**Week 2 – S2 –Assignment Homework Solution**

**Name: Ramesh Harisabapathi Chettiar**

**Roll Number: RA2411030010263**

**Course: Networking and Communications**

**Semester: 3**

**Date of Submission: 23/08/2025**

**Problem 1: Write a program to implement a simple spell checker that finds**

**and suggests corrections for misspelled words using string distance**

**calculation**

**Hint =>**

**a. Take user input for a sentence and a dictionary of correct words (stored in an array)**

**b. Create a method to split the sentence into words without using split():**

**● i. Use charAt() to identify word boundaries (spaces, punctuation)**

**● ii. Extract each word using substring() method**

**● iii. Store words in an array**

**c. Create a method to calculate string distance between two words:**

**● i. Count character differences between words of same length**

**● ii. For different lengths, calculate insertion/deletion distance**

**● iii. Return the distance as an integer**

**d. Create a method to find the closest matching word from dictionary:**

**● i. Compare input word with each dictionary word**

**● ii. Find the word with minimum distance**

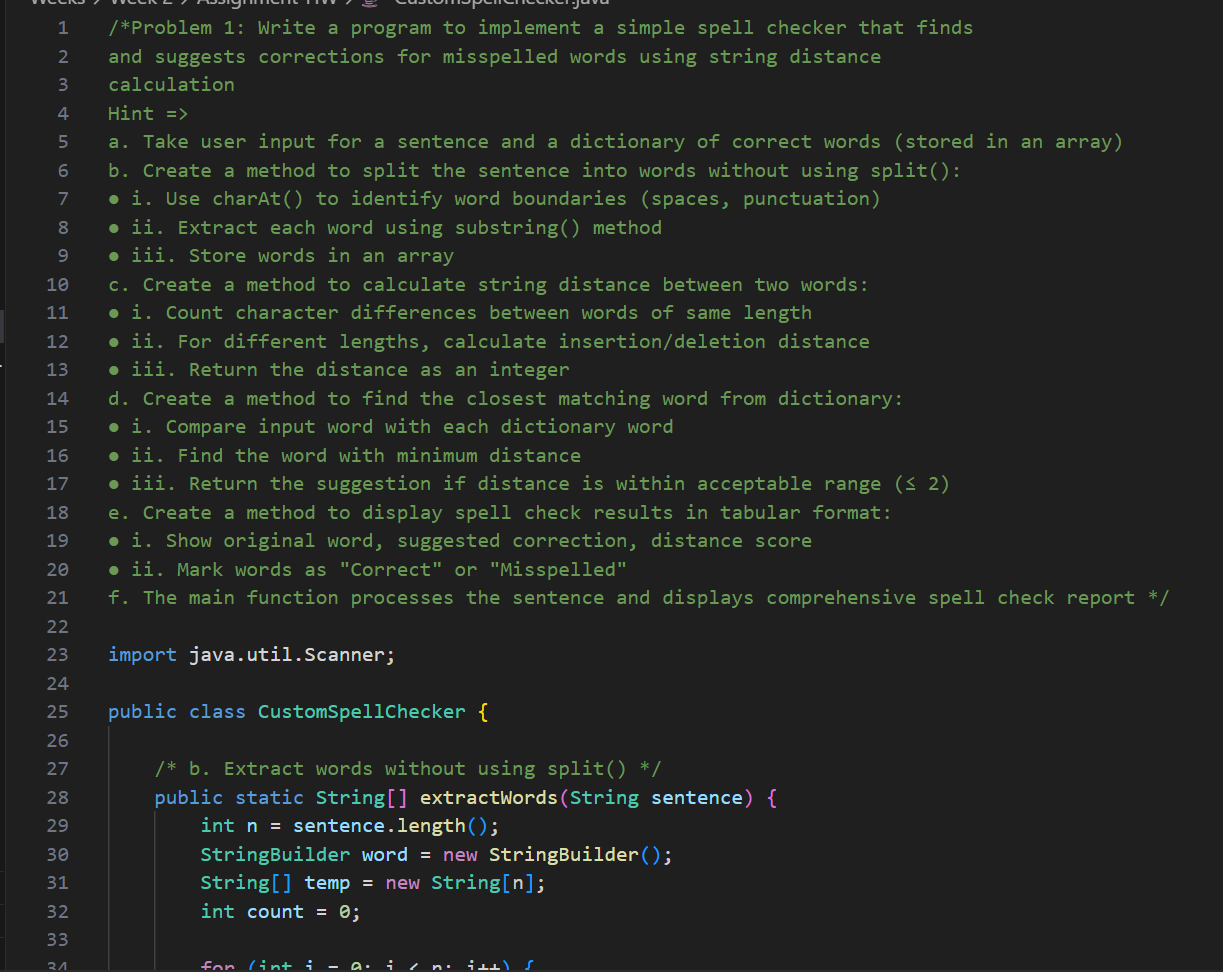
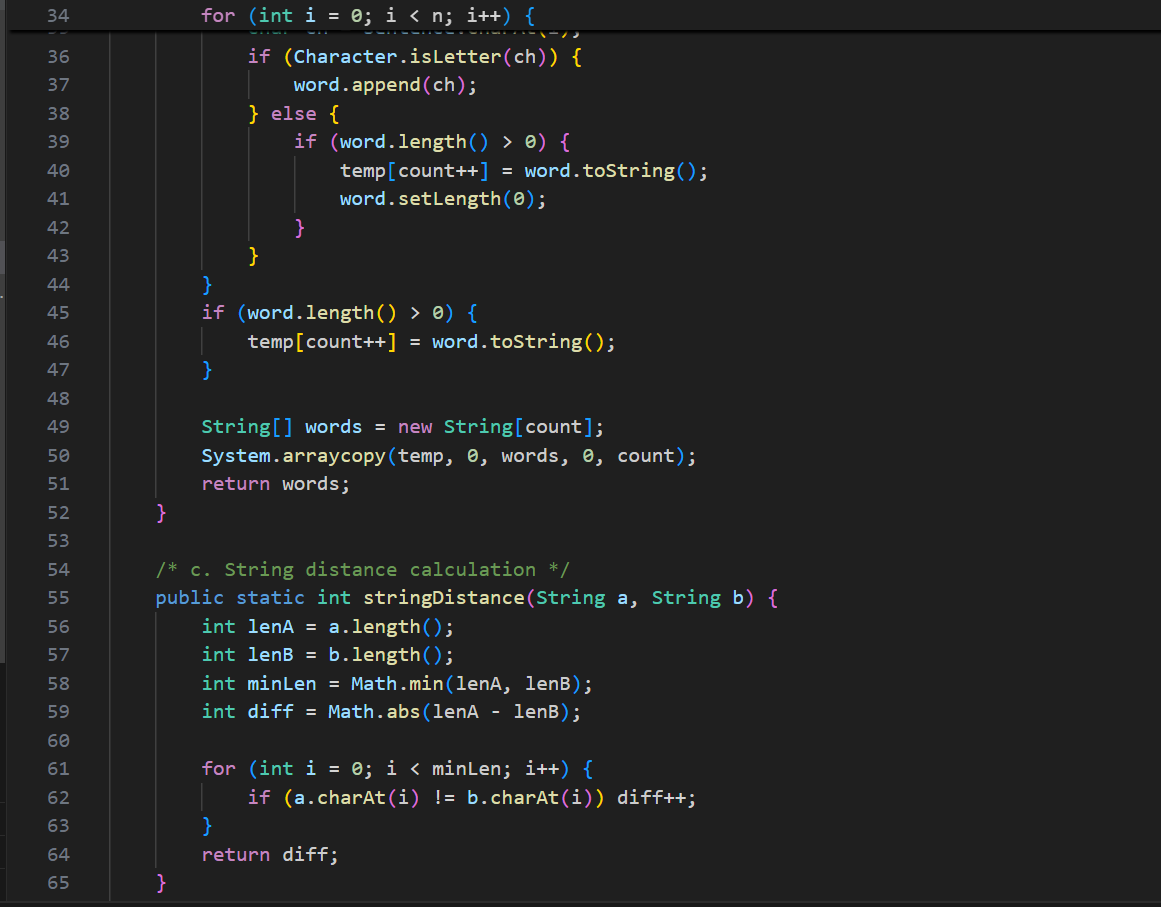
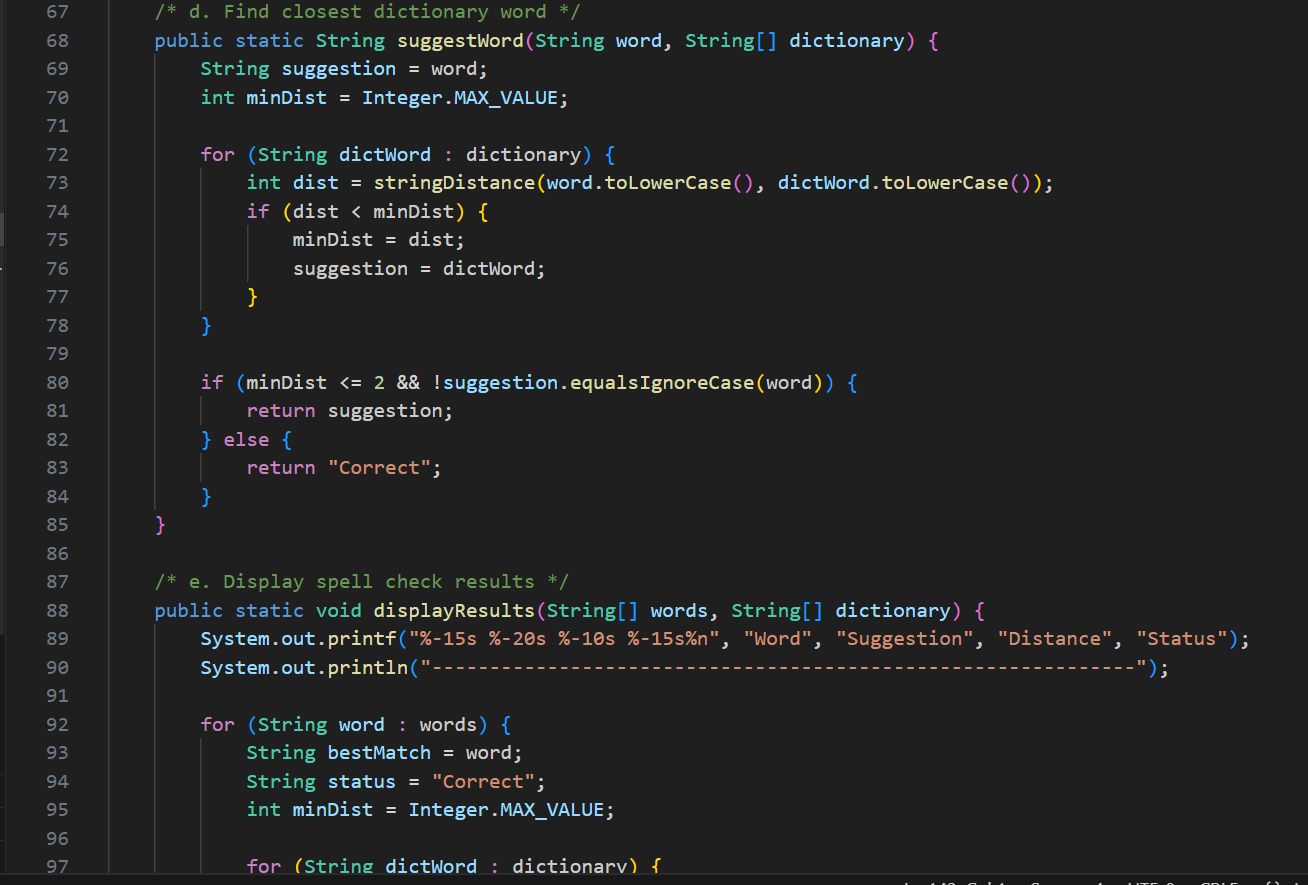
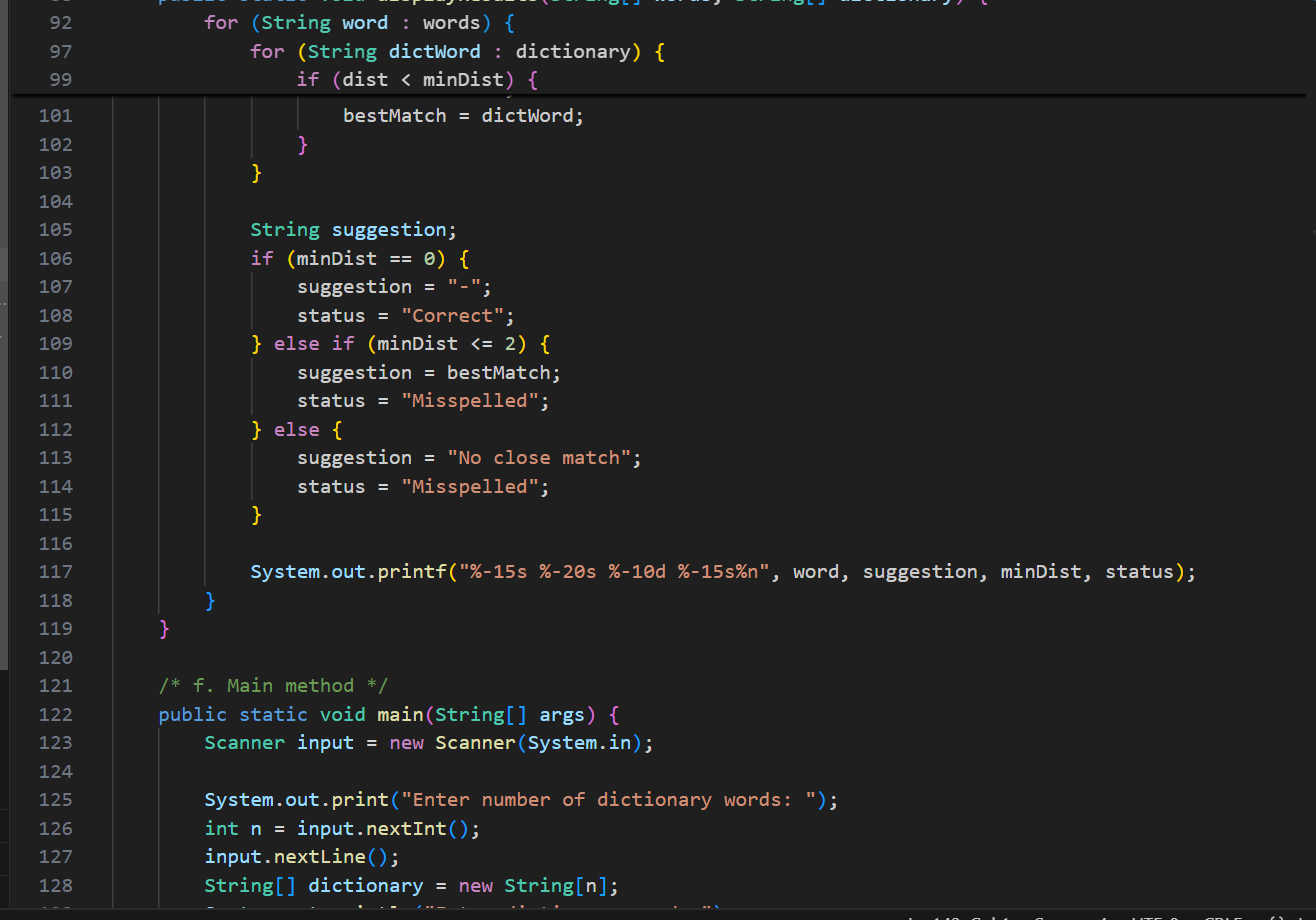
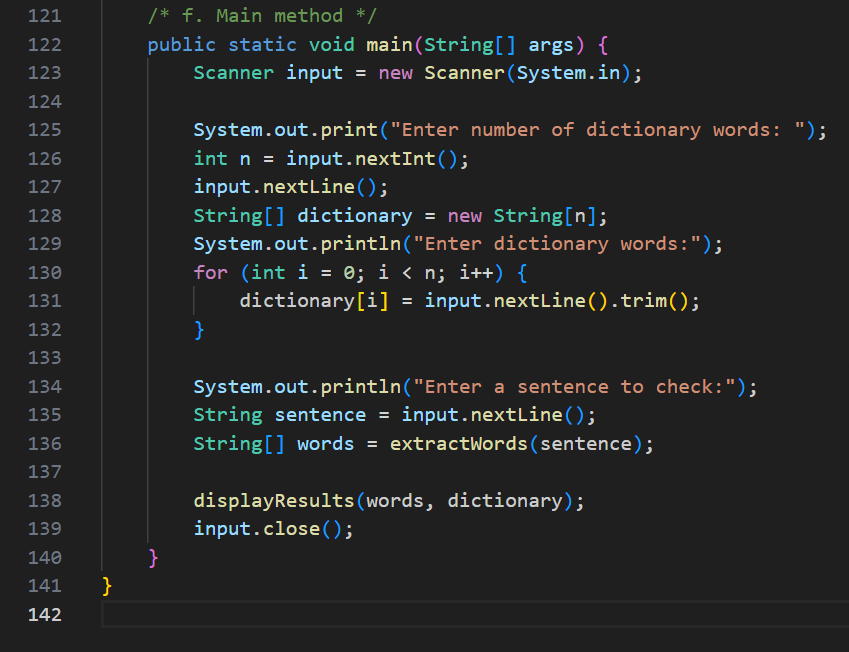
**● iii. Return the suggestion if distance is within acceptable range (≤ 2)**

