**Implement the following projects which focus on different aspects of Java programming, including:**

Object-Oriented Programming principles

Exception handling

File operations

Data structures

Input validation

User interface design

**7. File Word Counter**

Develop a program that reads a text file and counts words, lines, and characters using object oriented programming in Java.

Key points:

- Use File I/O

- Handle file not found exceptions

- Create summary report

import java.io.\*;

import java.nio.file.\*;

import java.util.\*;

class FileStatistics {

private int wordCount;

private int lineCount;

private int charCount;

private Map<String, Integer> wordFrequency;

public FileStatistics() {

this.wordCount = 0;

this.lineCount = 0;

this.charCount = 0;

this.wordFrequency = new HashMap<>();

}

// Getters

public int getWordCount() { return wordCount; }

public int getLineCount() { return lineCount; }

public int getCharCount() { return charCount; }

public Map<String, Integer> getWordFrequency() { return wordFrequency; }

// Setters with validation

public void incrementWordCount() { this.wordCount++; }

public void incrementLineCount() { this.lineCount++; }

public void addToCharCount(int chars) { this.charCount += chars; }

public void addWord(String word) {

wordFrequency.put(word, wordFrequency.getOrDefault(word, 0) + 1);

}

}

class FileAnalysisReport {

private final FileStatistics statistics;

private final String filename;

public FileAnalysisReport(FileStatistics statistics, String filename) {

this.statistics = statistics;

this.filename = filename;

}

public String generateReport() {

StringBuilder report = new StringBuilder();

report.append("\nFile Analysis Report for: ").append(filename)

.append("\n====================================")

.append("\nTotal Lines: ").append(statistics.getLineCount())

.append("\nTotal Words: ").append(statistics.getWordCount())

.append("\nTotal Characters: ").append(statistics.getCharCount())

.append("\n\nTop 5 Most Frequent Words:")

.append("\n----------------------------\n");

// Get top 5 most frequent words

statistics.getWordFrequency().entrySet().stream()

.sorted(Map.Entry.<String, Integer>comparingByValue().reversed())

.limit(5)

.forEach(entry -> report.append(String.format("%-20s: %d occurrences\n",

"\"" + entry.getKey() + "\"", entry.getValue())));

return report.toString();

}

public void saveReportToFile(String outputPath) throws IOException {

try (BufferedWriter writer = new BufferedWriter(new FileWriter(outputPath))) {

writer.write(generateReport());

}

}

}

class FileWordCounter {

private final FileStatistics statistics;

public FileWordCounter() {

this.statistics = new FileStatistics();

}

public FileStatistics analyzeFile(String filePath) throws IOException {

try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {

String line;

while ((line = reader.readLine()) != null) {

processLine(line);

statistics.incrementLineCount();

}

}

return statistics;

}

private void processLine(String line) {

// Count characters (including whitespace)

statistics.addToCharCount(line.length());

// Process words

if (!line.trim().isEmpty()) {

String[] words = line.trim().split("\\s+");

for (String word : words) {

// Clean the word and convert to lowercase for consistent counting

word = word.replaceAll("[^a-zA-Z0-9']", "").toLowerCase();

if (!word.isEmpty()) {

statistics.incrementWordCount();

statistics.addWord(word);

}

}

}

}

public static void main(String[] args) {

try {

// Example usage

String inputFile = "sample.txt";

String outputFile = "report.txt";

// Create sample file if it doesn't exist (for demonstration)

if (!Files.exists(Paths.get(inputFile))) {

createSampleFile(inputFile);

}

// Analyze file

FileWordCounter counter = new FileWordCounter();

FileStatistics statistics = counter.analyzeFile(inputFile);

// Generate and save report

FileAnalysisReport report = new FileAnalysisReport(statistics, inputFile);

System.out.println(report.generateReport());

report.saveReportToFile(outputFile);

System.out.println("\nReport has been saved to: " + outputFile);

} catch (FileNotFoundException e) {

System.err.println("Error: File not found - " + e.getMessage());

} catch (IOException e) {

System.err.println("Error: Unable to process file - " + e.getMessage());

} catch (Exception e) {

System.err.println("Unexpected error occurred: " + e.getMessage());

}

}

// Helper method to create a sample file for testing

private static void createSampleFile(String filename) throws IOException {

List<String> sampleText = Arrays.asList(

"The quick brown fox jumps over the lazy dog.",

"Hello world! This is a sample text file.",

"Java programming is fun and educational.",

"The quick brown fox is very quick indeed.",

"Programming in Java helps learn OOP concepts."

);

Files.write(Paths.get(filename), sampleText);

}

}

**o/p**

File Analysis Report for: sample.txt

====================================

Total Lines: 5

Total Words: 38

Total Characters: 210

Top 5 Most Frequent Words:

----------------------------

"quick" : 3 occurrences

"is" : 3 occurrences

"the" : 3 occurrences

"fox" : 2 occurrences

"java" : 2 occurrences

Report has been saved to: report.txt