CODE:

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

public class email {
   public static void main(String[] args) {
      // File path of the uploaded .eml file
      String filePath = "email_1.eml"; // Replace with your actual file name
      try {
         // Read the file content
         String emailContent = readFile(filePath);
         // Extract and display header information
         parseEmailHeader(emailContent);
         // Check for attachments
         checkForAttachments(emailContent);
      } catch (IOException e) {
         System.out.println("Error reading the file: " + e.getMessage());
      }
   }

   private static String readFile(String filePath) throws IOException {
      StringBuilder contentBuilder
= new StringBuilder();
      try (BufferedReader br = new BufferedReader(new FileReader(filePath))) {
         String currentLine;
         while ((currentLine = br.readLine()) != null) {
            contentBuilder.append(currentLine).append("\n");

         }
      }
      return contentBuilder.toString();
   }

   private static void parseEmailHeader(String emailContent) {
```

```
    // Patterns to match specific header fields
    String fromPattern = "From: (.+)";
    String toPattern = "To: (.+)";
    String datePattern = "Date: (.+)";
    String receivedPattern = "Received: (.+)";
    String mimeVersionPattern = "MIME-Version: (.+)";
    String messageIdPattern = "Message-ID: (.+)";
    String dkimSignaturePattern = "DKIM-Signature: (.+)";
    // Display results
    System.out.println("Extracted Email Header Information:");
    extractAndPrint(emailContent, fromPattern, "From");
    extractAndPrint(emailContent, toPattern, "To");
    extractAndPrint(emailContent, datePattern, "Date");
    extractAndPrint(emailContent, receivedPattern, "Received");
    extractAndPrint(emailContent, mimeVersionPattern, "MIME-Version");
    extractAndPrint(emailContent, messageIdPattern, "Message-ID");
    extractAndPrint(emailContent, dkimSignaturePattern, "DKIM-Signature");
}

private static void extractAndPrint(String emailContent, String patternString, String label) {
    Pattern pattern = Pattern.compile(patternString, Pattern.MULTILINE);
    Matcher matcher = pattern.matcher(emailContent);
    while (matcher.find()) {
        System.out.println(label + ": " + matcher.group(1));
    }
}

private static void checkForAttachments(String emailContent) {
    // Pattern to detect boundaries and content disposition
    String boundaryPatternString = "boundary=\"([^\"]+)\"";
    String contentDispositionPatternString = "Content-Disposition: attachment; filename=\"([^\"]+)\"";
    String contentTypePatternString = "Content-Type: ([^;]+);.*";
    Pattern boundaryPattern = Pattern.compile(boundaryPatternString, Pattern.MULTILINE);
    Matcher boundaryMatcher = boundaryPattern.matcher(emailContent);
    if (boundaryMatcher.find()) {
```

```
            String boundary = boundaryMatcher.group(1);
            // Split the email content into parts based on the boundary
            String[] parts = emailContent.split("--" + boundary);
            for (String part : parts) {
                Matcher dispositionMatcher = Pattern.compile(contentDispositionPatternString,
Pattern.MULTILINE).matcher(part);
                Matcher contentTypeMatcher = Pattern.compile(contentTypePatternString, Pattern.MULTILINE).matcher(part);
                if (dispositionMatcher.find() && contentTypeMatcher.find()) {
                    String fileName = dispositionMatcher.group(1);
                    String contentType = contentTypeMatcher.group(1);
                    System.out.println("\nAttachment Detected:");
                    System.out.println("File Name: " + fileName);
                    System.out.println("File Type: " + contentType);
                    String requiredSoftware = determineSoftware(contentType);
                    System.out.println("Required Software: " + requiredSoftware);
                }
            }
        } else {
            System.out.println("\nNo attachments found.");
        }
    }

    private static String determineSoftware(String contentType) {
        // Map common content types to required software
        Map<String, String> contentTypeToSoftware = new HashMap<>();
        contentTypeToSoftware.put("application/pdf", "Adobe Acrobat Reader or any PDF viewer");
        contentTypeToSoftware.put("application/msword", "Microsoft Word or compatible word processor");
        contentTypeToSoftware.put("application/vnd.ms-excel", "Microsoft Excel or compatible spreadsheet software");
        contentTypeToSoftware.put("image/jpeg", "Image viewer or photo editor");
        contentTypeToSoftware.put("image/png", "Image viewer or photo editor");
        contentTypeToSoftware.put("application/zip", "WinRAR, 7-Zip, or any archive extraction tool");
        return contentTypeToSoftware.getOrDefault(contentType, "Unknown software, please check the file type");
    }
}
```

OUTPUT:

PS C:\Users\MEGHA> cd "c:\Users\MEGHA\OneDrive\Desktop\college work\BE\csdf\" ; if ($?) { javac email.java } ; if ($?) { java email }

Extracted Email Header Information:

From: "Coursera" <no-reply@t.mail.coursera.org>

To: meghakuhu@gmail.com

To: meghakuhu@gmail.com

Date: Thu, 08 Aug 2024 14:37:17 +0000

Received: by 2002:ab3:303:0:b0:273:d70d:bc4b with SMTP id n3csp915688ltg;

Received: by 2002:a05:6359:4109:b0:1ac:660a:8a10 with SMTP id e5c5f4694b2df-1b15cf6613cmr203954555d.3.1723127838632;

Received: from mta-83-13.sparkpostmail.com (mta-83-13.sparkpostmail.com. [192.174.83.13])

MIME-Version: 1.0

Message-ID: <C0.58.33133.D18D4B66@hi.mta2vrest.cc.prd.sparkpost>

DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed; d=t.mail.coursera.org;

PS C:\Users\MEGHA\OneDrive\Desktop\college work\BE\csdf>