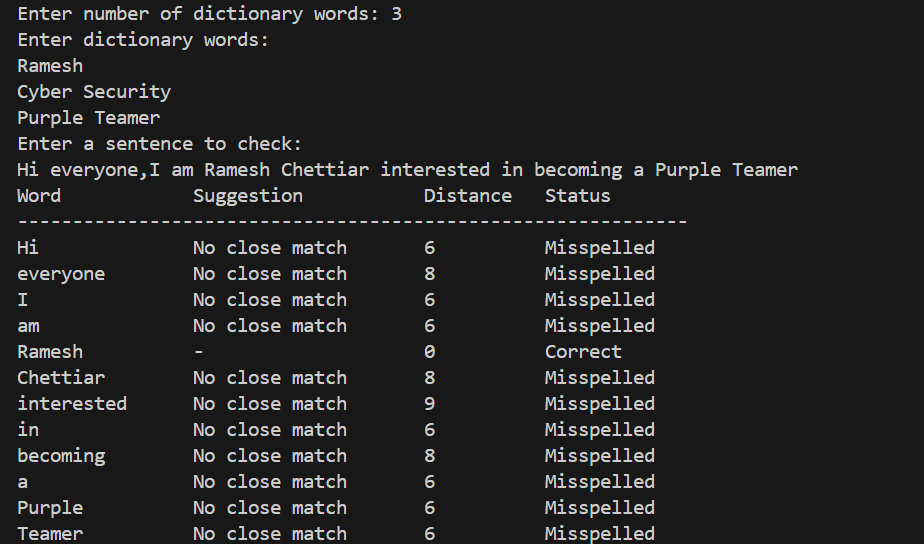
**e. Create a method to display spell check results in tabular format:**

