浙江大学计算机科学与技术学院

Java 程序设计课程报告

2015—2016 学年冬学期

题目	基于 Java 爬虫的 Web 搜索引擎
学号	3160104734
学生姓名	沈子衿
所在专业	软件工程
所在班级	软件工程 1601 班

目　　录

1 引言

本次，我们开发的是一个基于 Java 爬虫的 Web 搜索引擎。该搜索引擎分为两个子模块：爬虫模块和搜索模块。通过爬取问答类或课程类网站的网页信息并建立索引的方式实现在线/离线的搜索功能。

本次实验，我们爬取的对象是丁香园（https://dxy.com/diseases/）。

1．1 设计目的

基于 Java 爬虫的 Web 搜索引擎的具体功能和设计思路如下：

(1) 搜索引擎本质上是两个子模块的结合，即爬虫模块和搜索模块；

(2) 爬虫模块基于 Java 的 JSoup 爬虫库进行丁香园网页的爬取和解析。在本模块中，丁香园网页上各类疾病的病因、诊断方式、预防方式等相关信息将被爬取并结构化，存储在本地的对应根目录下。

(3) 搜索模块基于 Lucene 库进行索引的建立和信息的检索。在本模块中，读取到本地的结构化信息将被格式化为便于处理的 Document 对象（以信息中的问题-答案为单位），Lucene 库相关函数基于这些对象建立内容索引，并提供关键字的检索支持。用户可以通过命令行进行内容检索，并展示内容列表。

(4) 原则上，爬虫模块和搜索模块是高度耦合的，即，程序启动后，先执行爬虫模块代码将信息爬取下来，再执行搜索模块代码建立索引和交互。但是，由于丁香园网站部署了较为严密的反爬策略，爬取一定数量的信息后 IP 地址会被禁止访问 10min，而为了最大限度地降低被封锁的频率，我基于代理服务器、客户端伪装和随机休眠的方式部署了简易的反反爬策略，其结果是信息爬取时间十分漫长（在部署 3 个服务器+随机休眠间隔 1000±700ms 的条件下，时间约为两小时）。显然，我们不能每运行一次程序就执行一次爬虫。因此，在本次提交的作业中，我将两个模块解耦了。即，搜索模块可以基于现成数据集单独运行。

1．2　设计说明

本程序采用 Java 程序设计语言，在 Jetbrains 基金会开发的 Intellij IDEA 平台下编辑、编译与调试。具体程序由我一人开发而成。

表 1 各成员分工表

成员名称	完成的主要工作	
	程序设计	**课程报告**
沈子衿	负责整个程序前期的需求分析和整体功能的架构 程序后期的测试与运行	报告的第 1 章、第 2 章和第 4 章
沈子衿	负责程序中爬虫模块的设计与编码	报告的第 3 章
沈子衿	负责程序中搜索模块与用户交互部分的设计与编码	目录、总结和参考文献的整理 报告后期的格式设置

