DEPARTMENT OF COMPUTER SCIENCE AND APPLICATIONS
PANJAB UNIVERSITY

Java Practical Assignment – File Handling

Submitted To :

Asst. Professor Jasleen Kaur

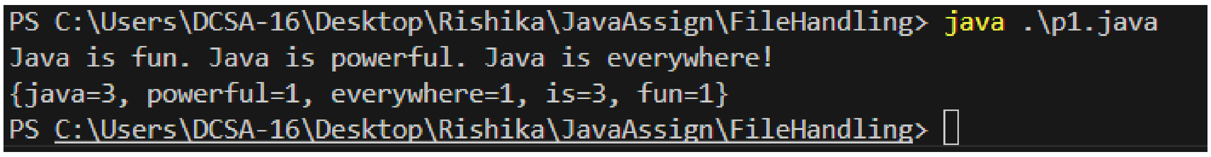
Submitted By :

Rishika Gautam (88)

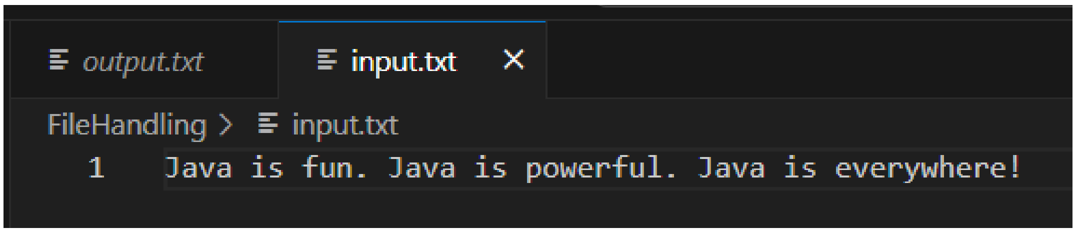
MCA II (Evening)

1. Create a Java program that Reads a text file (input.txt) using FileInputStream and Scanner. It should count the frequency of each word (ignore case and punctuation). Outputs the word frequencies to another file (word_count.txt) using FileOutputStream.

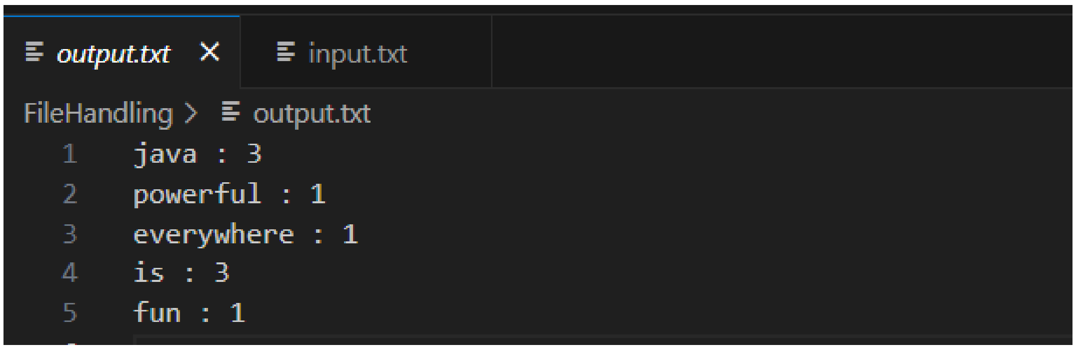
```java
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.util.HashMap;
import java.util.Scanner;

public class p1 {
    public static void main(String[] args) throws IOException {

        Scanner sc = new Scanner(System.in);
        String text = sc.nextLine();

        try (FileWriter writer = new FileWriter("input.txt")) {

            writer.write(text);

        }catch(IOException e){
            System.out.println("Error"+e.getMessage());
        }

        FileInputStream fileS = new FileInputStream("input.txt");
        Scanner fileSc = new Scanner(fileS);
        HashMap<String, Integer> hash = new HashMap<>();

        while (fileSc.hasNext()) {
            String word = fileSc.next().replaceAll("[^a-zA-
z]","").toLowerCase();

            if(!word.isEmpty()){
                hash.put(word, hash.getOrDefault(word,0)+1);
            }
        }

        System.out.println(hash);

        FileWriter output = new FileWriter("output.txt");

        for(String word : hash.keySet()){
            output.write(word+" : "+ hash.get(word)+"\n");
        }

        output.close();
        fileS.close();
        sc.close();
        fileSc.close();
    }
}
```



```
PS C:\Users\DCSA-16\Desktop\Rishika\JavaAssign\FileHandling> java .\p1.java
Java is fun. Java is powerful. Java is everywhere!
{java=3, powerful=1, everywhere=1, is=3, fun=1}
PS C:\Users\DCSA-16\Desktop\Rishika\JavaAssign\FileHandling> 
```



```
Java is fun. Java is powerful. Java is everywhere!
```