**● i. Show original word, suggested correction, distance score**

**● ii. Mark words as "Correct" or "Misspelled"**

**f. The main function processes the sentence and displays comprehensive spell check report**

**** ****   

OUTPUT🡪 

**Problem 2: Write a program to create a password strength analyzer and**

**generator using ASCII values and StringBuilder**

**Hint =>**

**a. Take user input for multiple passwords to analyze**

**b. Create a method to analyze password strength using ASCII values:**

**● i. Count uppercase letters (ASCII 65-90)**

**● ii. Count lowercase letters (ASCII 97-122)**

**● iii. Count digits (ASCII 48-57)**

**● iv. Count special characters (other printable ASCII)**

**● v. Check for common patterns and sequences**

**c. Create a method to calculate password strength score:**

**● i. Length points: +2 per character above 8**

**● ii. Character variety: +10 for each type present**

**● iii. Deduct points for common patterns (123, abc, qwerty)**

**● iv. Return strength level: Weak (0-20), Medium (21-50), Strong (51+)**

**d. Create a method using StringBuilder to generate strong passwords:**

**● i. Take desired length as parameter**

**● ii. Ensure at least one character from each category**

**● iii. Fill remaining positions with random characters**

**● iv. Shuffle the password for better randomness**

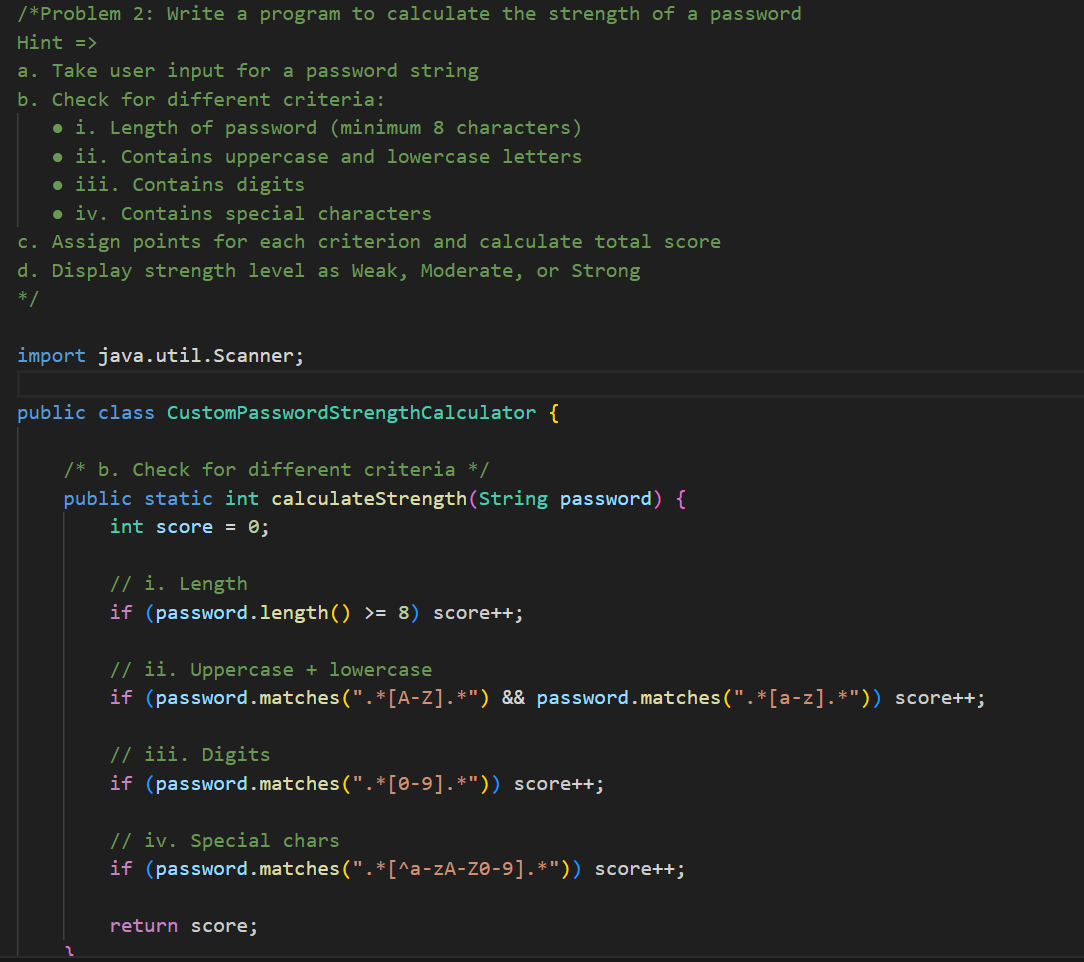
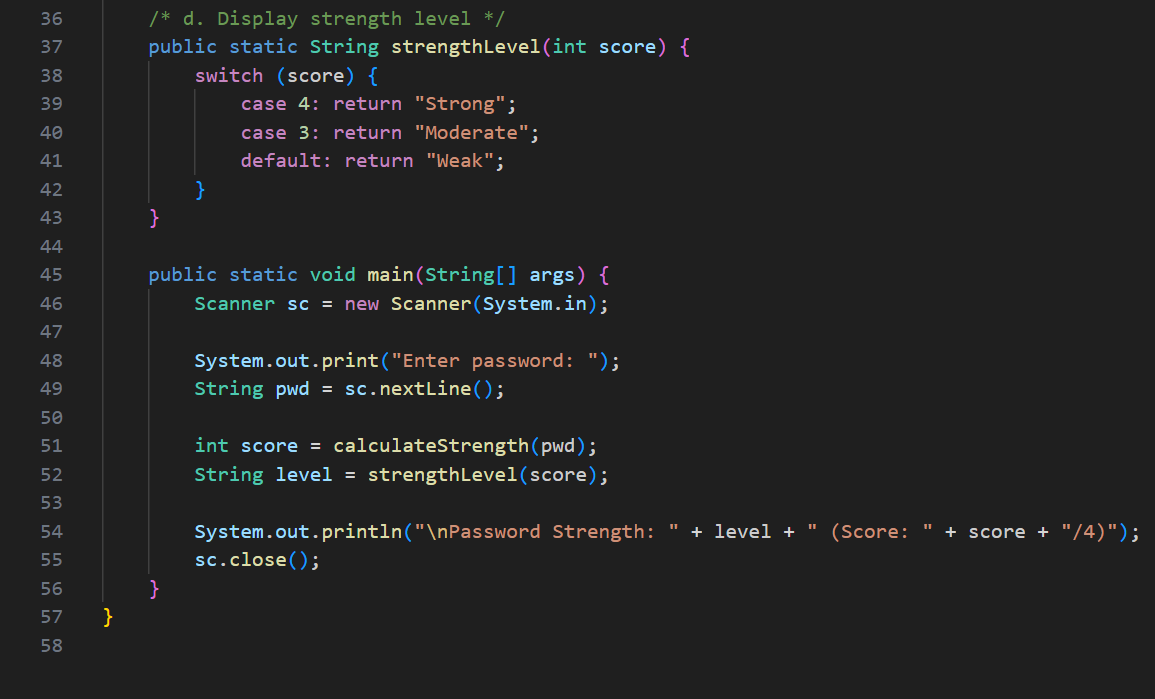
**e. Create a method to display analysis results in tabular format:**

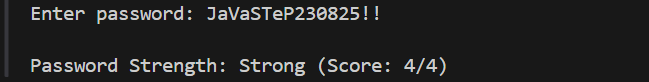
**i. Password, Length, Uppercase count, Lowercase count, Digits, Special chars, Score,**

