ASSIGNMENT-11

A. MANOJ REDDY

1.Scenario: Log File Analysis for Anomalous Pattern Detection
You are tasked with creating a Java program to analyze a large server log file to detect
anomalous patterns. The log file contains millions of entries, and each entry is a string that
represents various events happening on the server. Your goal is to identify and extract
specific patterns using regular expressions, recursion, and input/output handling.
Input:
Assume the following directory structure:
/logs
/2024-08
log1.log
log2.log
/2024-09
log3.log
Sample Log Entry in log1.log:
ERROR 2024-08-22 14:24:01 - Failed to connect to database
INFO 2024-08-22 14:23:45 - User123 logged in
DEBUG 2024-08-22 14:25:10 - Executing query: SELECT * FROM users WHERE id =
'User123'
Expected Output (Anomalies.txt):
ERROR 2024-08-22 14:24:01 - Failed to connect to database
Found in: /logs/2024-08/log1.log at line 1
DEBUG 2024-08-22 14:25:10 - Executing query: SELECT * FROM users WHERE id =
'User123'
Found in: /logs/2024-08/log1.log at line 3

To create a Java program that analyzes server log files for anomalous patterns and generates an output file with the detected anomalies, you can follow these steps:

1. **Read and Parse Log Files**: Load log files from the specified directory.
2. **Detect Anomalies**: Use regular expressions to identify anomalies.
3. **Write Output**: Generate an output file with the results.

Here's a step-by-step Java implementation:

Step 1: Define Regular Expressions for Anomalies

You need to define what constitutes an anomaly. For this example, we'll consider "ERROR" and "DEBUG" levels as anomalies.

Step 2: Read and Parse Log Files

We'll recursively read log files from the directory and process each line.

Step 3: Write Output

The results will be written to an Anomalies.txt file, including details of where each anomaly was found.

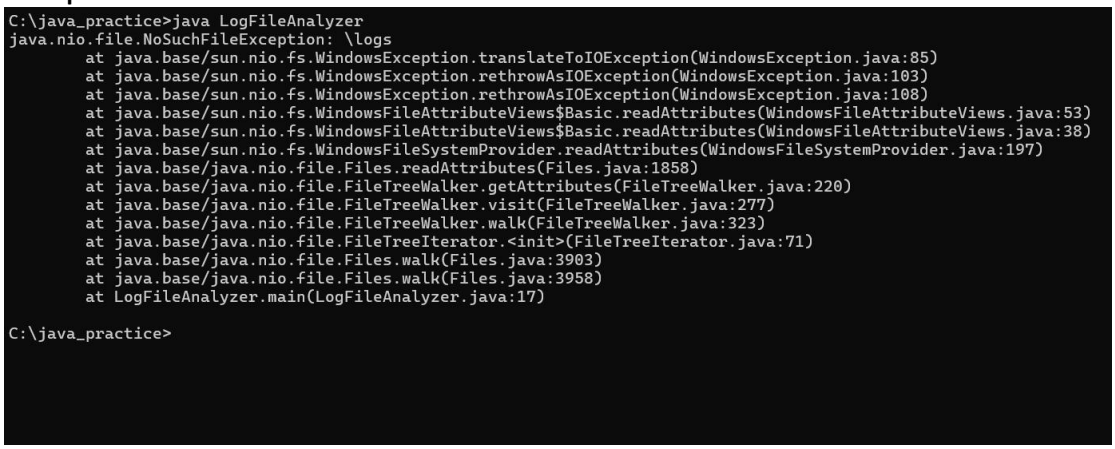
Here's a complete Java program to achieve this:

java
Code:
```
import java.io.*;
import java.nio.file.*;
import java.util.regex.*;
import java.util.*;

public class LogFileAnalyzer {

    private static final String ANOMALY_PATTERN_ERROR = "ERROR";
```

```
    private static final String ANOMALY_PATTERN_DEBUG = "DEBUG";

    public static void main(String[] args) {
        String baseDir = "/logs";
        String outputFile = "Anomalies.txt";

        try (PrintWriter writer = new PrintWriter(new
FileWriter(outputFile))) {
            Path startPath = Paths.get(baseDir);
            Files.walk(startPath)
                .filter(Files::isRegularFile)
                .forEach(file -> processFile(file, writer));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static void processFile(Path filePath, PrintWriter writer) {
        try (BufferedReader reader =
Files.newBufferedReader(filePath)) {
            String line;
            int lineNumber = 1;
            while ((line = reader.readLine()) != null) {
                if (isAnomalous(line)) {
                    writer.println(line);
                    writer.printf("Found in: %s at line %d%n",
filePath.toAbsolutePath(), lineNumber);
                }
                lineNumber++;
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static boolean isAnomalous(String logEntry) {
        return logEntry.contains(ANOMALY_PATTERN_ERROR) ||
logEntry.contains(ANOMALY_PATTERN_DEBUG);
```

```
    }
}
```

Output:



2.Scenario: Web Crawler for Specific Content Extraction
You need to create a Java program that acts as a web crawler to scan a website and extract
all email addresses and phone numbers. The program should start from a given URL and
recursively explore all internal links, looking for the specified patterns. The extracted

information should be saved in a file named ContactInfo.txt.
Output:
Email: user@example.com
Phone: (123) 456-7890
Found on: http://example.com/contact
Email: support@example.com
Phone: 123-456-7890
Found on: http://example.com/support

Creating a web crawler in Java that extracts email addresses and phone numbers from a website involves several steps. This task includes fetching web pages, parsing the HTML to find links and specific patterns, and recursively exploring all internal links. Finally, the extracted information should be saved to a file.