2. 总体设计

2．1　功能模块设计

本程序需实现的主要功能有：

(1) 使用 JSoup 爬虫爬取丁香园网站上的信息，并对信息进行抽取、分组和结构化；

(2) 基于代理服务器、实现基本的反制反爬虫策略；

(3) 将结构化的信息进一步模块化成为 Document 对象，并基于这些对象建立索引；

(4) 基于已经建立的索引，设计带有关键词检索功能的命令行交互界面；

程序的总体功能如图 1 所示：

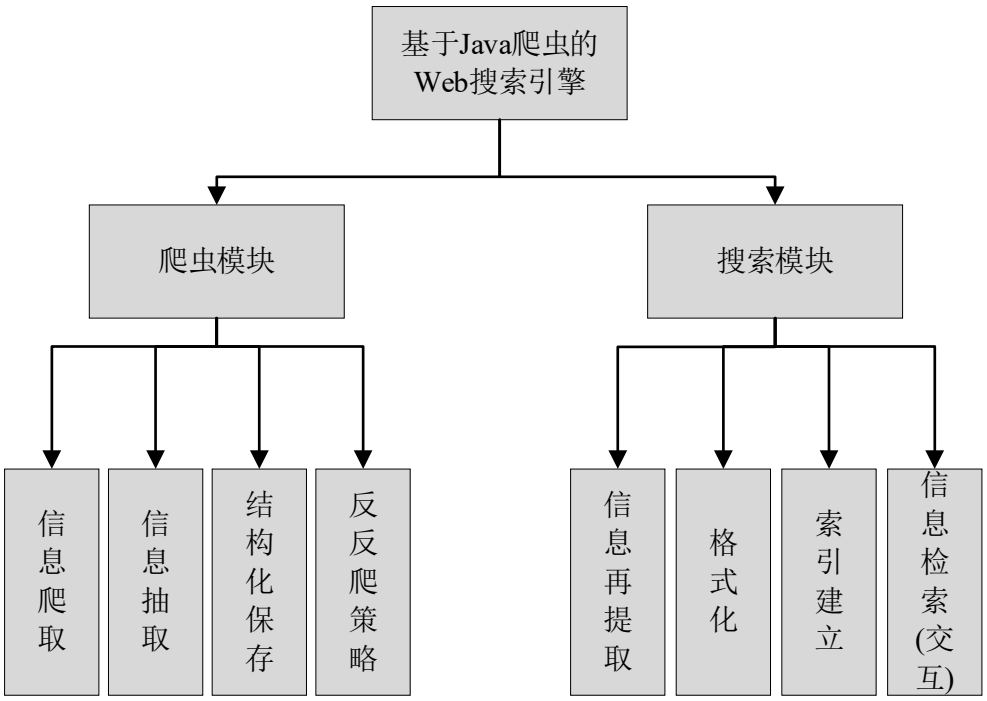


图 1 总体功能图

2.2 流程图设计

爬虫模块（实现逻辑）总体流程如图 2 所示：

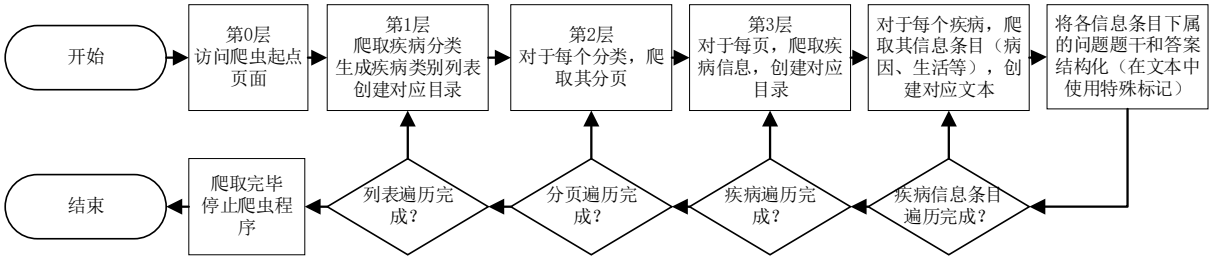


图 2 爬虫模块（实现逻辑）总体流程图

爬虫模块（反反爬策略）总体流程如图 3 所示：

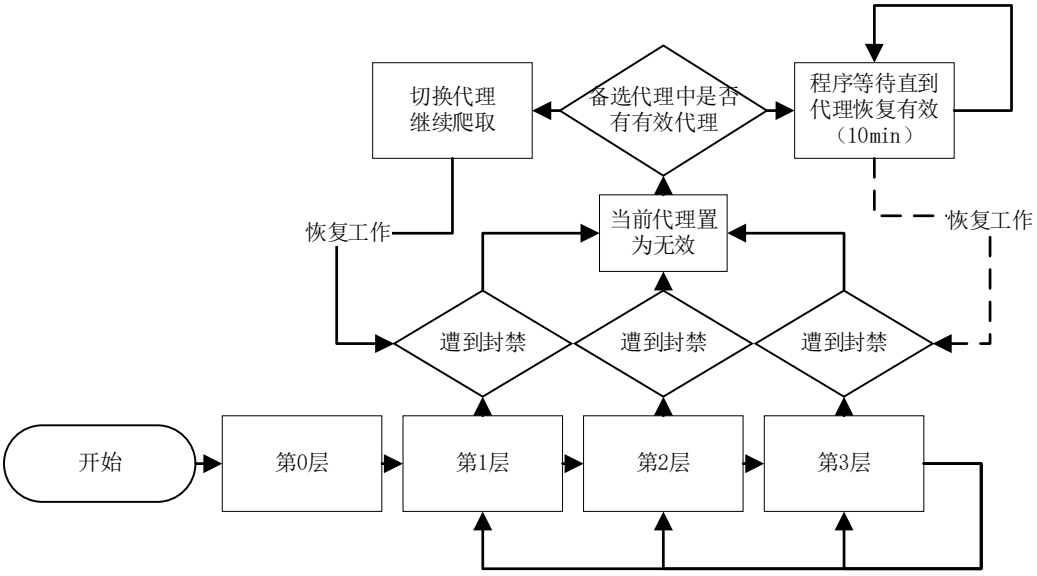


图 3 爬虫模块（反反爬策略）总体流程图

搜索模块（索引建立）总体流程如图 4 所示：

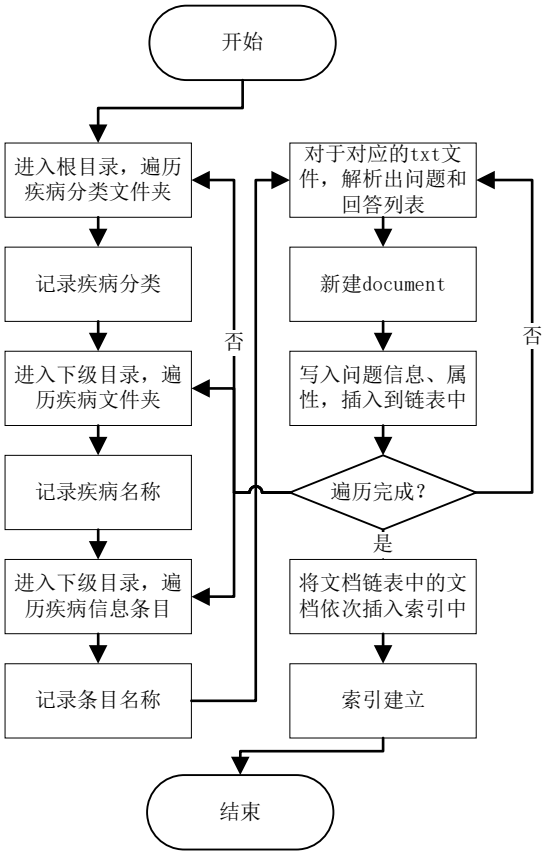


图 4 搜索模块（索引建立）总体流程图

搜索模块（信息检索与交互）总体流程如图 5 所示：

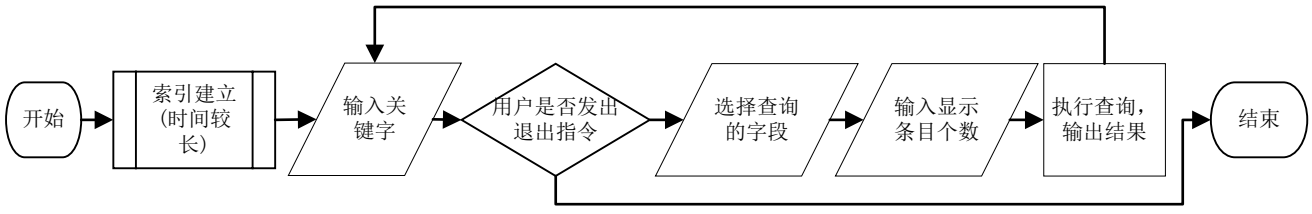


图 5 搜索模块（信息检索与交互）总体流程图

2. 详细设计

3.1 爬虫模块基本逻辑设计

爬虫模块的基本逻辑比较清楚明了。在叙述代码逻辑之前，我们首先需要确定爬取对象——丁香园的网页结构。

图 6 丁香园主页

打开丁香园主页（如图 6 所示），可以看到清晰的疾病条目和疾病分类。我们首先需要明确的，是从何处开始爬取、以及以何种策略开始爬取。经过调查，我认为页面上方的“疾病分类（科室）”标签可以作为入手点。第一，“科室”标签的 URL 脉络清晰、格式统一；第二，不同的“科室”之间原则上没有交集，可以避免数据冗余；第三， 便于对数据进行进一步分类。因此，我选择按照科室进行爬取。相关代码如下：

```
1.  // 使用 class 名访问科室链接所在的 navbar
2.              Elements diseaseClusters = doc.getElementsByClass("section-navbar-item");
3.              System.out.println(title);
4.              // 对于每个科室
5.              int count = 0;
6.              // 遍历所有科室
7.              for(Element diseaseCluster:diseaseClusters)
8.              {
9.                  count++;
10.                 if(count < 2) continue;
11.                 System.out.println(diseaseCluster.text());
12.                 // 为每个科室在本地新建目录
13.                 String path = "D:\\javaio\\";
```

```
                File file = new File("D:\\javaio\\" + diseaseCluster.text());
                if(!file.exists())
                {
                    file.mkdir();
                }
                // 获取当前科室链接
                Element diseaseClusterLink = diseaseCluster.select("a").first();
                // 获取当前科室链接
                String diseaseClusterHref = diseaseClusterLink.attr("href");
                Random rand = new Random();
                // 随机休眠，规避反爬策略
                Thread.sleep( rand.nextInt(1000));
                Document innerDoc = new Document("xxx");
                // 反反爬处理
                for(;;)
                {
                    try
                    {
                        // 进入科室链接，获取当前科室对应的页面内容
                        Connection cl = JSoup.connect(diseaseClusterHref).proxy(pflag?p1:nul