**Strength**

**f. The main function analyzes existing passwords and generates new strong passwords based**

**on user requirements**

**** ****

**OUTPUT🡪** ****

**Problem 3: Write a program to implement a text-based data compression**

**algorithm using character frequency and StringBuilder**

**Hint =>**

**a. Take user input for text to compress**

**b. Create a method to count character frequency without using HashMap:**

**● i. Create arrays to store characters and their frequencies**

**● ii. Use charAt() to iterate through text**

**● iii. Count occurrences of each unique character**

**● iv. Return parallel arrays of characters and frequencies**

**c. Create a method to create compression codes using StringBuilder:**

**● i. Assign shorter codes to more frequent characters**

**● ii. Use numbers/symbols for common characters**

**● iii. Create a mapping table of original character to code**

**● iv. Return the mapping as a 2D array**

**d. Create a method to compress text using the generated codes:**

**● i. Replace each character with its corresponding code**

**● ii. Use StringBuilder for efficient string building**

**● iii. Calculate compression ratio (original size vs compressed size)**

**e. Create a method to decompress the text:**

**● i. Reverse the compression process using the mapping table**

**● ii. Validate that decompression returns original text**

**f. Create a method to display compression analysis:**

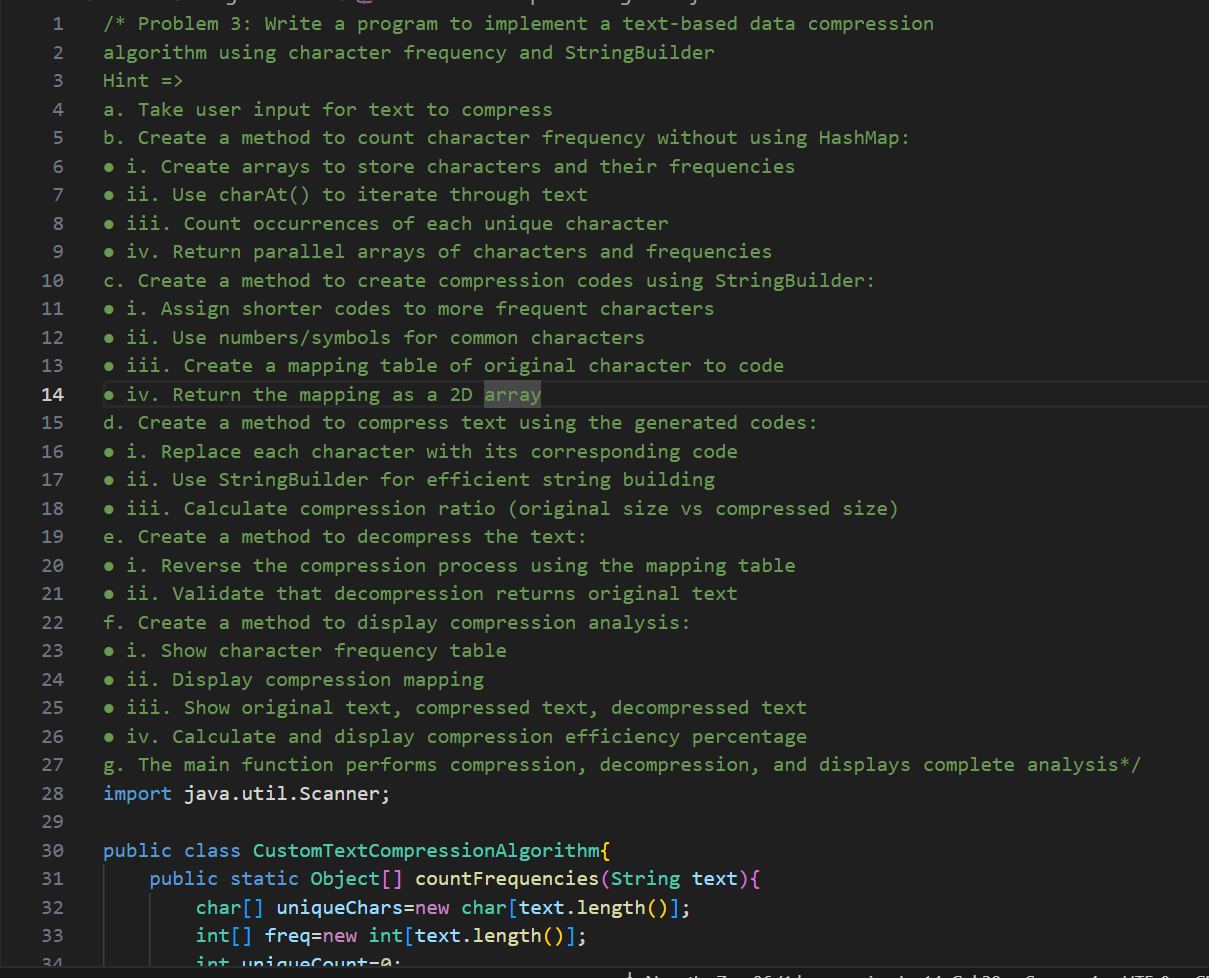
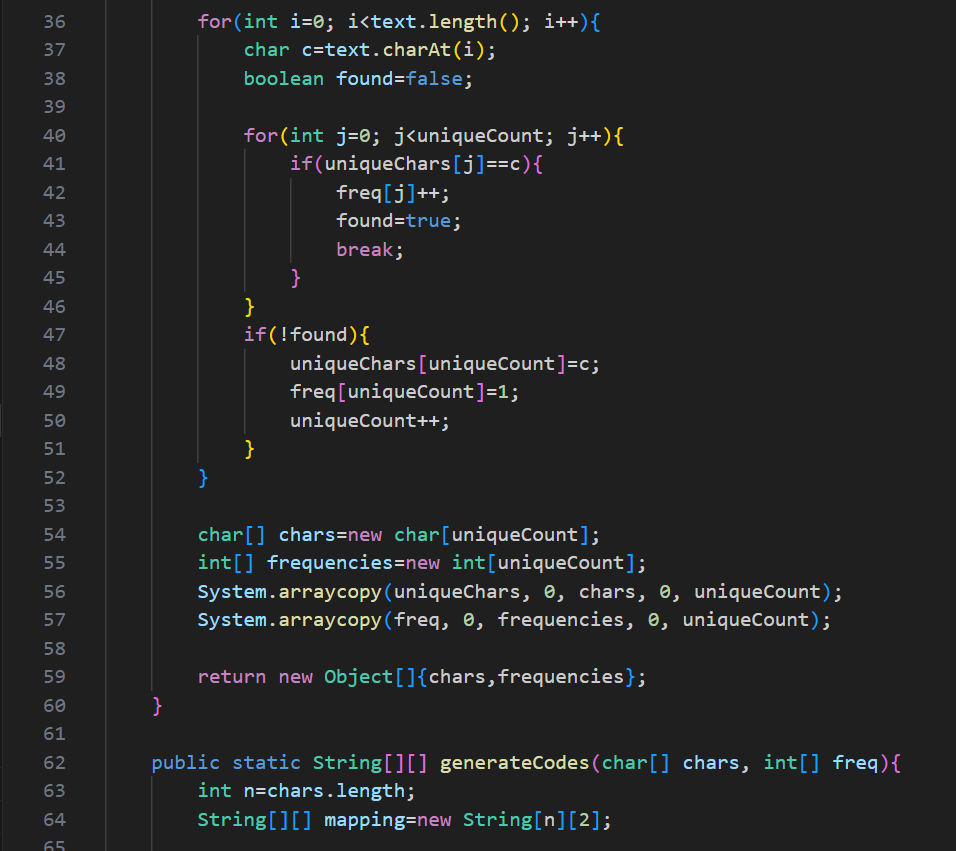
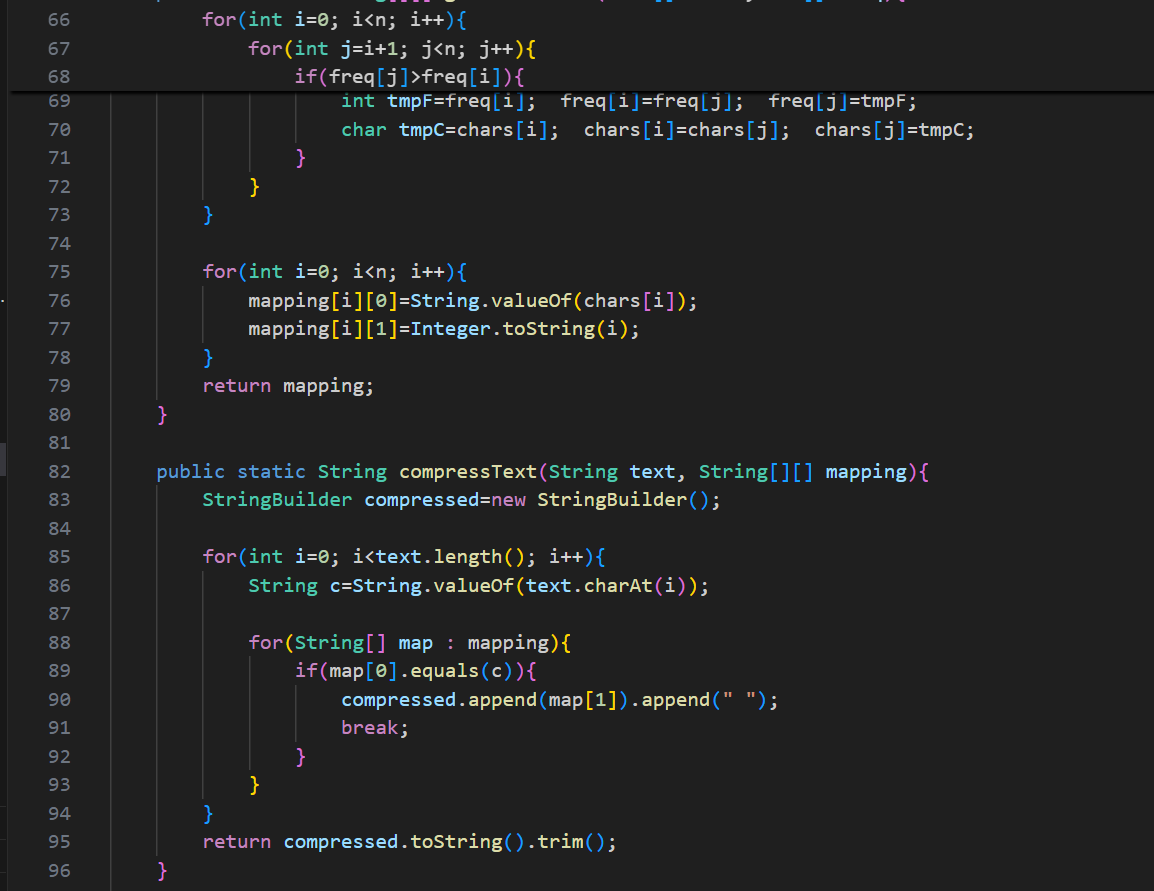
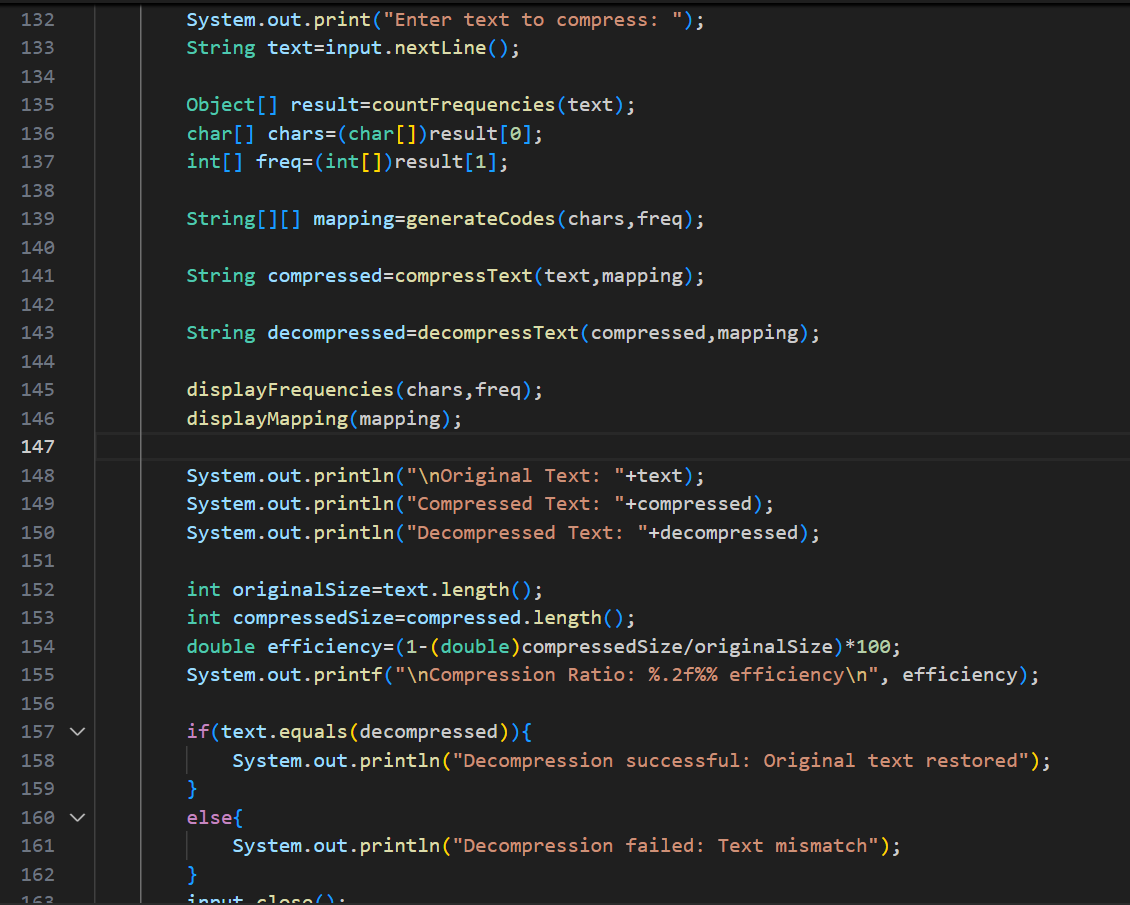
**● i. Show character frequency table**