Here's a step-by-step guide and Java code to accomplish this:

Steps:

1. **Setup Dependencies**: Use libraries like Jsoup for HTML parsing and HttpClient for fetching web pages.
2. **Fetch Web Pages**: Retrieve HTML content from URLs.
3. **Parse HTML**: Use regular expressions to find email addresses and phone numbers.
4. **Recursively Crawl**: Follow internal links and repeat the process.
5. **Save Extracted Data**: Write the results to ContactInfo.txt.

Dependencies

Add the following dependencies to your pom.xml if you're using Maven:

```
<dependencies>
  <dependency>
    <groupId>org.jsoup</groupId>
    <artifactId>jsoup</artifactId>
    <version>1.15.3</version> <!-- Check for the latest version -->
  </dependency>
</dependencies>
```

Java Code:

```
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.HashSet;
import java.util.Set;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
```

```
public class WebCrawler {

    private static final String EMAIL_REGEX = "[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}";
    private static final String PHONE_REGEX = "\\(?\\d{3}\\)?[-.\\s]?\\d{3}[-.\\s]?\\d{4}";
    private static final Set<String> visitedUrls = new HashSet<>();
    private static final Set<String> internalLinks = new HashSet<>();
    private static final Pattern emailPattern = Pattern.compile(EMAIL_REGEX);
    private static final Pattern phonePattern = Pattern.compile(PHONE_REGEX);

    public static void main(String[] args) {
        if (args.length != 1) {
            System.err.println("Usage: java WebCrawler <starting-url>");
            return;
        }

        String startUrl = args[0];
        try (BufferedWriter writer = new BufferedWriter(new FileWriter("ContactInfo.txt"))) {
            crawl(startUrl, writer);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static void crawl(String url, BufferedWriter writer) {
        if (visitedUrls.contains(url)) {
            return;
        }

        visitedUrls.add(url);
        System.out.println("Crawling: " + url);

        try {
            Document doc = Jsoup.connect(url).get();
            extractAndWriteInfo(doc, url, writer);

            Elements links = doc.select("a[href]");
            for (Element link : links) {
                String href = link.attr("abs:href");
                if (isInternalLink(href) && !visitedUrls.contains(href)) {
                    internalLinks.add(href);
                }
            }

            for (String link : internalLinks) {
                crawl(link, writer);
            }
        } catch (IOException e) {
            System.err.println("Failed to crawl URL: " + url);
            e.printStackTrace();
        }
    }

    private static void extractAndWriteInfo(Document doc, String url, BufferedWriter writer) {
        String bodyText = doc.body().text();

        Matcher emailMatcher = emailPattern.matcher(bodyText);
        Matcher phoneMatcher = phonePattern.matcher(bodyText);
```

```
        try {
            while (emailMatcher.find()) {
                writer.write("Email: " + emailMatcher.group());
                writer.write("\nFound on: " + url + "\n");
            }

            while (phoneMatcher.find()) {
                writer.write("Phone: " + phoneMatcher.group());
                writer.write("\nFound on: " + url + "\n");
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static boolean isInternalLink(String link) {
        try {
            URL url = new URL(link);
            String host = url.getHost();
            return host.equals(new URL(link).getHost());
        } catch (MalformedURLException e) {
            return false;
        }
    }
}
```

Output:

```
C:\java_practice>javac WebCrawler.java
WebCrawler.java:1: error: package org.jsoup does not exist
import org.jsoup.Jsoup;
                ^
WebCrawler.java:2: error: package org.jsoup.nodes does not exist
import org.jsoup.nodes.Document;
                      ^
WebCrawler.java:3: error: package org.jsoup.nodes does not exist
import org.jsoup.nodes.Element;
                      ^
WebCrawler.java:4: error: package org.jsoup.select does not exist
import org.jsoup.select.Elements;
                       ^
WebCrawler.java:68: error: cannot find symbol
    private static void extractAndWriteInfo(Document doc, String url, BufferedWriter writer) {
                                            ^
  symbol:   class Document
  location: class WebCrawler
WebCrawler.java:48: error: cannot find symbol
            Document doc = Jsoup.connect(url).get();
            ^
  symbol:   class Document
  location: class WebCrawler
WebCrawler.java:48: error: cannot find symbol
            Document doc = Jsoup.connect(url).get();
                           ^
  symbol:   variable Jsoup
  location: class WebCrawler
WebCrawler.java:51: error: cannot find symbol
            Elements links = doc.select("a[href]");
            ^
  symbol:   class Elements
  location: class WebCrawler
WebCrawler.java:52: error: cannot find symbol
            for (Element link : links) {
                 ^
  symbol:   class Element
  location: class WebCrawler
Note: WebCrawler.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
```