    l).ignoreContentType(true).userAgent(agent);
                        innerDoc = cl.get();
                        break;
                    }
                    catch (Exception e)
                    {
                        // 切换线路
                        if(pflag == false)
                        {
                            pflag = true;
                            Thread.sleep(500);
                        }
                        // 两个线路全部被禁止访问，等待十分钟
                        else
                        {
                            pflag = false;
                            System.out.println("retry...");
                            Thread.sleep(610000);
                        }
                    }
                }
```

接下来的任务，是爬取科室页面下的疾病列表。在这里我遇到了一个问题：科室页面有一个基于动态分页的反爬虫机制。当用户打开科室页面时，页面上只会

显示有限数量的疾病信息，只有当用户作出“拖动网页往下滚动”这一操作时，新的疾病信息才会被加载出来。

这对试图一次性获取完整列表的我是一个挑战。经过调查研究，我发现，这种动态分页机制本质上也是通过URL实现的——只需要在第一页的URL后加上page参数，指定页码就可以了（以皮肤性病科为例，访问其疾病列表第二页的方式为https://dxy.com/diseases/pifuxingbing/**?page=2**）。此外，每一个科室对应疾病列表的最大页数也是明文写在 html 里的，这为我们遍历所有页面提供了方便。如图7所示：

图 7 丁香园动态分页机制原理与破解

相关代码如下：

```
int max_page = 0;
                // 定位到存储最大页数的 pagination 位置（虽然类型是 Elements 但实际上只有一个）
                Elements page_elements = innerDoc.getElementsByClass("pagination");
                for(Element page_element:page_elements)
                {
                    // 获取存储最大页码的表项内容（aria-label 属性中包含 page 字段）
                    String page = page_element.select("[aria-label^=Page]").last().text();
                    //将表项内容解析为整型数值
                    max_page = Integer.parseInt(page);

```

```
11.                 }
12.                 System.out.println("max page : " + max_page);
13.                 // 对于存在的每一页，依次请求
14.                 for(int i = 1; i <= max_page; i++){/*------*/}
```

接下来，我们需要依次访问爬取下来的疾病信息页面，并进行分割和结构化。疾病信息页面如图所示（以皮肤性病科——sweet 综合征页面为例）：

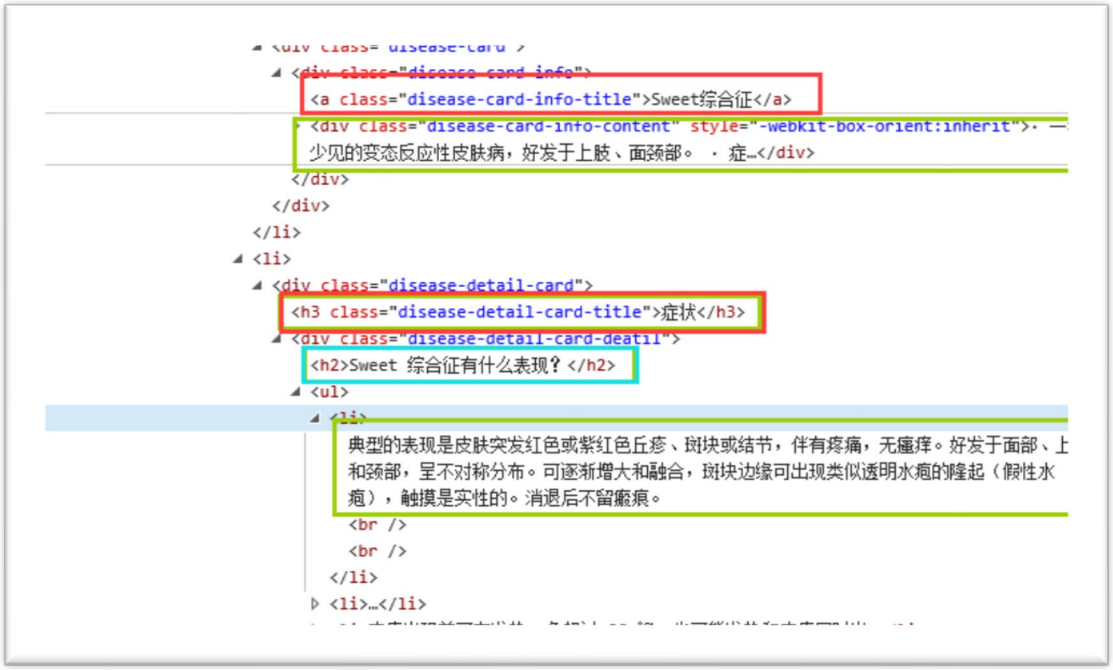


图 8 丁香园疾病信息页面结构解析

可以看出，丁香园疾病信息页面的基本结构为：信息条目（问题分类）——问题与解答。由于网站编辑者的问题，也有少数没有归类的信息条目（比如一开始的疾病定义）、部分空信息条目和没有问题题干的解答。

经过抽样分析，考虑到空信息条目的现象很少（即使存在，也比较容易做特殊处理），而问题题干缺失、问题解答格式混乱的情况则非常严重，我决定采用每一个疾病和对应科室目录下的同名本地目录对应、每一个条目和目录下的 txt 文档对应（如症状.txt， 病因.txt），而问题和答案则以结构化的方式存储在 txt 中，不单独保存为文件。结构化保存方式如图：

图 9 本地结构化保存方式

问题和答案不单独保存还有一个原因：我的室友比我先完成本次作业，它试图将每个问题都保存为一个文件，其后果是产生了大量磁盘碎片，同时使磁盘 IO 时间延长了许多，降低了爬虫和索引的效率。这是我力图避免的。

