Java应用技术实验报告：网页信息提取

姓名：应承峻

学号：3170103456

1 实验内容

- 读取网页，去除广告等无关部分，主要内容在屏幕显示或将其以txt文件方式存盘。
- 要求：不使用第三方工具，自己使用String方法或正则表达式作文本处理。
- 至少打包交 3 个文件：1)源程序，2)程序报告word/pdf，3)下载小说txt。

2 实验思路

本实验我们通过`WebReader`类完成整个功能。该类中的数据与几个主要方法如下：

私有数据域

```
private final String gurl;
private String text = new String();
```

gurl字符串存储了需要爬取的网页URL，而text字符串存储了爬取网页的源代码。

构造方法

```
public WebReader(String url) {
    this.gurl = url;
    readWebpage();
}
```

该构造方法只接受一个参数url，即网页的URL，存储在私有域中，随机调用`readWebPage`方法爬取网页。

爬取网页

```
private void readWebpage() {
    URL url = null;
    InputStream in = null;
    BufferedReader buffer = null;
    String line;
    try {
        url = new URL(this.gurl);   //构造URL对象
        in = url.openStream();  //打开输入流
        buffer = new BufferedReader(new InputStreamReader(in, "utf-8"));
        while ((line = buffer.readLine()) != null) {
            //Java自定义的换行符号，这种方法具有良好的跨平台性
            this.text = this.text + line + System.getProperty("line.separator");
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            if (in != null) in.close();  //关闭输入流
        } catch (IOException ioe) {
```

```
            ioe.printStackTrace();
        }
    }
}
```

爬取网页通过`URL`类构造一个`url`对象，然后每次通过读取一行的方式将源代码加载到`text`私有域中并换行。此外需要捕获可能出现的异常，如无法连接、缓冲区读取时出现的异常等等。最后需要关闭输入流，注意此时也需要捕获异常。

信息过滤

```
private String textFilter() {
    String content = new String();  //初始化
    String regex = "< *title *>.*< */ *title *>|< *p *>.*< *p *>";  //匹配<title>和<p>标签
    Pattern r = Pattern.compile(regex);
    Matcher m = r.matcher(this.text);
    while (m.find()) {
        //将匹配到的标签去除并换行
        content += m.group().replaceAll("< *title *>", "")
            .replaceAll("< */ *title *>|< *p *>",
System.getProperty("line.separator"));
    }
    return content;
}
```

在获取到源代码后，存盘前，我们需要通过`textFilter`这一私有方法对爬取的网页源代码进行过滤。在这里我们通过正则表达式的方式进行匹配，从爬取的源码来看，小说标题一般被`<title></title>`标签包裹着，而小说内容通常使用`<p></p>`标签进行包裹。但是奇怪的是，爬取下的网页源代码中，小说部分内容的`</p>`标签全部被替换成了`<p>`标签。因此我们通过`<p><p>`标签对来匹配。

综上，我们用于过滤的正则标签应该是`< *title *>.*< */ *title *>|< *p *>.*< *p *>`，其中`*`符号用于匹配标签名前后的空格，而`|`表示或运算。通过该正则表达式我们能够匹配到符合上述条件的文本。我们使用`Pattern`类和`Matcher`类进行正则匹配。注意，匹配到后的结果是包含标签的，因此我们需要通过`replaceAll`方法将标签替换掉。

文件存盘

```
public void storeWebpage(String filepath) {
    File file = new File(filepath);
    if (file.exists()) {
        System.out.println("File " + filepath
            + " has already exists, still write to this file ?(y for yes)");
        Scanner in = new Scanner(System.in);
        if (!in.nextLine().equals("y"))  return;
    }
    try {
        file.createNewFile();
        FileOutputStream out = new FileOutputStream(file);
        out.write(textFilter().getBytes());
        if (out != null) out.close();
    } catch (IOException ioe) {
        ioe.printStackTrace();
    }
}
```

我们使用`File`类进行文件的操作，`storeWebPage`方法接受一个`String`类型的参数表示小说存储的位置。在存储前，我们需要判断该路径上的文件是否存在，如果存在则需要再次询问是否确定覆盖文件，否则我们就在路径处创建该文件，通过`FileOutputStream`输出流的`write`方法将过滤后的文本输出到文件中。此处需要注意的是该方法只接受`byte []`即字节数组而不接受`String`对象，因此需要

先调用`getBytes()`方法将字节数组转换成字符串。

3 实验结果

执行`javac -encoding UTF-8 WebReader.java`命令后对文件进行编译后得到字节码文件，然后通过命令`java WebReader`运行程序，得到文本`source.txt`，具体爬取的内容见文本。

4 实验心得

通过本次实验，我对文件类(File)和输入输出流(InputStream/OutputStream)的主要功能和编程接口有了更加深入的理解，在实验的过程中，我遇到了两个主要的问题。一个是在爬取网页时，每次读取一行后需要加换行符`\n`，但是在输出字符串时，并没有体现换行符，上网查阅得到，可能是因为不同平台导致换行方式不同，因此我们采用了System.getProperty("line.separator")来获得换行符，调用这种方法获得的换行符具有良好的跨平台性，从而解决了这一问题。另一个问题是，在爬取网页后发现爬取到的是乱码，原因是在爬取时网页的编码格式和预定义的格式不符，因此需要先从网页的`<meta>`标签中得到网页的编码格式为`utf-8`，然后在`bufferReader`创建时指定编码格式：`buffer = new BufferedReader(new InputStreamReader(in, "utf-8"));`