浙江大学

本科实验报告

姓名：

学院：　生物医学工程与仪器科学学院

系：　生物医学工程系

专业：　生物医学工程

学号：

指导教师：　耿晨歌

2024 年 10 月 26 日

浙江大学实验报告

课程名称：高级程序设计　　实验类型：＿＿＿＿＿＿＿＿

实验项目名称：＿＿＿＿文本解析＿＿＿＿＿＿

同组学生姓名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

指导老师：耿晨歌

实验地点：＿＿＿＿＿＿＿＿＿＿实验日期：2024 年 10 月 26 日

一、实验目的和要求（必填）

本实验旨在练习文件输入/输出操作、正则表达式匹配、超链接生成与 HTML 格式化等编程技术。通过设计 Java 程序，完成对输入文本中的特定信息（电话号码、电子邮件地址、URL、IP 地址）的解析，并将其转换为带有超链接的 HTML 输出，强化对文本处理和正则表达式的掌握。最终生成的 HTML 文件应能正确识别并超链接各类信息，为构建解析工具提供实现参考。

二、实验内容和原理（必填）

在一段文本中查找 电话号码、email、url、ip 地址，并将原文转为带超链接的 html 输出，要求：

- 以命令行参数方式给文件名，打开文件，解析其中的文本
- 超链接的形式是 显示文本，除 url 外链接目标可自行设计
- 电话号码应支持国内固话号码和手机号码
- Url 应支持 http://, https://, ftp:// 以及无前缀的 url 网址，email 同理
- ip 地址支持 ipv4 地址，(可选，支持 ipv6 地址)
- 输出到一个 .html 后缀的新文件

三、主要仪器设备

开发环境：Visual Studio Code

编程语言：Java

测试工具：JUnit

四、操作方法与实验步骤

1、**创建 Maven 项目**：新建 Java 文件 TextParser.java，用于编写文本解析的主逻辑。

2、**编写正则表达式**：定义并测试 URL、Email、电话、IPv4 和 IPv6 地址的正则表达式，确保匹配各类输入格式。

3、**编写超链接生成逻辑**：通过 parseText 方法，使用占位符方式解析并替换匹配内容为超链接格式。

4、**文件操作**：实现文件的读取、写入操作，根据传入的命令行参数读取文件名，输出到同名 .html 文件中。

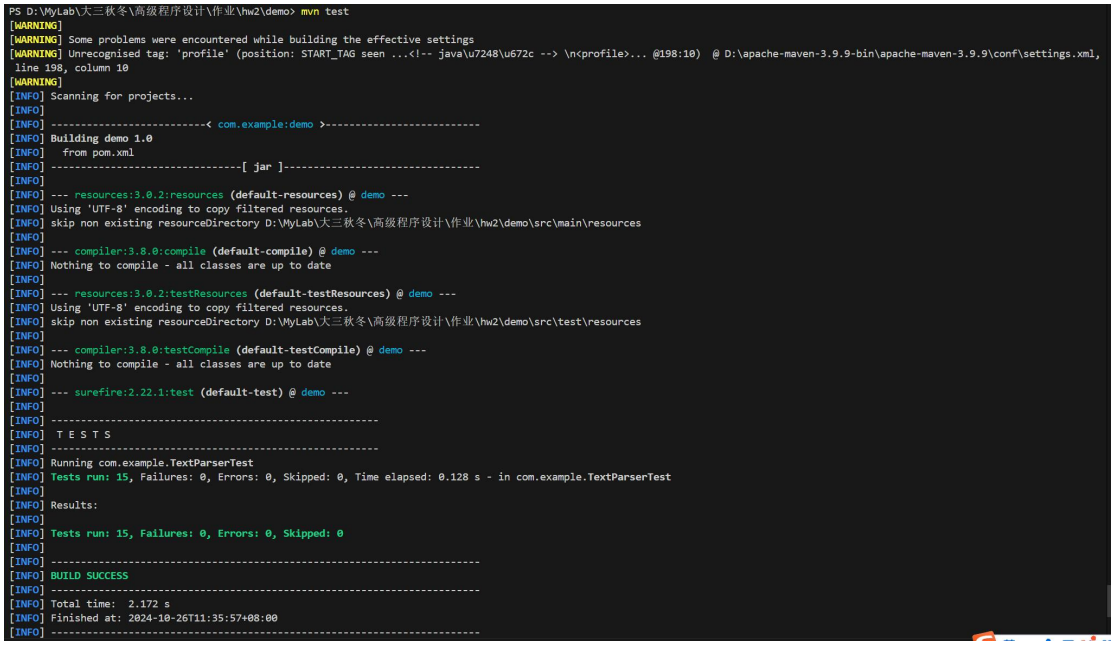
5、**测试代码**：使用 JUnit 单元测试验证解析效果，特别关注 URL 前缀、特殊字符、各种电话和 IP 格式。