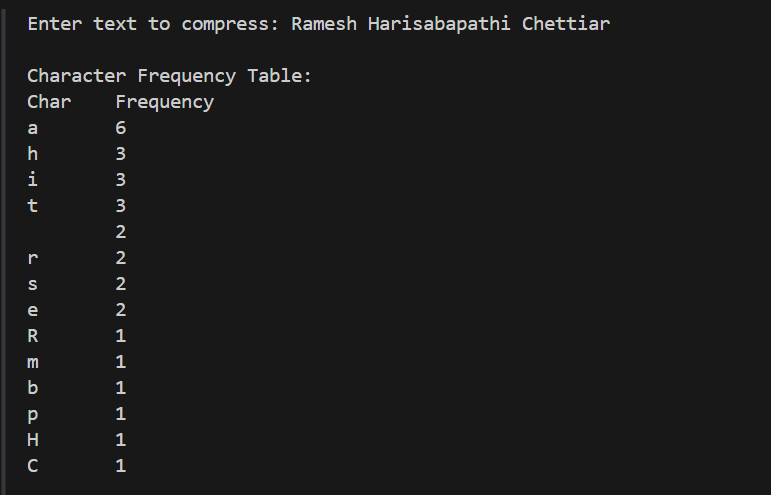
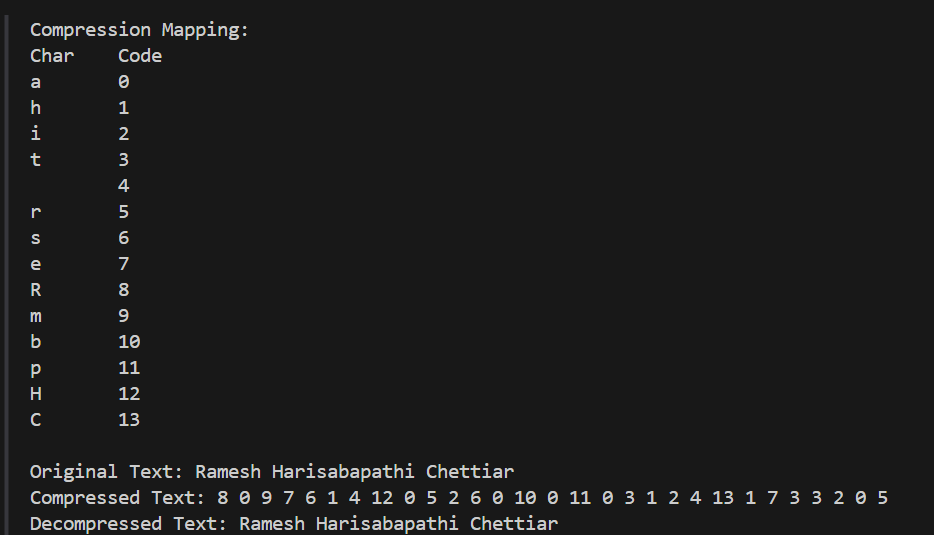
**● ii. Display compression mapping**