对应代码实现如下：

```
1.  public static void processSinglePage(String url, String path) throws Exception
2.      {
3.          try {
4.                  Document mainPage = new Document("xxx");
5.                  for(;;) {
6.                      try {
7.                          // 连接到对应疾病信息页面
8.                          mainPage = JSoup.connect(url).ignoreContentType(true).proxy(pflag?p1
    :null).userAgent(agent).get();
9.                          break;
10.                     } catch (Exception e) {
11.                         // 切换线路
12.                         if(pflag == false){
13.                             pflag = true;
14.                             Thread.sleep(700);
15.                         }
16.                         // 两个线路都被禁用，等待 10 分钟
17.                         else{
18.                             pflag = false;
19.                             System.out.println("retry...");
20.                             Thread.sleep(610000);
21.                         }
22.                     }
23.                 }
24.                 // 获取当前疾病的名称
25.                 String mainTitle = mainPage.getElementsByClass("disease-card-info-
    title").first().text(); // 保存为文件名用
26.                 // 处理可能存在的异常字符
27.                 mainTitle = mainTitle.replace('/', '-');
28.                 System.out.println(mainTitle);
29.                 File file = new File(path + "\\" + mainTitle);
30.                 // 如果当前疾病对应的目录不存在，创建对应的目录
31.                 if(!file.exists())
32.                 {
33.                     file.mkdir();
34.                 }
35.                 //
36.                 // 获取疾病基本信息
```

```
                Element firstIntro = mainPage.getElementsByClass("disease-card-info-content").first();
                System.out.println("--------" + firstIntro.text());
                // 解析条目列表
                Elements attributeList = mainPage.getElementsByClass("disease-detail-card");
                // 对于每一个条目
                for(Element attribute : attributeList)
                {
                    // 解析条目标题
                    String attrTitle = attribute.select(".disease-detail-card-title").first().text();
                    // 解析条目内容
                    String attrContent = attribute.select(".disease-detail-card-deatil").first().text();
                    // 写入文件

                    String attrText = "";
                    Element attrElement = attribute.select(".disease-detail-card-deatil").first();
                    Elements attrSingles = attrElement.children();
                    // 对于每一个条目下面的 Q/A
                    for(Element child : attrSingles)
                    {
                        // 如果是问题题干，在前面加上特殊标识
                        if(child.is("h2"))
                        {
                            attrText += ("\n##Q:" + child.text() + "\n");
                        }
                        // 如果是普通内容
                        else
                        {
                            attrText += (child.text() + "\n");
                        }
                    }
                    // 在文件最末尾加上结束标识
                    attrText += "##";
                    BufferedWriter bw=new BufferedWriter(new FileWriter(path + "\\" + mainTitle + "\\" + attrTitle + ".txt"));
                    // 将结构化的条目内容写入对应路径下的对应文本文件中
                    bw.write(attrText);
                    bw.close();
                }
                // 将来源写入对应文件夹的 source.txt 中存储
```

```
75.                 BufferedWriter bw=new BufferedWriter(new FileWriter(path + "\\" + mainTitle
    + "\\" + "source.txt"));
76.                 bw.write(url + "\n");
77.                 // 关闭文件
78.                 bw.close();
79.         }catch(HttpStatusException e){
80.             e.printStackTrace();
81.         }
82.     }
```

至此，信息爬取和抽取工作基本完成。

3.2 反爬虫应对策略设计

在爬虫模块中还有一个悬而未决，但是不可忽视的问题——丁香园是有严格的反爬虫机制的。除了我们上一小节提到的动态分页策略，丁香园的反爬策略主要体现在，相同 IP 在一段时间内请求网站信息超过一定次数（查阅资料了解到该次数为 30），该 IP 就会被列入网站黑名单 10min。在这 10min 里，该 IP 请求丁香园任何页面都将返回 404 错误（包括使用浏览器访问）。

图 10 被禁止访问后的 404 页面

一般的应对策略无非是进行客户端伪装（伪装成浏览器）、每请求一次后随机休眠和配置代理，但经过我的尝试，这些策略只能尽量延长被禁止访问前爬虫运行的时间，而不能从根本上规避丁香园的反爬机制。

经过测试，我选择结合以上三种策略部署简易应对机制。配置如下：

- 客户端伪装："Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)" + "Chrome/56.0.2924.87 Safari/537.36";
- 随机休眠：0 – 1000 ms

- 服务器：2 台（包括本机）

最终，整个爬虫过程花费了 2 小时 25 分钟左右。

在撰写报告的过程中，我尝试将代理服务器数量增加了 1 台（服务器数量 3 台），随机休眠时间调整为 300-1700ms（1000±700ms），测得花销为 1 小时 20 分钟左右。

我的某个同学使用了 3 台服务器，但将随机休眠范围扩大到了 3000ms，花销也相应地扩大到接近 3 小时。他也证明，小范围扩大随机休眠时间可以请求的次数有时可能会超过 30 次，但这一增益并不会随着休眠时间的无限扩大而扩大，即便扩大到接近半分钟也无济于事。最有效的方式是使用更多的代理服务器。

由于测试的时间开销过大，且购置代理服务器需要一定花销，我没有进行进一步测试。

以 2 台服务器为例，我的应对机制主要包含以下步骤：首先，每次请求前，程序会进行 0-1000ms 的随机休眠；其次，每次请求都使用代理，默认使用本机代理，当本机 IP 被禁用（connect 请求返回 404）后，程序会将使用代理的 flag 置为 true，换用代理服务器继续请求、爬取；当代理服务器也被禁用后，考虑到此时本机依然处于被禁用状态，程序会休眠 10min（实际上不需要休眠这么久， 设置为 10min 是为了确保禁用全部解除），10min 中后以本机为代理开始下一轮请求。直到爬取完成。

配置逻辑如下）：

```
1.  static Proxy p1 = new Proxy(Proxy.Type.HTTP, new InetSocketAddress("116.255.170.46",16819));
2.  static Proxy p2 = new Proxy(Proxy.Type.HTTP, new InetSocketAddress("183.196.170.247",9000));
3.  static String agent ="Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Ge
    cko)" + "  Chrome/56.0.2924.87 Safari/537.36";
```

服务器切换请求逻辑如下（以请求科室为例）：

```
1.  for(;;)
2.                  {
3.                      try
4.                      {
5.                          // 进入科室链接，获取当前科室对应的页面内容，pflag 判断是否使用代理
6.                          Connection cl = JSoup.connect(diseaseClusterHref).proxy(pflag?p1:nul
    l).ignoreContentType(true).userAgent(agent);
```

```
                    innerDoc = cl.get();
                    break;
                }
                catch (Exception e)
                {
                    // 切换线路
                    if(pflag == false)
                    {
                        pflag = true;
                        Thread.sleep(500);
                    }
                    // 两个线路全部被禁止访问，等待十分钟
                    else
                    {
                        pflag = false;
                        System.out.println("retry...");
                        Thread.sleep(610000);
                    }
                }
            }
```