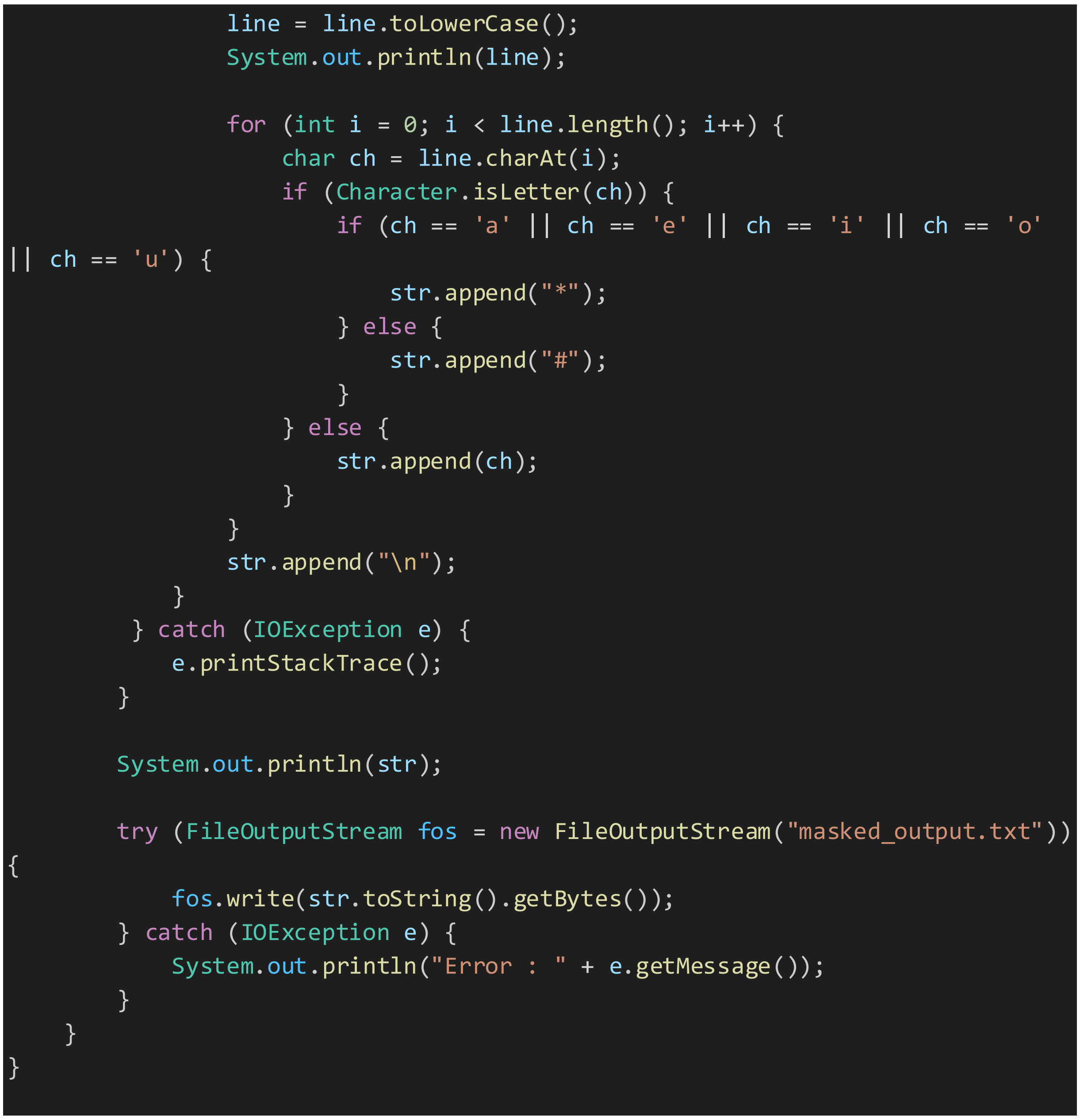


```
java : 3
powerful : 1
everywhere : 1
is : 3
fun : 1
```

