lucene.queryparser.classic.QueryParser;
import org.apache.lucene.search.*;
import org.apache.lucene.store.Directory;
import org.apache.lucene.store.FSDirectory;
import org.apache.lucene.util.Version;
import org.wltea.analyzer.lucene.IKAnalyzer;


public class Index
{
    public void create_index(String filePath)
    {
        File f = new File(filePath);
        IndexWriter iwr = null;
        try
        {
            Directory dir = FSDirectory.open(f);
            Analyzer analyzer = new IKAnalyzer();

            IndexWriterConfig conf = new IndexWriterConfig(Version.LUCENE_4_10_0, a
nalyzer);
            iwr = new IndexWriter(dir, conf);

            ArrayList<Document> document_list = get_document();
            for (Document element : document_list)
            {
                iwr.addDocument(element);
            }
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
        try
        {
            assert iwr != null;
            iwr.close();
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
```

```
55.     }
56.
57.     public ArrayList<Document> get_document() throws IOException
58.     {
59.         ArrayList<Document> document_list = new ArrayList<Document>();
60.         FileInputStream input = new FileInputStream("database/output.txt");
61.         String[] source = new String(input.readAllBytes()).split("\n\n");
62.
63.         for (int i = 0; i < source.length; i++)
64.         {
65.             String[] single_book = source[i].split("\n");
66.             Document temp_document = new Document();
67.
68.             temp_document.add(new TextField("title", single_book[0], Field.Store.YE
    S));
69.             temp_document.add(new TextField("author", single_book[1], Field.Store.Y
    ES));
70.             temp_document.add(new TextField("classification", single_book[2], Field
    .Store.YES));
71.             temp_document.add(new TextField("publication", single_book[3], Field.St
    ore.YES));
72.             temp_document.add(new TextField("picture_link", single_book[4], Field.S
    tore.YES));
73.             temp_document.add(new TextField("price", single_book[5], Field.Store.YE
    S));
74.             temp_document.add(new TextField("content_description", single_book[6],
    Field.Store.YES));
75.             temp_document.add(new TextField("author_description", single_book[7], F
    ield.Store.YES));
76.             temp_document.add(new TextField("catalog", single_book[8], Field.Store.
    YES));
77.
78.             document_list.add(temp_document);
79.         }
80.
81.         return document_list;
82.     }
83.
84.     public ArrayList<Book> search(String filePath, String keyword, String[] fields,
     int choice)
85.     {
86.         File f = new File(filePath);
87.         ArrayList<Book> result = new ArrayList<Book>();
88.
89.         try
90.         {
91.             IndexSearcher searcher = new IndexSearcher(DirectoryReader.open(FSDirec
    tory.open(f)));
92.             Analyzer analyzer = new IKAnalyzer();
93.             Query query = null;
94.
95.             if (choice == 1)
96.             {
97.                 QueryParser parser = new QueryParser(Version.LUCENE_4_10_0, fields[
    0], analyzer);
98.                 query = parser.parse(keyword);
99.             }
100.                else
101.                {
102.                    MultiFieldQueryParser parser = new MultiFieldQueryParser(fields,
     analyzer);
```

```
                    query = parser.parse(keyword);
                }

                TopDocs hits = searcher.search(query, 10);
                System.out.println("We have find " + hits.scoreDocs.length + " results in total.\n");

                for (ScoreDoc document : hits.scoreDocs)
                {
                    Document temp_document = searcher.doc(document.doc);
                    Book single_book = new Book(
                            temp_document.get("title"),
                            temp_document.get("author"),
                            temp_document.get("classification"),
                            temp_document.get("publication"),
                            temp_document.get("picture"),
                            temp_document.get("price"),
                            temp_document.get("content_description"),
                            temp_document.get("author_description"),
                            temp_document.get("catalog")
                    );

                    result.add(single_book);
                }
            }
            catch (IOException | ParseException e)
            {
                e.printStackTrace();
            }

            return result;
        }
    }
```

4. test.java

```
import javax.crypto.AEADBadTagException;
import java.io.File;
import java.io.IOException;
import java.io.Writer;
import java.lang.reflect.UndeclaredThrowableException;
import java.util.ArrayList;
import java.util.Scanner;

public class test
{
    public static void main(String[] args) throws IOException, InterruptedException