**● iii. Show original text, compressed text, decompressed text**

**● iv. Calculate and display compression efficiency percentage**

**g. The main function performs compression, decompression, and displays complete analysis**

**** ****    

**OUTPUT🡪** ****  

**Problem 4: Write a program to create a text-based calculator that can parse**

**and evaluate mathematical expressions from strings**

**Hint =>**

**a. Take user input for mathematical expressions as strings (e.g., "15 + 23 \* 4 - 10")**

**b. Create a method to validate expression format:**

**● i. Check for valid characters (digits, operators, spaces, parentheses)**

**● ii. Validate operator placement and parentheses matching**

**● iii. Use ASCII values to identify different character types**

**● iv. Return boolean indicating if expression is valid**

**c. Create a method to parse numbers from string:**

**● i. Use charAt() to identify digit sequences**

**● ii. Extract multi-digit numbers using substring()**

**● iii. Convert string numbers to integers**

**● iv. Store numbers and operators in separate arrays**

**d. Create a method to evaluate expression using order of operations:**

**● i. Handle multiplication and division first**

**● ii. Then handle addition and subtraction**

**● iii. Process from left to right for same precedence**

**● iv. Use StringBuilder to show step-by-step calculation**

**e. Create a method to handle parentheses:**

**● i. Find innermost parentheses using indexOf() and lastIndexOf()**

**● ii. Evaluate expressions inside parentheses first**

**● iii. Replace parenthetical results in main expression**

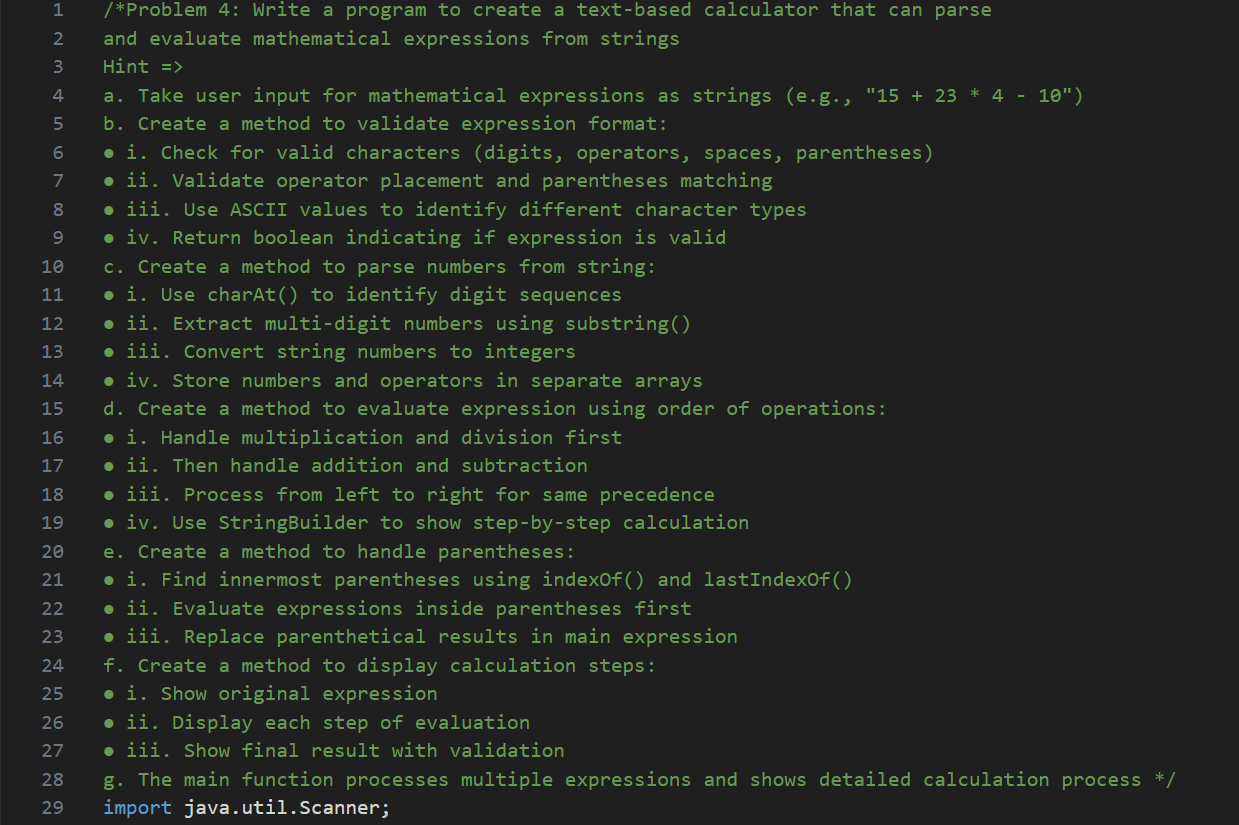
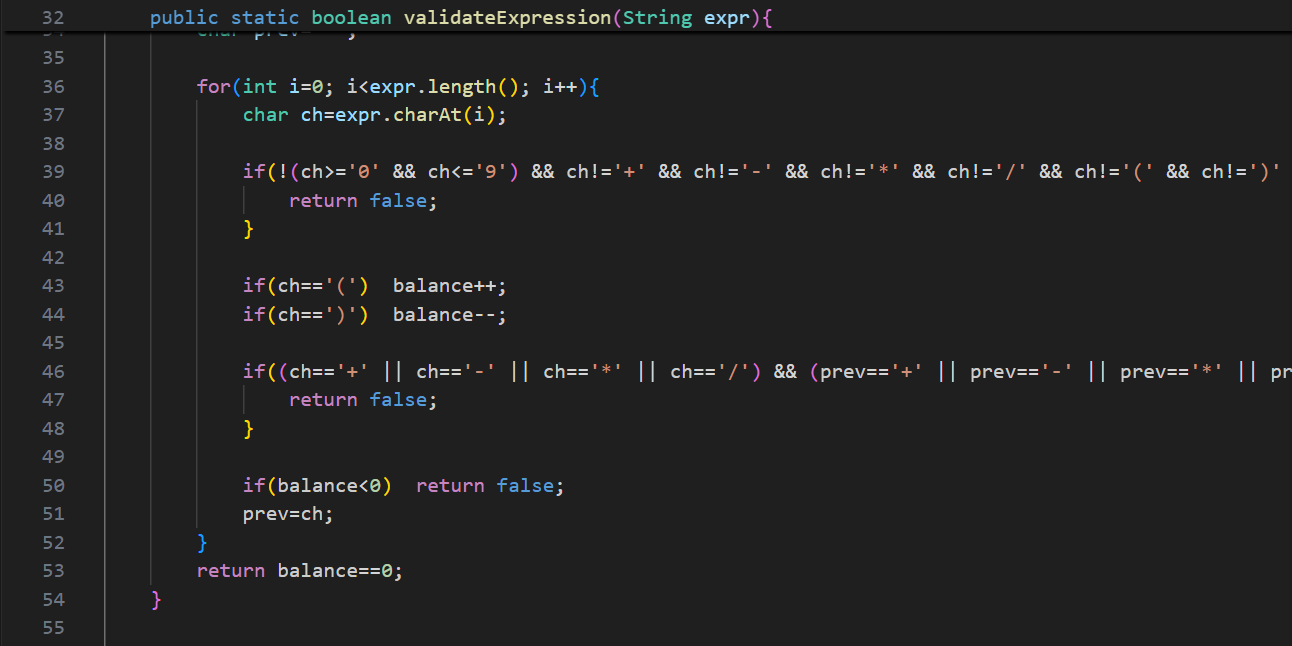
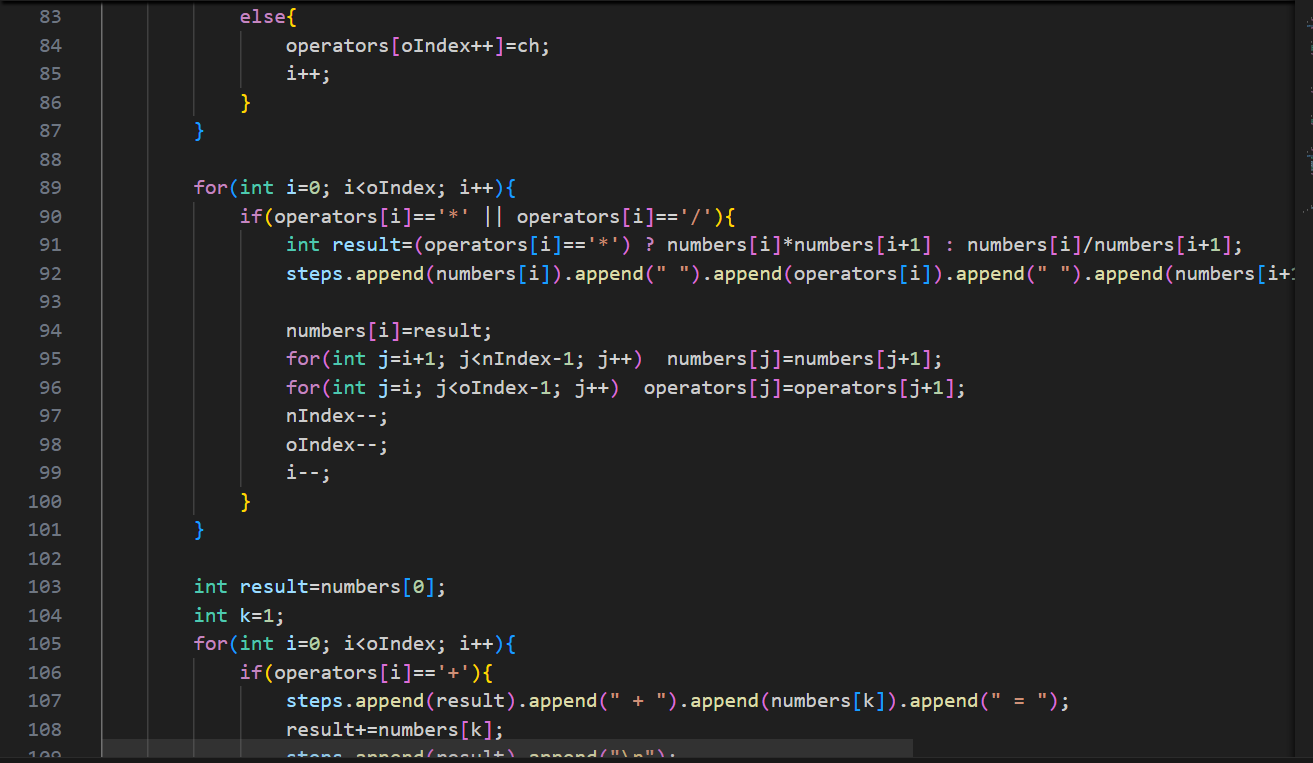
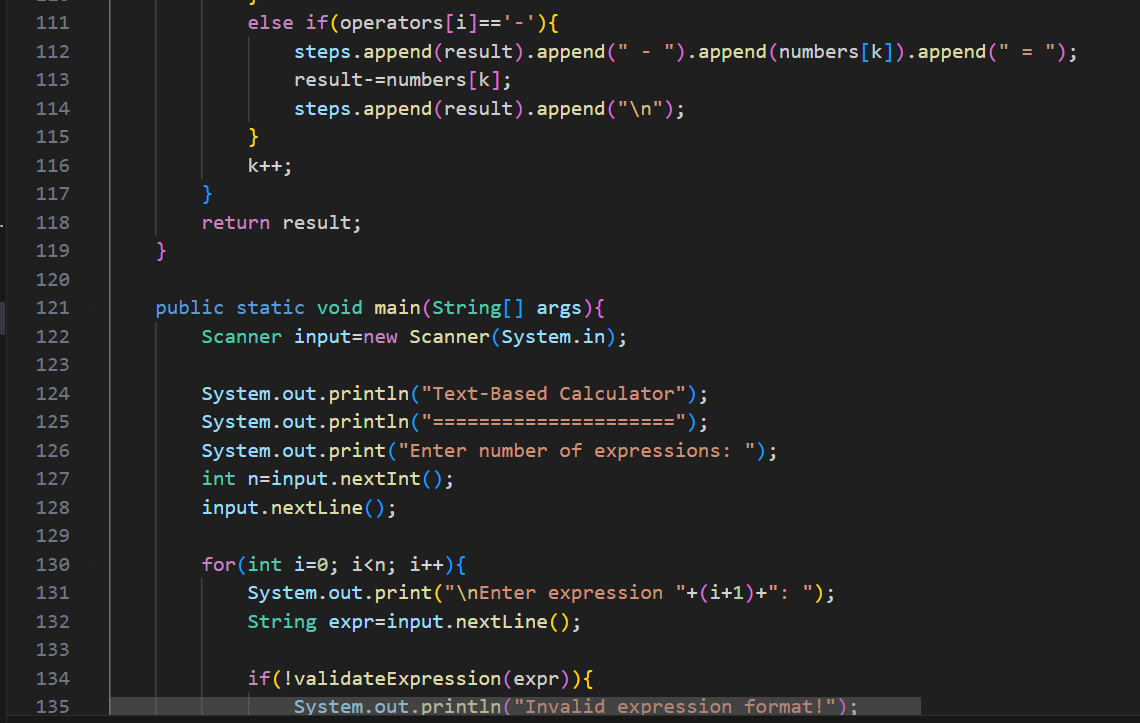
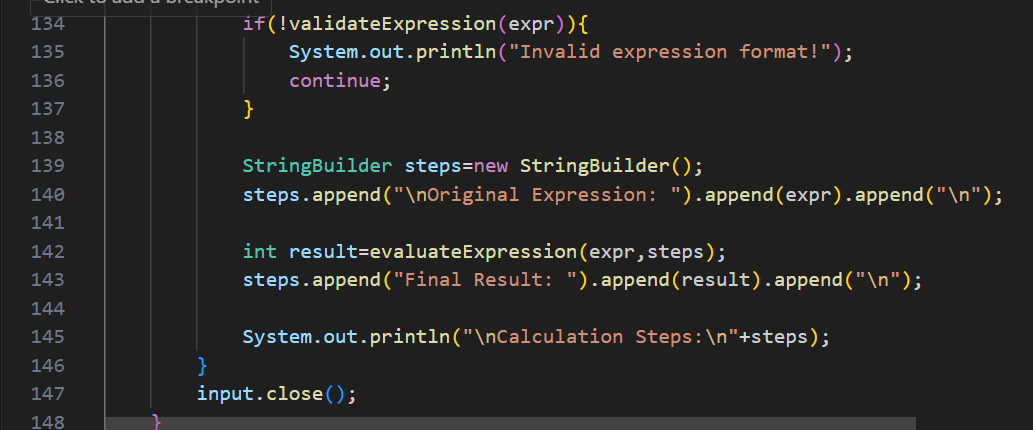
**f. Create a method to display calculation steps:**