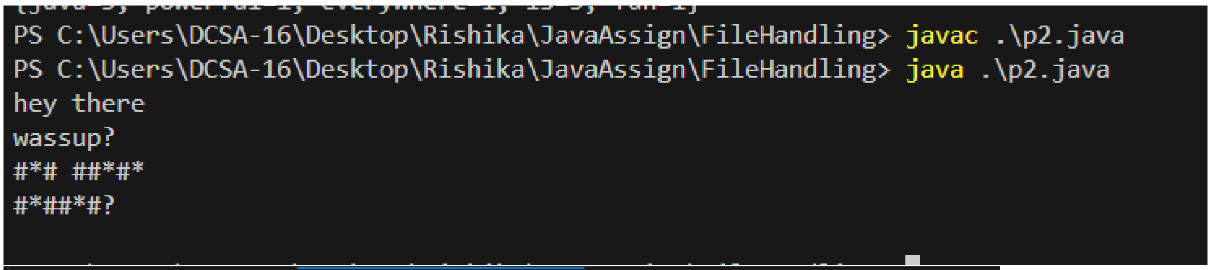
2. Write a Java program that:

- **Reads the contents of a file (input2.txt) using FileInputStream and Scanner.**
- **Replaces every vowel in each word with * and every consonant with #.**
- **Writes the modified content to a new file (masked_output.txt) using FileOutputStream.**

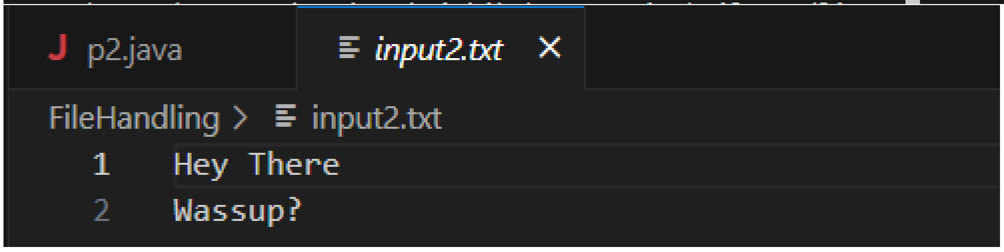
```
import java.io.*;
import java.util.Scanner;

public class p2 {
    public static void main(String[] args) {

        StringBuilder str = new StringBuilder();
        try (
            FileInputStream fileS = new FileInputStream("input2.txt");
            Scanner sc = new Scanner(fileS);
         ){
            while(sc.hasNextLine()){
                String line = sc.nextLine();
```



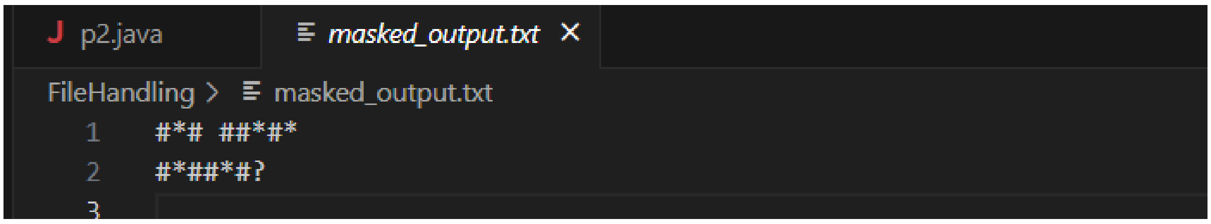
```java
                line = line.toLowerCase();
                System.out.println(line);

                for (int i = 0; i < line.length(); i++) {
                    char ch = line.charAt(i);
                    if (Character.isLetter(ch)) {
                        if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o'
|| ch == 'u') {
                            str.append("*");
                        } else {
                            str.append("#");
                        }
                    } else {
                        str.append(ch);
                    }
                }
                str.append("\n");
            }
         } catch (IOException e) {
            e.printStackTrace();
        }

        System.out.println(str);

        try (FileOutputStream fos = new FileOutputStream("masked_output.txt"))
{
            fos.write(str.toString().getBytes());
        } catch (IOException e) {
            System.out.println("Error : " + e.getMessage());
        }
    }
}
```



```
PS C:\Users\DCSA-16\Desktop\Rishika\JavaAssign\FileHandling> javac .\p2.java
PS C:\Users\DCSA-16\Desktop\Rishika\JavaAssign\FileHandling> java .\p2.java
hey there
wassup?
#*# ##*#*
#*##*#?
```



```
Hey There
Wassup?
```