    {
        Preparation();

        /* --------------------
- To skip the crawling process, comment out the following 6 lines ----------------
--- */
        /*
         *  System.out.println("Now please input the keyword that you want to search from http://www.dangdang.com/");
         *  String keyword = Input();
         *  System.out.println("Now we are going to crawl the information you need from website.");
```

```
20.          *  System.out.println("Before the chromedriver exits, do not do anything t
    o ensure the process is normal.");
21.
22.          *  Crawl(keyword);
23.          */
24.
25.         while (true)
26.         {
27.             System.out.println("There are two modes for you.");
28.             System.out.println("1.\tSearch by a single attribute.");
29.             System.out.println("2.\tSearch by all attributes.");
30.             System.out.println("3.\tQuit the program.");
31.
32.             System.out.print("Now please input your choice as an integer: ");
33.             int choice = Integer.parseInt(Input());
34.             String[] fields = new String[]{"title,", "author", "classification", "p
    ublication", "picture_link", "price", "content_description", "author_description",
    "catalog"};
35.
36.             System.out.println("Now please input another keyword that you want to s
    earch");
37.             String keyword = Input();
38.             if (choice == 1)
39.             {
40.                 System.out.println("Now please input the attribute's name");
41.                 String attribute = new Scanner(System.in).nextLine();
42.                 fields[0] = attribute;
43.             }
44.             else if (choice == 3)
45.             {
46.                 break;
47.             }
48.
49.             ArrayList<Book> book_list = Create_Index(keyword, fields, choice);
50.             Output(book_list);
51.         }
52.
53.         End();
54.     }
55.
56.     public static void Preparation()
57.     {
58.         System.out.println("This program is developed by Leming Shen, majoring in S
    oftware Engineering,");
59.         System.out.println("Copyright © 2020 College of Computer Science and Techno
    logy, Zhejiang University.");
60.         System.out.println("Email to ZJU.SLM@gmail.com for more information. All ri
    ghts reserved.\n");
61.     }
62.
63.     public static String Input()
64.     {
65.         return new Scanner(System.in).nextLine();
66.     }
67.
68.     public static void Crawl(String keyword) throws IOException, InterruptedExcepti
    on
69.     {
70.         Dangdang_Crawler crawler = new Dangdang_Crawler(keyword);
71.         System.out.println("Crawling . . . . . .\n");
72.         ArrayList<Book> result = crawler.Crawl();
```

```
73.
74.         System.out.println("\nAll right! We have get all the information!");
75.         System.out.println("Now we are writing the data to a local file . . . . . .
    ");
76.
77.         crawler.Write(result);
78.         System.out.println("Data being stored completely!\n");
79.     }
80.
81.     public static ArrayList<Book> Create_Index(String keyword, String[] fields, int
     choice)
82.     {
83.         Index book_index = new Index();
84.
85.         System.out.println("\nNow we are going to build a index to the document we
    got earlier.");
86.         File directory = new File("");
87.         String filePath = directory.getAbsolutePath() + "\\database\\index";
88.
89.         book_index.create_index(filePath);
90.         return book_index.search(filePath, keyword, fields, choice);
91.     }
92.
93.     public static void Output(ArrayList<Book> book_list)
94.     {
95.         System.out.println("Here are the list of books' information we got for you
    :)");
96.
97.         for (int i = 0; i < book_list.size(); i++)
98.         {
99.             System.out.println("Book number " + (i + 1) + ".");
100.
101.                Book single_book = book_list.get(i);
102.                ArrayList<String> information = single_book.get_all_information();
103.
104.                for (String element : information)
105.                {
106.                    System.out.println(element);
107.                }
108.
109.                System.out.println();
110.            }
111.        }
112.
113.        public static void End()
114.        {
115.            System.out.println("\n\nThank you for using! Bye!");
116.        }
117.    }
```