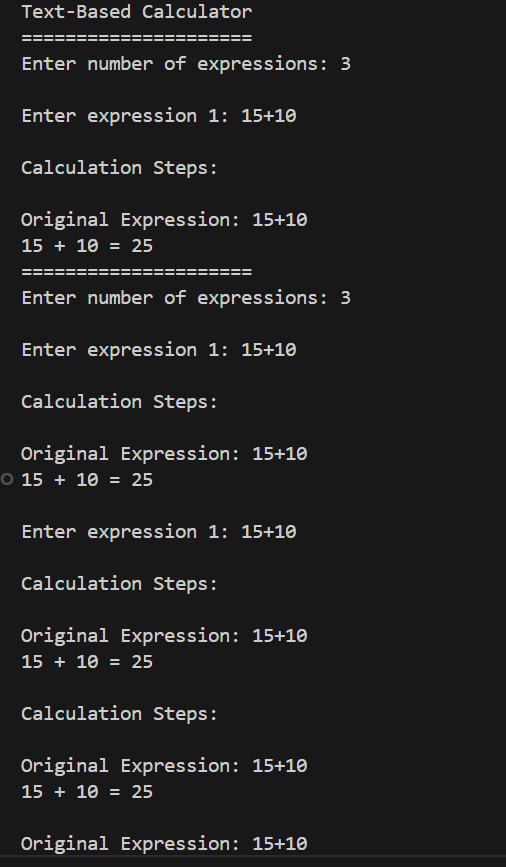
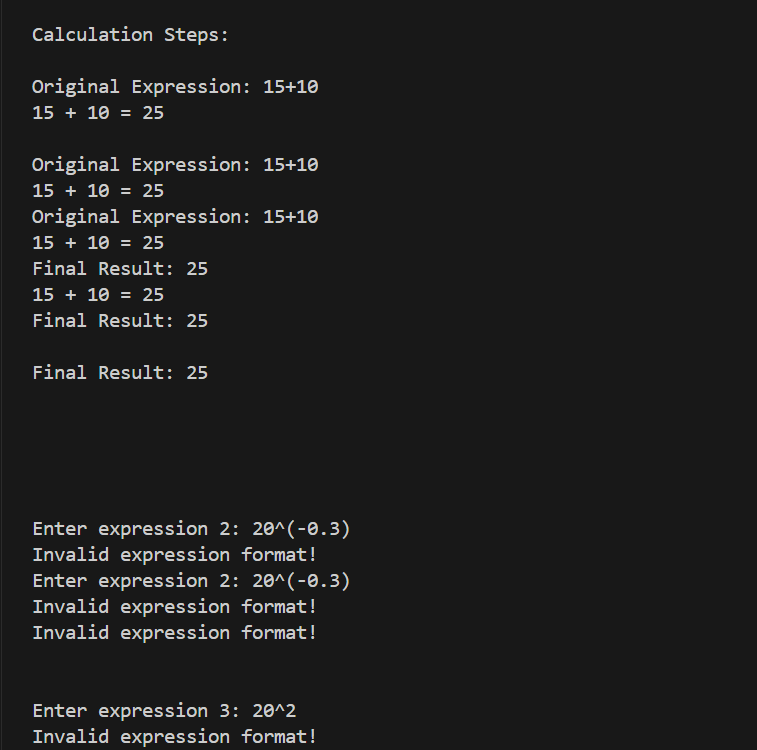
**● i. Show original expression**