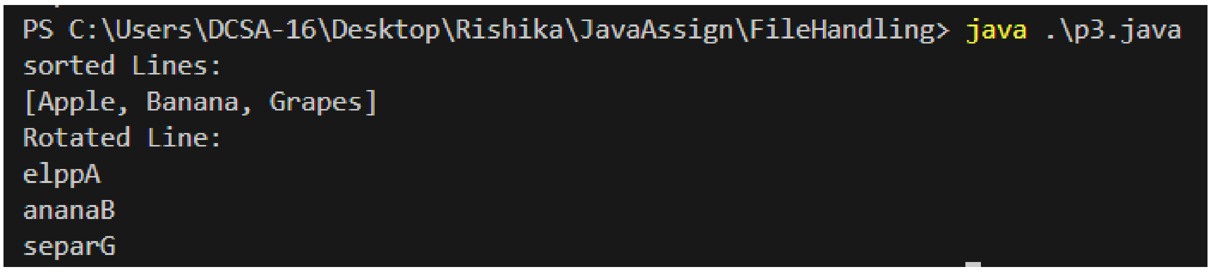


```
#*# ##*#*
#*###*#?
```

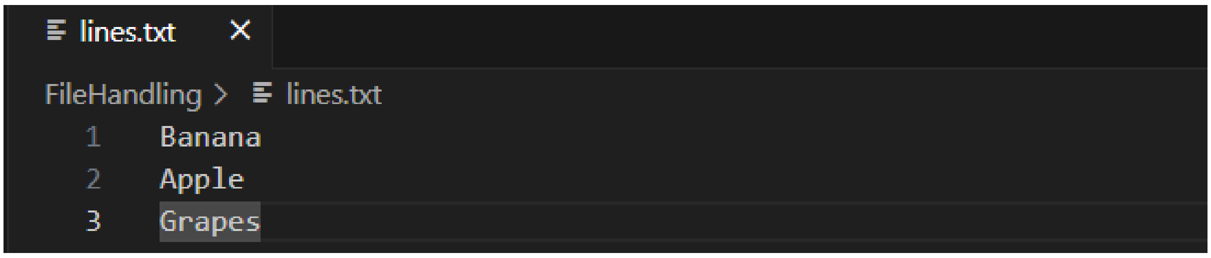
3. Create a Java program that:

- **Reads all lines from a file (lines.txt) using FileInputStream and Scanner.**
- **Sorts the lines in ascending alphabetical order.**
- **Reverses each line individually (character-wise).**
- **Writes the sorted and reversed lines to a new file (reversed_sorted_lines.txt) using FileOutputStream.**

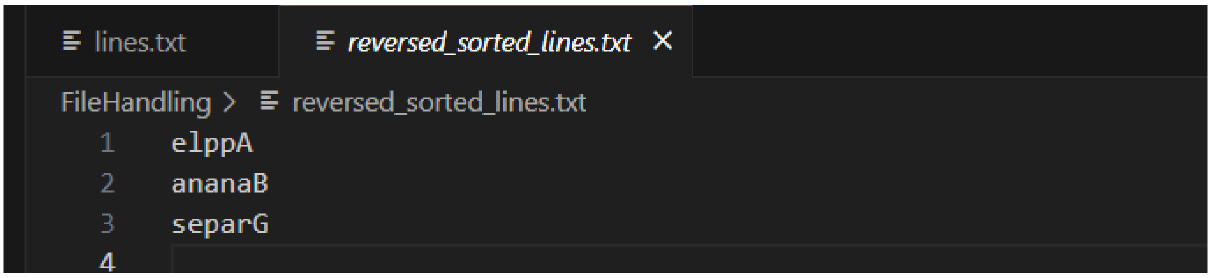
```java
import java.io.*;
import java.util.*;

public class p3 {
    public static void main(String[] args) {
        ArrayList<String> lines = new ArrayList<>();

        try (
            FileInputStream fis = new FileInputStream("lines.txt");
            Scanner scanner = new Scanner(fis)
        ){
            while (scanner.hasNextLine()) {
                lines.add(scanner.nextLine());
            }
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }

        Collections.sort(lines);
        System.out.println("sorted Lines: \n"+lines);

        try (
            FileOutputStream fos = new
FileOutputStream("reversed_sorted_lines.txt");
            PrintWriter writer = new PrintWriter(fos)
        ){
            System.out.println("Rotated Line:");
            for (String line : lines) {
                String reversed = new
StringBuilder(line).reverse().toString();
                System.out.println(reversed);
                writer.println(reversed);
            }
        } catch (IOException e) {
            System.out.println("Error : " + e.getMessage());
        }

    }
}
```



```
PS C:\Users\DCSA-16\Desktop\Rishika\JavaAssign\FileHandling> java .\p3.java
sorted Lines:
[Apple, Banana, Grapes]
Rotated Line:
elppA
ananaB
separG
```