3.3 搜索模块索引建立步骤设计

获取数据集的下一步是使用 Lucene 库对数据集建立索引。Lucene 库封装了绝大多数索引建立操作，我只需要建立数据单元 Document,然后将这些 Document 依次添加到 IndexWriter 对象中，再调用相应方法即可。问题的关键是我们如何划分 Document。

经过研究，我决定以问题-回答为单位划分 Document：

```
Field fid = new TextField("编号",Integer.toString(docID),Field.Store.YES);
Field fquestion = new TextField("问题",q,Field.Store.YES);
Field fanswer = new TextField("回答",a,Field.Store.YES);
Field fillName = new TextField("病名",disease.getName(),Field.Store.YES);
Field fcluster = new TextField("科室",cluster.getName(),Field.Store.YES);
Field fscope = new TextField("范畴",attr.getName(),Field.Store.YES);
Field furl = new TextField("来源",url,Field.Store.YES);
```

每一个 Document 包含：一个递增的问题编号、问题题干、答案、病名、科室、范畴（条目）和来源。问题和答案通过 txt 中的特殊符号来解析、切割，再依次写入对应的 Document。注意，这里之所以使用 textField，是因为 textFeild 不会执行全字匹配：

```
tempFile = tempFile.replace('/','-');
```

```
2.  tempFile = tempFile.replace('\\','-');
3.  // 对 document 进行切割
4.  String[] mixList = tempFile.split("##");
5.  for(String member : mixList){...}
```

```
6.   for(String member : mixList)
7.       {
8.           member = member.trim();
9.           if(!member.isEmpty())
10.          {
11.              String q;
12.              String a;
13.              // member 是一个问题和答案的集合
14.              int line = member.indexOf("\n");
15.              // 如果没有问题题干，就是用问题范畴（条目）作为题干
16.              if(line == -1)
17.              {
18.                  q = attr.getName();
19.                  a = member;
20.              }
21.              else
22.              {
23.                  q = member.substring(0, line).trim();
24.                  a = member.substring(line).trim(); // 也可以 line + 1
25.              }
26.              // 新建一个 document 并把信息导入.注意这里还没有添加来源
27.              doc = new Document();
28.              Field fid = new TextField("编号",Integer.toString(docID),Field.Store.YES);
29.              Field fquestion = new TextField("问题",q,Field.Store.YES);
30.              Field fanswer = new TextField("回答",a,Field.Store.YES);
31.              Field fillName = new TextField("病名",disease.getName(),Field.Store.YES);
32.              Field fcluster = new TextField("科室",cluster.getName(),Field.Store.YES);
33.              Field fscope = new TextField("范畴",attr.getName(),Field.Store.YES);
34.              Field furl = new TextField("来源",url,Field.Store.YES);
35.              // 将信息加入 document
36.              doc.add(fid);
37.              doc.add(fquestion);
38.              doc.add(fanswer);
39.              doc.add(fillName);
40.              doc.add(fcluster);
41.              doc.add(fscope);
42.              doc.add(furl);
43.              docList.add(doc);
44.              // docID 增加
45.              docID++;
```

```
46.         }
47.     }
```

然后将 Document 列表中的 Document 依次添加入索引对象，直到索引建立完成：

```
1.  public void createIndex(String filePath){
2.          File f=new File(filePath);
3.          iwr=null;
4.          try {
5.              Directory dir=FSDirectory.open(f);
6.              Analyzer analyzer = new IKAnalyzer();
7.              IndexWriterConfig conf=new IndexWriterConfig(Version.LUCENE_4_10_0,analyzer);
8.              iwr=new IndexWriter(dir,conf);//建立 IndexWriter。固定套路
9.              try {
10.                 // 清空之前建立的索引
11.                 iwr.deleteAll();
12.             }catch(Exception e){
13.                 e.printStackTrace();
14.             }
15.             // 获取所有 document
16.             LinkedList<Document> docs = getAllDocument();
17.             //添加 doc，Lucene 的检索是以 document 为基本单位
18.             for(Document doc : docs)
19.                 iwr.addDocument(doc);
20.         } catch (IOException e) {
21.             // TODO Auto-generated catch block
22.             e.printStackTrace();
23.         }
24.         try {
25.             iwr.close();
26.         } catch (IOException e) {
27.             // TODO Auto-generated catch block
28.             e.printStackTrace();
29.         }
30.     }
```

等待一段时间后，索引建立完成。

3.4 搜索模块命令行交互步骤设计

命令行交互逻辑较为简单：首先提示用户输入关键字，然后提示用户输入要查找的字段，最后提示用户输入显示结果的个数。

如果用户在开头输入!quit,则程序退出。

Lucene 建立的索引支持不完全匹配，但不支持模糊查找。

相关代码逻辑如下：

```
1.  public void search(String filePath){
2.          File f=new File(filePath);
3.          Scanner sc = new Scanner(System.in);
4.          try {
5.              while(true){
6.                  IndexSearcher searcher=new IndexSearcher(DirectoryReader.open(FSDirectory.op
    en(f)));
7.                  System.out.println("【请输入您要查找的关键字,输入!quit 退出】: ");
8.                  String en;
9.                  while((en = sc.nextLine().trim()).isEmpty());
10.                 String queryStr=en.trim();
11.                 if(queryStr.equals("!quit")){
12.                     System.out.println("再见! ");
13.                     return;
14.                 }
15.                 Analyzer analyzer = new IKAnalyzer();
16.                 //指定 field 为“name”，Lucene 会按照关键词搜索每个 doc 中的 name。
17.                 System.out.println("请选择该关键字适用的范围: 【1.问题 2. 来源 3. 回答 4. 病
    名 5. 科室 6. 问题种类（范畴）】");
18.                 int key = sc.nextInt();
19.                 String scope;
20.                 // 选择要查找的字段
21.                 switch(key)
22.                 {
23.                     case 1:
24.                         scope = "问题";
25.                         break;
26.                     case 2:
27.                         scope = "来源";
28.                         break;
29.                     case 3:
30.                         scope = "回答";
31.                         break;
32.                     case 4:
33.                         scope = "病名";
34.                         break;
35.                     case 5:
36.                         scope = "科室";
37.                         break;
38.                     case 6:
39.                         scope = "范畴";
```

```
                        break;
                    default:
                        scope = "问题";
                        break;
                }
                System.out.print("请选择结果输出个数：");
                int key_2 = 0;
                while(key_2 <= 0) {
                    key_2 = sc.nextInt();
                }
                QueryParser parser = new QueryParser(scope, analyzer);

                Query query=parser.parse(queryStr);
                TopDocs hits=searcher.search(query,key_2);//前面几行代码也是固定套路，使用时直接改 field 和关键词即可
                System.out.println("\n");
                // 遍历输出结果
                for(ScoreDoc doc:hits.scoreDocs) {
                    Document d = searcher.doc(doc.doc);
                    System.out.println("问题：" + d.get("问题"));
                    System.out.println("来源：" + d.get("来源"));
                    System.out.println("回答：" + d.get("回答"));
                    System.out.println("病名：" + d.get("病名"));
                    System.out.println("科室：" + d.get("科室"));
                    System.out.println("范畴：" + d.get("范畴"));
                    System.out.println("\n");
                }
            }
        } catch (IOException | ParseException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
```