6、**运行测试**：通过命令行运行程序并观察生成的 HTML 文件，检查所有信息的超链接效果。

五、实验数据记录和处理

html 测试结果：

Junit 测试结果：



六、实验结果与分析（必填）

1、最开始设计的匹配 URL 的正则表达式没有考虑到无前缀的 url

```
private static final String URL_REGEX = "(https?|ftp)://[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,6}";
```

它能匹配 http://www.example.com，但却无法匹配 www.example.com，因此改进时加入了可选的分组：

```
    private static final String URL_REGEX = "((https?|ftp)://|www\\.)?[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,6}";
```

2、最开始没有考虑到 URL 可能包含路径、查询参数、片段标识符等部分。但是抬头查看学在浙大作业信息的时候突然意识到，这部分也可以加入来加强代码的鲁棒性。

3、原先是直接在 input.txt 上加上.html 变成 input.txt.html，后面新定义了 removeFileExtension 的方法去掉了输入文件名的原有后缀，然后再加上 .html 作为输出文件的后缀。

4、实验要求中“电话号码应支持国内固话号码和手机号码”的情况其实可以分为以下几种：

（1） 国内固话号码（座机号码）

- **带区号的固话号码**：通常区号由 3 到 4 位数字组成，电话号码为 7 到 8 位。

格式示例：010-12345678（北京），021-87654321（上海），0755-1234567（深圳）

·不带区号的固话号码：仅限于本地拨打，通常为 7 到 8 位。

格式示例：12345678，1234567

（2） 国内手机号码

·标准格式：中国大陆手机号码一般为 11 位数字，以 1 开头，第二位为 3-9 之间的数字。

格式示例：13800138000，15212345678

·带国际区号：在国际格式中，通常以 +86（或 0086）开头，后跟 11 位手机号。

格式示例：+8613800138000，008613800138000

因此需要使用((\\\\+?86)?1[3-9]\\\\d{9})|((\\\\d{3,4}-)?\\\\d{7,8})的正则。

5、在 parseIP 方法中，代码首先用 IPv4 正则进行匹配，替换完后再用 IPv6 进行匹配替换。这样可能会导致部分地址的混淆或误替换；所以我选择先大后小，先将原始文本完全替换 IPv6 地址，再进行 IPv4 的替换，以确保两类地址不互相干扰。

6、先解析URL再解析Email就会出现：

Email: admin@example.com

先解析Email再解析URL则会出现：

Email: admin@example.com

修改方式尝试了很多，比如负向前瞻和负向后顾，又或是Jsoup库；但最后有效的改动还是重构代码逻辑，使用replaceWithPlaceholders方法替换文本中的匹配项为占位符（如__URL__0__），并将原始值存储在对应的列表中；随后通过restorePlaceholders方法遍历每种占位符列表，将其替换为最终的HTML超链接。

七、讨论、心得

在本次实验中，通过实现对多种文本类型的正则匹配和超链接转换，加深了对正则表达式的理解。遇到的挑战在于匹配多种 URL 格式和支持复杂的 IP 地址结构，特别是 IPv6。在解决此问题的过程中，我学会了通过正则分组和替换占位符避免嵌套替换的冲突。尽管正则表达式的逻辑较为复杂，但通过不断测试和调整，最终得到了较为通用的解决方案。

该实验还让我意识到，在处理文本替换时，代码结构和逻辑的清晰性至关重要。为了避免重复匹配和替换出错，将不同类型的内容分层解析，并采用先大后小的匹配策略，提升了代码的可维护性和扩展性。