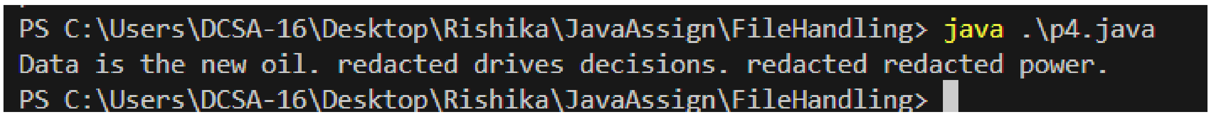
```
Banana
Apple
Grapes
```


lines.txt
reversed_sorted_lines.txt
FileHandling > reversed_sorted_lines.txt

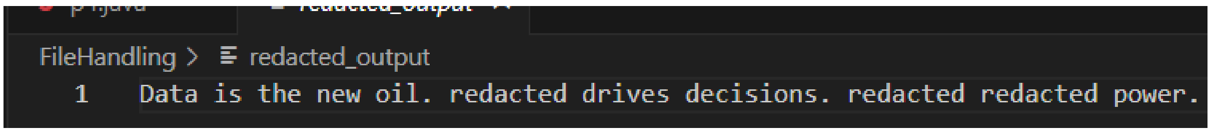
```
elppA
ananaB
separG

```

4. Write a Java program that:
• Reads a paragraph from a text file (paragraph.txt) using FileInputStream and Scanner.
• Replaces every second occurrence of each word with the string "REDACTED".
• Writes the modified paragraph to a new file (redacted_output.txt) using FileOutputStream.

```
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.util.HashMap;
import java.util.Scanner;

public class p4 {
    public static void main(String[] args) {

        HashMap<String, Integer> hash = new HashMap<>();
        try (
                FileInputStream fileS = new FileInputStream("paragraph.txt");
                Scanner FileSc = new Scanner(fileS);
                FileWriter output = new FileWriter("redacted_output");
        ) {
            while (FileSc.hasNext()) {
                String word = FileSc.next();
                int count = hash.getOrDefault(word, 0);

                if (count >= 1) {
                    System.out.print("redacted ");
                    output.write("redacted ");
                } else {
                    System.out.print(word + " ");
                    output.write(word + " ");
                }

                hash.put(word, count + 1);
            }
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```



```
PS C:\Users\DCSA-16\Desktop\Rishika\JavaAssign\FileHandling> java .\p4.java
Data is the new oil. redacted drives decisions. redacted redacted power.
PS C:\Users\DCSA-16\Desktop\Rishika\JavaAssign\FileHandling>
```



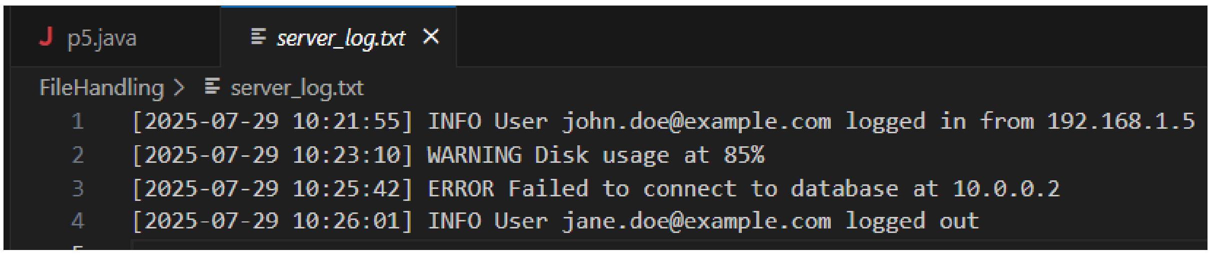
FileHandling > redacted_output