4. 测试与运行

4.1 程序测试

经过数轮测试和优化，本次作业完成的两个程序目前均运行正常。

在测试的过程中，程序暴露出如下问题：

1. 程序运行时间过长。如前所述，我的爬虫程序在两个 IP 都被禁用

后会默认休眠 10min，但实际上并不需要休眠这么长时间；

2. 对特殊字符兼容性不强。在爬取的数据中出现‘/’‘\’等字符，且相关数据被作为文件/目录名时，程序可能会报错；
3. 命令行交互中循环输入键盘缓冲区问题。在命令行交互代码中，输入是通过 Scanner 对象实现的,但在第二次循环时，Scanner 对象的 nextLine 函数会读取上一次循环在键盘缓冲区中残留的换行符，造成程序出错；

对于问题 1，可以采用多线程轮换休眠的方式解决。但经过测试，在 2 台服务器的情况下时间间隔最多缩短到 8-9min（即单个代理单次运行时间为 1-2min），优化成本大于优化收益，故未作优化；

对于问题 2，我在代码中对每一个可能成为文件名的字符串进行检查，用转义字符或其他字符替换可能出现的非法字符；

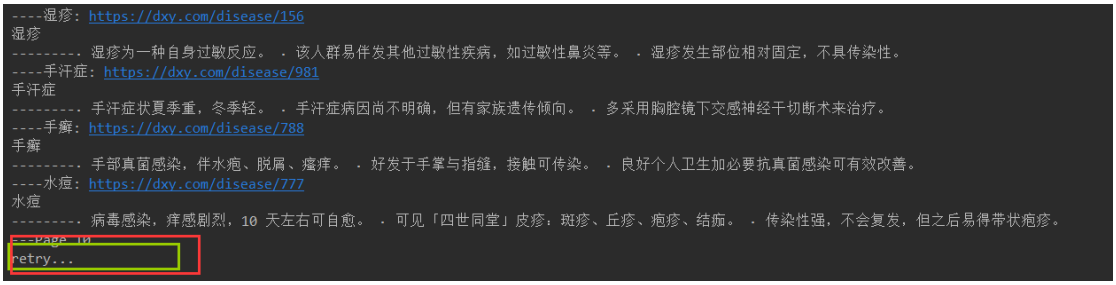
对于问题 3，我对循环的第一个 nextLine 进行了清空缓冲区的处理：

```
while((en = sc.nextLine().trim()).isEmpty());
```