八、附件（代码）

```
package com.example;


import java.io.*;
import java.util.ArrayList;
import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
public class TextParser {
    private static final String URL_REGEX =
"((https?|ftp)://)?(www\\.)?[a-zA-Z0-9.-]+(\\.[a-zA-Z]{2,6})([-a-zA-Z0-9:%_+.~#?&//=]*)";
    private static final String IPv6_REGEX =
"((([0-9A-Fa-f]{1,4}:){7}([0-9A-Fa-f]{1,4}|:))|(([0-9A-Fa-f]{1,4}:){1,7}:)|" +
        "(([0-9A-Fa-f]{1,4}:){1,6}:[0-9A-Fa-f]{1,4})|(([0-9A-Fa-f]{1,4}:){1,5}(:[0-9A-Fa-f]{1,4}){1,2})|" +
        "(([0-9A-Fa-f]{1,4}:){1,4}(:[0-9A-Fa-f]{1,4}){1,3})|(([0-9A-Fa-f]{1,4}:){1,3}(:[0-9A-Fa-f]{1,4}){1,4}
)|" +
        "(([0-9A-Fa-f]{1,4}:){1,2}(:[0-9A-Fa-f]{1,4}){1,5})|([0-9A-Fa-f]{1,4}:((:[0-9A-Fa-f]{1,4}){1,6}))|"
+
        "(:((:[0-9A-Fa-f]{1,4}){1,7}|:)))(%.+)?";
    private static final String EMAIL_REGEX = "[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,6}";
    private static final String PHONE_REGEX = "(\\+?86)?\\(?\\d{3,4}\\)?[-\\s]?\\d{7,8}|\\d{3}[-\\s]?\\d{4}";
    private static final String IPv4_REGEX = "((25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\.){3}" +
        "(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)(:\\d{1,5})?";
    // 占位符列表
    private static List<String> urlPlaceholders = new ArrayList<>();
    private static List<String> emailPlaceholders = new ArrayList<>();
```

```
    private static List<String> phonePlaceholders = new ArrayList<>();
    private static List<String> ipv4Placeholders = new ArrayList<>();
    private static List<String> ipv6Placeholders = new ArrayList<>();
    public static void main(String[] args) {
        if (args.length != 1) {
            System.out.println("请提供文件名！");
            return;
        }
        String input = args[0];
        String output = removeFileExtension(input) + ".html";
        try (BufferedReader reader = new BufferedReader(new FileReader(input));
             BufferedWriter writer = new BufferedWriter(new FileWriter(output))) {
            String line;
            while ((line = reader.readLine()) != null) {
                line = parseText(line);
                writer.write(line);
                writer.newLine();
            }
        } catch (IOException e) {
            System.out.println("读取或写入文件时出错！");
            e.printStackTrace();
        }
    }
    // 处理文本中的所有内容
    public static String parseText(String text) {
        text = replaceWithPlaceholders(text, EMAIL_REGEX, emailPlaceholders, "__EMAIL__");
        text = replaceWithPlaceholders(text, URL_REGEX, urlPlaceholders, "__URL__");
        text = replaceWithPlaceholders(text, PHONE_REGEX, phonePlaceholders, "__PHONE__");
        text = replaceWithPlaceholders(text, IPv4_REGEX, ipv4Placeholders, "__IPv4__");
        text = replaceWithPlaceholders(text, IPv6_REGEX, ipv6Placeholders, "__IPv6__");

        return restorePlaceholders(text);
    }
    // 使用占位符替换文本中的匹配项
    private static String replaceWithPlaceholders(String text, String regex, List<String> placeholders, String
placeholderPrefix) {
        Pattern pattern = Pattern.compile(regex);
        Matcher matcher = pattern.matcher(text);
        int index = placeholders.size();
        StringBuffer sb = new StringBuffer();
        while (matcher.find()) {
            String match = matcher.group();
            placeholders.add(match);
            matcher.appendReplacement(sb, placeholderPrefix + index + "__");
```

```
            index++;
        }
        matcher.appendTail(sb);
        return sb.toString();
    }
    // 还原占位符为超链接
    private static String restorePlaceholders(String text) {
        for (int i = 0; i < urlPlaceholders.size(); i++) {
            String url = urlPlaceholders.get(i);
            if (!url.startsWith("http")) {
                url = "http://" + url;
            }
            String replacement = "<a href=\"" + url + "\">" + url + "</a>";
            text = text.replace("__URL__" + i + "__", replacement);
        }
        for (int i = 0; i < emailPlaceholders.size(); i++) {
            String email = emailPlaceholders.get(i);
            String replacement = "<a href=\"mailto:" + email + "\">" + email + "</a>";
            text = text.replace("__EMAIL__" + i + "__", replacement);
        }
        for (int i = 0; i < phonePlaceholders.size(); i++) {
            String phone = phonePlaceholders.get(i);
            String replacement = "<a href=\"tel:" + phone + "\">" + phone + "</a>";
            text = text.replace("__PHONE__" + i + "__", replacement);
        }
        for (int i = 0; i < ipv4Placeholders.size(); i++) {
            String ip = ipv4Placeholders.get(i);
            String replacement = "<a href=\"http://" + ip + "\">" + ip + "</a>";
            text = text.replace("__IPv4__" + i + "__", replacement);
        }
        for (int i = 0; i < ipv6Placeholders.size(); i++) {
            String ip = ipv6Placeholders.get(i);
            String replacement = "<a href=\"http://[" + ip + "]\">" + ip + "</a>";
            text = text.replace("__IPv6__" + i + "__", replacement);
        }
        return text;
    }
    // 去掉文件名后缀