```
Data is the new oil. redacted drives decisions. redacted redacted power.
```

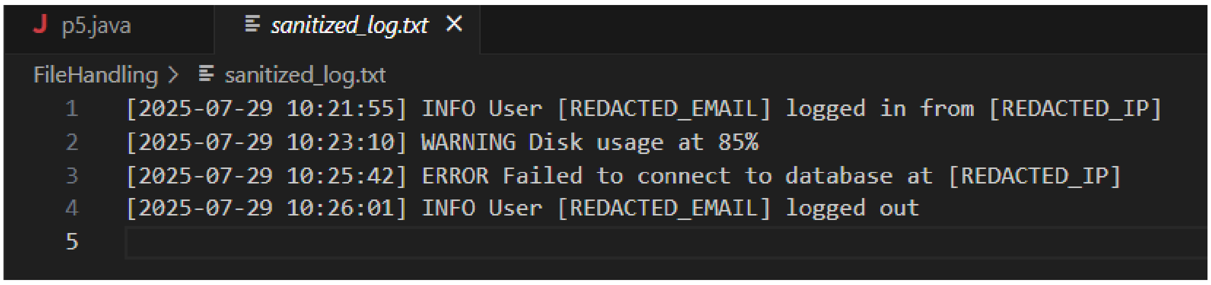
5. Write a Java program that analyzes a server log file (server_log.txt) containing multiple log entries. Each entry includes a timestamp, a log level, and a message. Your task is to:
a. Parse the log file using FileInputStream and Scanner.
b. Extract and categorize log entries by severity (INFO, WARNING, ERROR).
c. Count the number of occurrences of each log level.
d. Find and redact sensitive data (like email addresses and IP addresses).
e. Generate two output files:
• log_summary.txt — A summary of log counts per severity.
• sanitized_log.txt — The modified log with sensitive data redacted.

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

```java
public class p5 {
    public static void main(String[] args) throws IOException {
        FileInputStream fis = new FileInputStream("server_log.txt");
        Scanner scanner = new Scanner(fis);
        PrintWriter summaryWriter = new PrintWriter("log_summary.txt");
        PrintWriter sanitizedWriter = new PrintWriter("sanitized_log.txt");

        Map<String, Integer> logCounts = new HashMap<>();
        Pattern emailPattern = Pattern.compile("\\b[\\w.-]+@[\\w.-]+\\b");
        Pattern ipPattern =
Pattern.compile("\\b(?:\\d{1,3}\\.){3}\\d{1,3}\\b");

        while (scanner.hasNextLine()) {
            String line = scanner.nextLine();

            if (line.contains("INFO")) logCounts.merge("INFO", 1,
Integer::sum);
            else if (line.contains("WARNING")) logCounts.merge("WARNING", 1,
Integer::sum);
            else if (line.contains("ERROR")) logCounts.merge("ERROR", 1,
Integer::sum);

            line = emailPattern.matcher(line).replaceAll("[REDACTED_EMAIL]");
            line = ipPattern.matcher(line).replaceAll("[REDACTED_IP]");
            sanitizedWriter.println(line);
        }

        summaryWriter.println("Log Level Summary:");
        for (String level : List.of("INFO", "WARNING", "ERROR")) {
            summaryWriter.println(level + ": " + logCounts.getOrDefault(level,
0));
        }

        scanner.close();
        summaryWriter.close();
        sanitizedWriter.close();
    }
}
```



```
[2025-07-29 10:21:55] INFO User john.doe@example.com logged in from 192.168.1.5
[2025-07-29 10:23:10] WARNING Disk usage at 85%
[2025-07-29 10:25:42] ERROR Failed to connect to database at 10.0.0.2
[2025-07-29 10:26:01] INFO User jane.doe@example.com logged out
```



```
[2025-07-29 10:21:55] INFO User [REDACTED_EMAIL] logged in from [REDACTED_IP]
[2025-07-29 10:23:10] WARNING Disk usage at 85%
[2025-07-29 10:25:42] ERROR Failed to connect to database at [REDACTED_IP]
[2025-07-29 10:26:01] INFO User [REDACTED_EMAIL] logged out
```

Source Code-
Github : https://github.com/RiGa7/Advanced-Java-Assginment