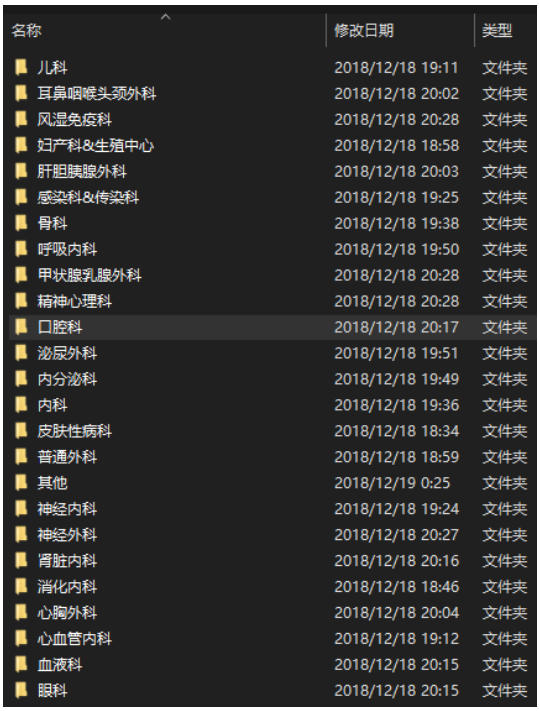
4.2 程序运行

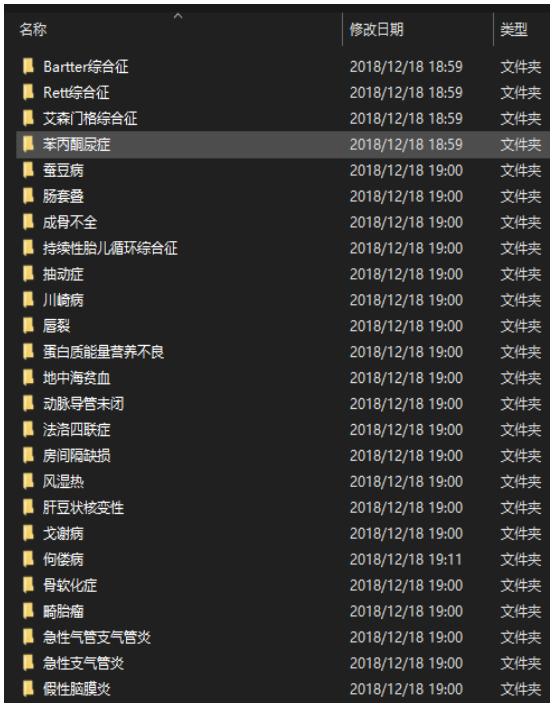
爬虫运行时的 Debug Console：

爬虫被反爬机制阻断：

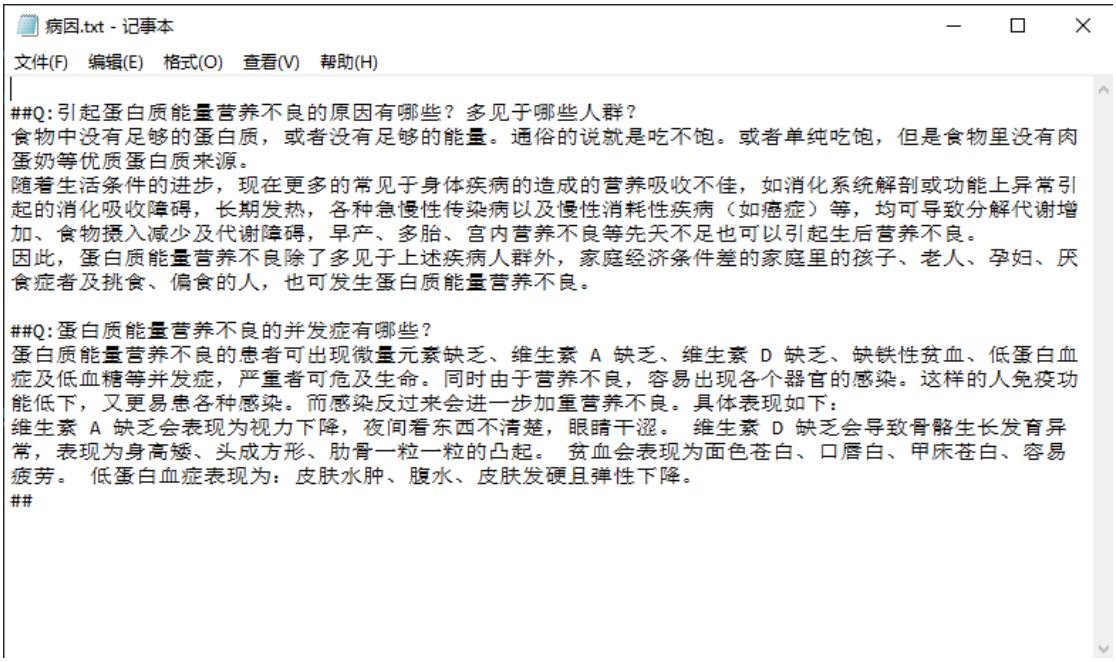

----湿疹：https://dxy.com/disease/156
湿疹
---------. 湿疹为一种自身过敏反应。 . 该人群易伴发其他过敏性疾病，如过敏性鼻炎等。 . 湿疹发生部位相对固定，不具传染性。
----手汗症：https://dxy.com/disease/981
手汗症
---------. 手汗症状夏季重，冬季轻。 . 手汗症病因尚不明确，但有家族遗传倾向。 . 多采用胸腔镜下交感神经干切断术来治疗。
----手癣：https://dxy.com/disease/788
手癣
---------. 手部真菌感染，伴水疱、脱屑、瘙痒。 . 好发于手掌与指缝，接触可传染。 . 良好个人卫生加必要抗真菌感染可有效改善。
----水痘：https://dxy.com/disease/777
水痘
---------. 病毒感染，痒感剧烈，10 天左右可自愈。 . 可见「四世同堂」皮疹，斑疹、丘疹、疱疹、结痂。 . 传染性强，不会复发，但之后易得带状疱疹。
---Page 10
retry...

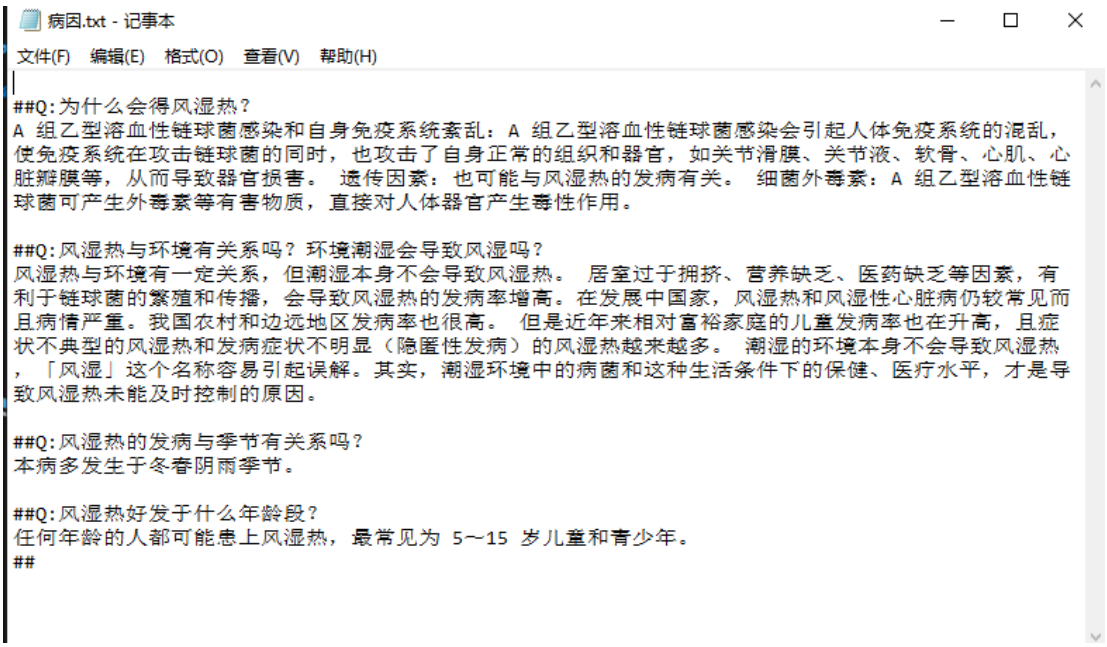
爬虫获得的数据集：


名称 修改日期 类型
儿科 2018/12/18 19:11 文件夹
耳鼻咽喉头颈外科 2018/12/18 20:02 文件夹
风湿免疫科 2018/12/18 20:28 文件夹
妇产科&生殖中心 2018/12/18 18:58 文件夹
肝胆胰腺外科 2018/12/18 20:03 文件夹
感染科&传染科 2018/12/18 19:25 文件夹
骨科 2018/12/18 19:38 文件夹
呼吸内科 2018/12/18 19:50 文件夹
甲状腺乳腺外科 2018/12/18 20:28 文件夹
精神心理科 2018/12/18 20:28 文件夹
口腔科 2018/12/18 20:17 文件夹
泌尿外科 2018/12/18 19:51 文件夹
内分泌科 2018/12/18 19:49 文件夹
内科 2018/12/18 19:36 文件夹
皮肤性病科 2018/12/18 18:34 文件夹
普通外科 2018/12/18 18:59 文件夹
其他 2018/12/19 0:25 文件夹
神经内科 2018/12/18 19:24 文件夹
神经外科 2018/12/18 20:27 文件夹
肾脏内科 2018/12/18 20:16 文件夹
消化内科 2018/12/18 18:46 文件夹
心胸外科 2018/12/18 20:04 文件夹
心血管内科 2018/12/18 19:12 文件夹
血液科 2018/12/18 20:15 文件夹
眼科 2018/12/18 20:15 文件夹