    public static String removeFileExtension(String filename) {
        int dotIndex = filename.lastIndexOf(".");
        if (dotIndex == -1) {
            return filename;
        }
        return filename.substring(0, dotIndex);
```

```
    }
}
```

```
package com.example;

import org.junit.Test;
import static org.junit.Assert.assertEquals;
public class TextParserTest {
    @Test
    public void testParseTextWithURL() {
        String input = "Visit http://example.com or www.example.com";
        String expected = "Visit <a href=\"http://example.com\">http://example.com</a> or <a
href=\"http://www.example.com\">http://www.example.com</a>";
        assertEquals(expected, TextParser.parseText(input));
    }
    @Test
    public void testParseTextWithEmail() {
        String input = "Contact us at test@example.com";
        String expected = "Contact us at <a href=\"mailto:test@example.com\">test@example.com</a>";
        assertEquals(expected, TextParser.parseText(input));
    }
    @Test
    public void testParseTextWithPhone() {
        String input = "Call us at +8613800138000 or 010-12345678";
        String expected = "Call us at <a href=\"tel:+8613800138000\">+8613800138000</a> or <a
href=\"tel:010-12345678\">010-12345678</a>";
        assertEquals(expected, TextParser.parseText(input));
    }
    @Test
    public void testParseTextWithIP() {
        String input = "Visit 192.168.1.1 or 2001:0db8:85a3:0000:0000:8a2e:0370:7334";
        String expected = "Visit <a href=\"http://192.168.1.1\">192.168.1.1</a> or <a
href=\"http://[2001:0db8:85a3:0000:0000:8a2e:0370:7334]\">2001:0db8:85a3:0000:0000:8a2e:0370:7334</a>";
        assertEquals(expected, TextParser.parseText(input));
    }
    @Test
    public void testRemoveFileExtension() {
        assertEquals("example", TextParser.removeFileExtension("example.txt"));
        assertEquals("example", TextParser.removeFileExtension("example."));
        assertEquals("example", TextParser.removeFileExtension("example"));
    }
    @Test
    public void testParseTextWithComplexURL() {
```

```
        String input = "Visit https://subdomain.example.com/page?query=test";
        String expected = "Visit <a
href=\"https://subdomain.example.com/page?query=test\">https://subdomain.example.com/page?query=test</a>";
        assertEquals(expected, TextParser.parseText(input));
    }
    @Test
    public void testParseTextWithURLWithoutProtocol() {
        String input = "Visit www.example.com/path";
        String expected = "Visit <a href=\"http://www.example.com/path\">http://www.example.com/path</a>";
        assertEquals(expected, TextParser.parseText(input));
    }
    @Test
    public void testParseTextWithURLAndPort() {
        String input = "Visit http://example.com:8080";
        String expected = "Visit <a href=\"http://example.com:8080\">http://example.com:8080</a>";
        assertEquals(expected, TextParser.parseText(input));
    }
    @Test
    public void testParseTextWithEmailAndSpecialCharacters() {
        String input = "Contact us at user+name@example.com";
        String expected = "Contact us at <a href=\"mailto:user+name@example.com\">user+name@example.com</a>";
        assertEquals(expected, TextParser.parseText(input));
    }
    @Test
    public void testParseTextWithMultipleEmails() {
        String input = "Send to admin@example.com, support@example.org";
        String expected = "Send to <a href=\"mailto:admin@example.com\">admin@example.com</a>, <a
href=\"mailto:support@example.org\">support@example.org</a>";
        assertEquals(expected, TextParser.parseText(input));
    }
    @Test
    public void testParseTextWithPhoneAndDifferentFormats() {
        String input = "Numbers: (010)12345678, 123-4567";
        String expected = "Numbers: <a href=\"tel:(010)12345678\">(010)12345678</a>, <a
href=\"tel:123-4567\">123-4567</a>";
        assertEquals(expected, TextParser.parseText(input));
    }
    @Test
    public void testParseTextWithInvalidIP() {
        String input = "Visit 999.999.999.999";
        String expected = "Visit 999.999.999.999";  // 不应被解析为 IP 链接
        assertEquals(expected, TextParser.parseText(input));
    }
    @Test
```

```
    public void testParseTextWithIPv6AbbreviatedNotation() {
        String input = "Visit ::1 for local testing";
        String expected = "Visit <a href=\"http://[::1]\">::1</a> for local testing";
        assertEquals(expected, TextParser.parseText(input));
    }
    @Test
    public void testMixedContentParsing() {
        String input = "Website: http://example.com, Email: admin@zju.edu.cn, Phone: 010-12345678, IP:
192.168.0.1";
        String expected = "Website: <a href=\"http://example.com\">http://example.com</a>, Email: <a
href=\"mailto:admin@zju.edu.cn\">admin@zju.edu.cn</a>, Phone: <a href=\"tel:010-12345678\">010-12345678</a>,
IP: <a href=\"http://192.168.0.1\">192.168.0.1</a>";
        assertEquals(expected, TextParser.parseText(input));
    }
    @Test
    public void testParseTextWithEmptyString() {
        String input = "";
        String expected = "";
        assertEquals(expected, TextParser.parseText(input));
    }
}
```