名称 修改日期 类型
Bartter综合征 2018/12/18 18:59 文件夹
Rett综合征 2018/12/18 18:59 文件夹
艾森门格综合征 2018/12/18 18:59 文件夹
苯丙酮尿症 2018/12/18 18:59 文件夹
蚕豆病 2018/12/18 19:00 文件夹
肠套叠 2018/12/18 19:00 文件夹
成骨不全 2018/12/18 19:00 文件夹
持续性胎儿循环综合征 2018/12/18 19:00 文件夹
抽动症 2018/12/18 19:00 文件夹
川崎病 2018/12/18 19:00 文件夹
唇裂 2018/12/18 19:00 文件夹
蛋白质能量营养不良 2018/12/18 19:00 文件夹
地中海贫血 2018/12/18 19:00 文件夹
动脉导管未闭 2018/12/18 19:00 文件夹
法洛四联症 2018/12/18 19:00 文件夹
房间隔缺损 2018/12/18 19:00 文件夹
风湿热 2018/12/18 19:00 文件夹
肝豆状核变性 2018/12/18 19:00 文件夹
戈谢病 2018/12/18 19:00 文件夹
佝偻病 2018/12/18 19:11 文件夹
骨软化症 2018/12/18 19:00 文件夹
畸胎瘤 2018/12/18 19:00 文件夹
急性气管支气管炎 2018/12/18 19:00 文件夹
急性支气管炎 2018/12/18 19:00 文件夹
假性脑膜炎 2018/12/18 19:00 文件夹


脑 › Documents (D:) › javaio › 儿科 › 蛋白质能量营养不良
搜索"蛋白...
名称 修改日期 类型
source.txt 2018/12/18 19:00 文本文档
病因.txt 2018/12/18 19:00 文本文档
生活.txt 2018/12/18 19:00 文本文档
预防.txt 2018/12/18 19:00 文本文档
诊断.txt 2018/12/18 19:00 文本文档
症状.txt 2018/12/18 19:00 文本文档
治疗.txt 2018/12/18 19:00 文本文档

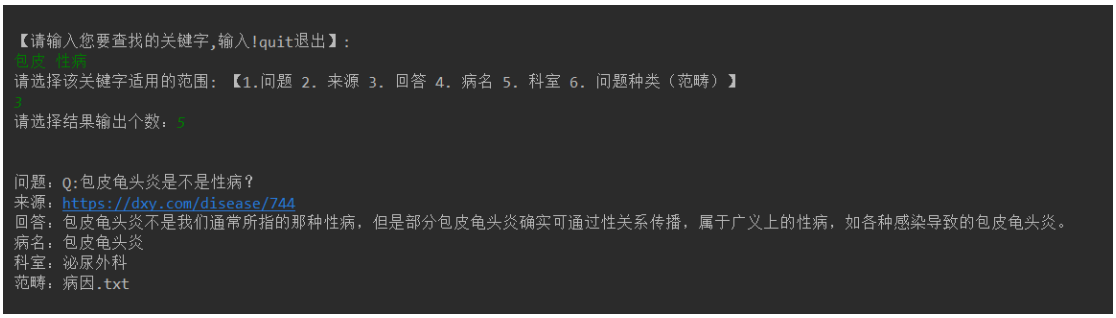




搜索模块命令行交互：

（单个关键字）

（多个关键字）



（切换查询字段）

```
【请输入您要查找的关键字,输入!quit退出】:
泌尿
请选择该关键字适用的范围: 【1.问题 2. 来源 3. 回答 4. 病名 5. 科室 6. 问题种类（范畴）】
5
请选择结果输出个数: 3


问题：Q:先天性睾丸发育不全综合征应该如何治疗？
来源：https://dxy.com/disease/5797
回答：内科治疗：主要以补充雄激素为主，口服、肌肉注射均可，越早诊断越早治疗，恢复第二性征等效果越好，但无助于生育。 外科治疗：以
病名：先天性睾丸发育不全综合征
科室：泌尿外科
范畴：治疗.txt


问题：Q:先天性睾丸发育不全综合征患者可能出现哪些合并症？
来源：https://dxy.com/disease/5797
回答：该病常伴有多系统疾病，如甲状腺功能异常、血糖异常，甚至糖尿病、二尖瓣脱垂、乳腺癌、唇裂、隐睾、自身免疫性疾病等。
病名：先天性睾丸发育不全综合征
科室：泌尿外科
范畴：治疗.txt


问题：Q:患该病的男性能正常孕育自己的亲生孩子吗？
来源：https://dxy.com/disease/5797
回答：以往这种疾病属于绝对不育的范围。随着医疗技术的进步，在一定程度上解决了这个问题。利用显微外科技术和人工辅助生殖技术，首先在
病名：先天性睾丸发育不全综合征
科室：泌尿外科
范畴：生活.txt
```

（程序退出）

```
【请输入您要查找的关键字,输入!quit退出】:
!quit
再见!


Process finished with exit code 0
```

5．总结

本次作业内容丰富，工作量较大，我系统学习了使用 JSoup 进行互联网爬虫和使用 Lucene 库进行索引建立和文本检索的相关知识，了解了反爬虫策略及其简单应对措施，复习了计算机网络关于应用层的相关论述，同时也复习了之前学习过的 html+javascript 以及 ajax 请求相关知识，巩固了 Java 应用技术课程中学习的基础知识，受益良多。

选择丁香园作为爬虫对象时，我就做好了打持久战的准备，只是没想到耗费如此多的时间。虽然付出很多，但看着整整齐齐的数据集和里面干货满满的内容，还是蛮有成就感的。

本次作业产品还有非常广阔的优化空间，在未来的学习和工作中我会充分吸取其中的经验和教训。

参考文献

[1] 耿祥义．Java 大学实用教程[M]．北京：清华大学出版社，2009．

[2] 耿祥义．Java 课程设计[M]．北京：清华大学出版社，2008．

[3] 丁振凡．Java 语言实验教程[M]．北京：北京邮电大学出版社，2005．

[4] 郑莉．Java 语言程序设计[M]．北京：清华大学出版社，2006．

[5] 丁香医生|可信赖的医疗健康信息和服务 https://dxy.com/disease

[6] java 爬虫教程：JSOUP https://xiaolongonly.cn/2016/05/06/Reptile1/

[7] JSoup Java HTML Parser, with best of DOM, CSS, and jquery https://JSoup.org/