**● ii. Display each step of evaluation**

**● iii. Show final result with validation**

**g. The main function processes multiple expressions and shows detailed calculation process**

**** ****    

**OUTPUT🡪** **** 

**Problem 5: Write a program to analyze and format structured data from**

**CSV-like text input using string manipulation methods**

**Hint =>**

**a. Take user input for CSV-like data (comma-separated values in multiple lines)**

**b. Create a method to parse CSV data without using split():**

**● i. Use charAt() to identify commas and newlines**

**● ii. Extract each field using substring() method**

**● iii. Handle quoted fields that may contain commas**

**● iv. Store data in a 2D array structure**

**c. Create a method to validate and clean data:**

**● i. Remove leading/trailing spaces from each field**

**● ii. Validate numeric fields using ASCII values**

**● iii. Check for missing or invalid data**

**● iv. Apply data type conversions where needed**

**d. Create a method to perform data analysis:**

**● i. Calculate column statistics (min, max, average for numeric columns)**

**● ii. Count unique values in categorical columns**

**● iii. Identify data quality issues (missing, invalid entries)**

**e. Create a method using StringBuilder to format output:**

**● i. Create aligned tabular display with fixed column widths**

**● ii. Add borders and headers for better readability**

**● iii. Format numeric values with proper decimal places**

**● iv. Highlight data quality issues**

**f. Create a method to generate data summary report:**

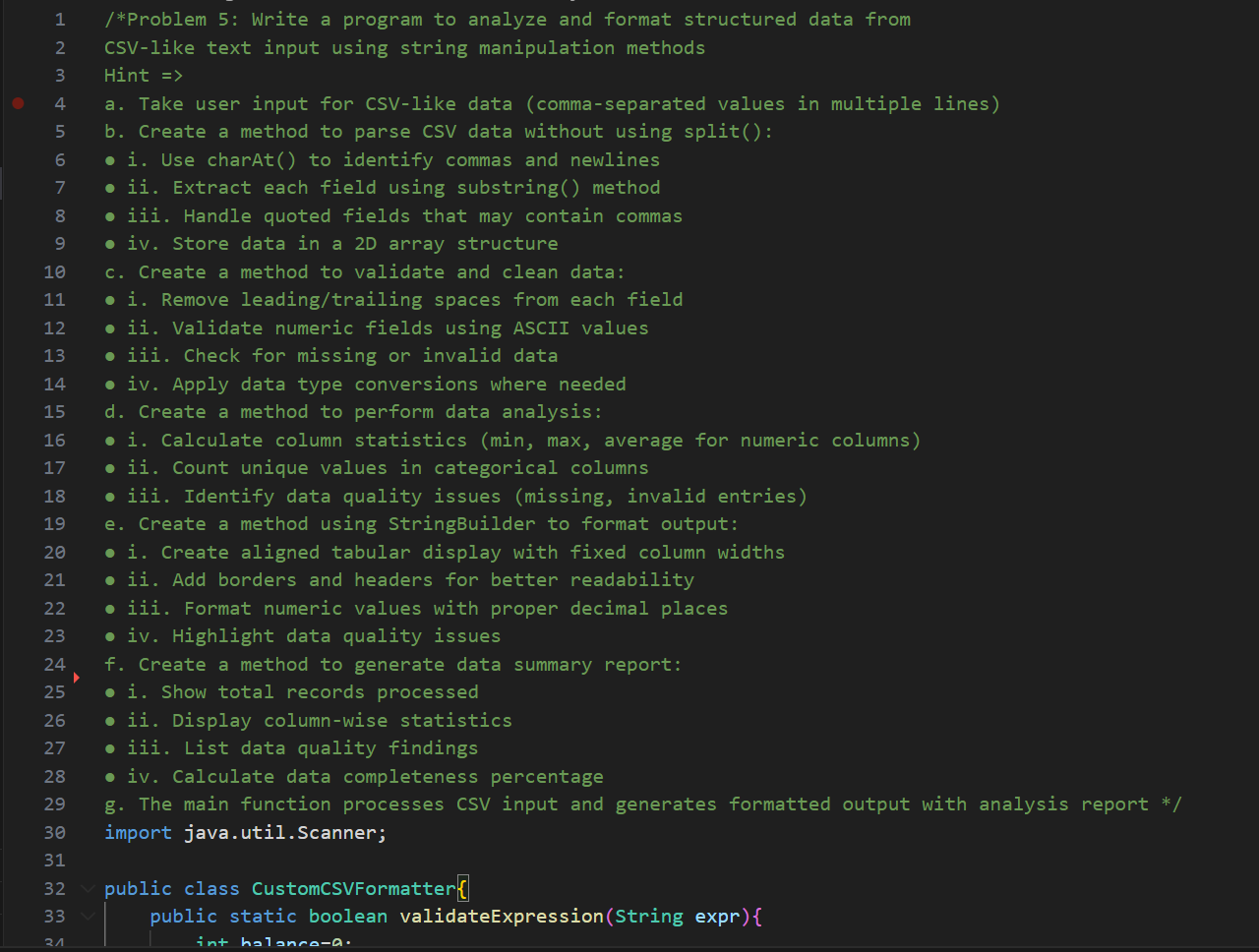
**● i. Show total records processed**

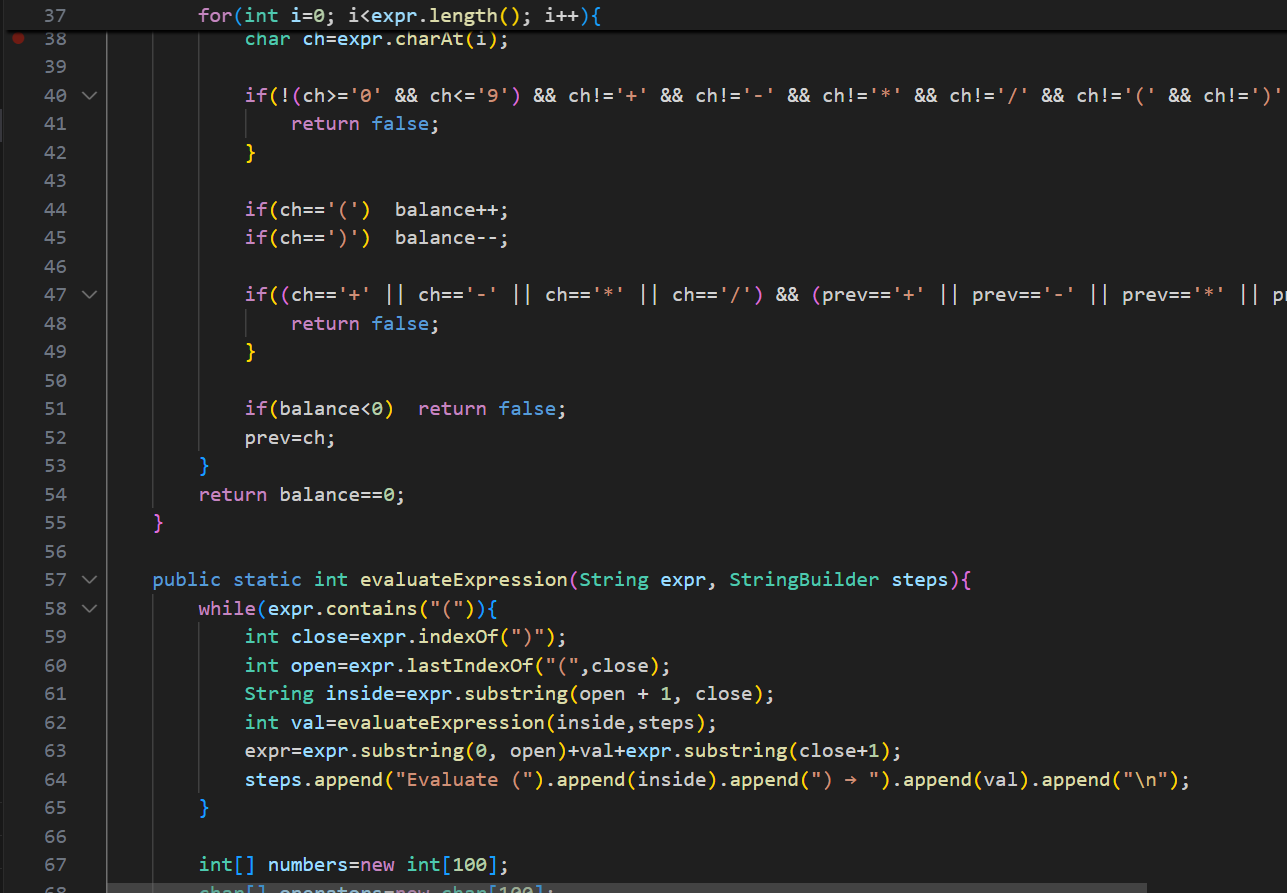
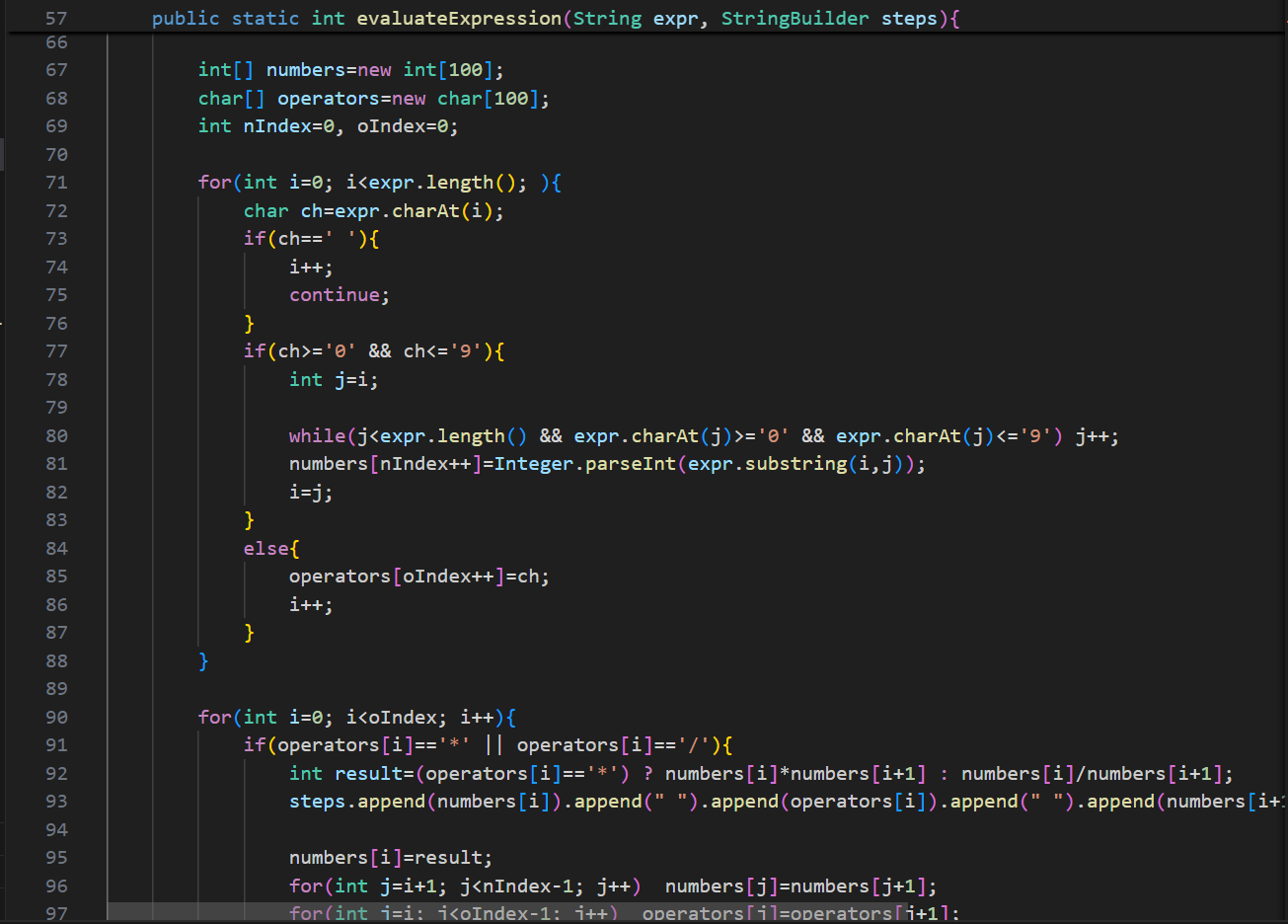
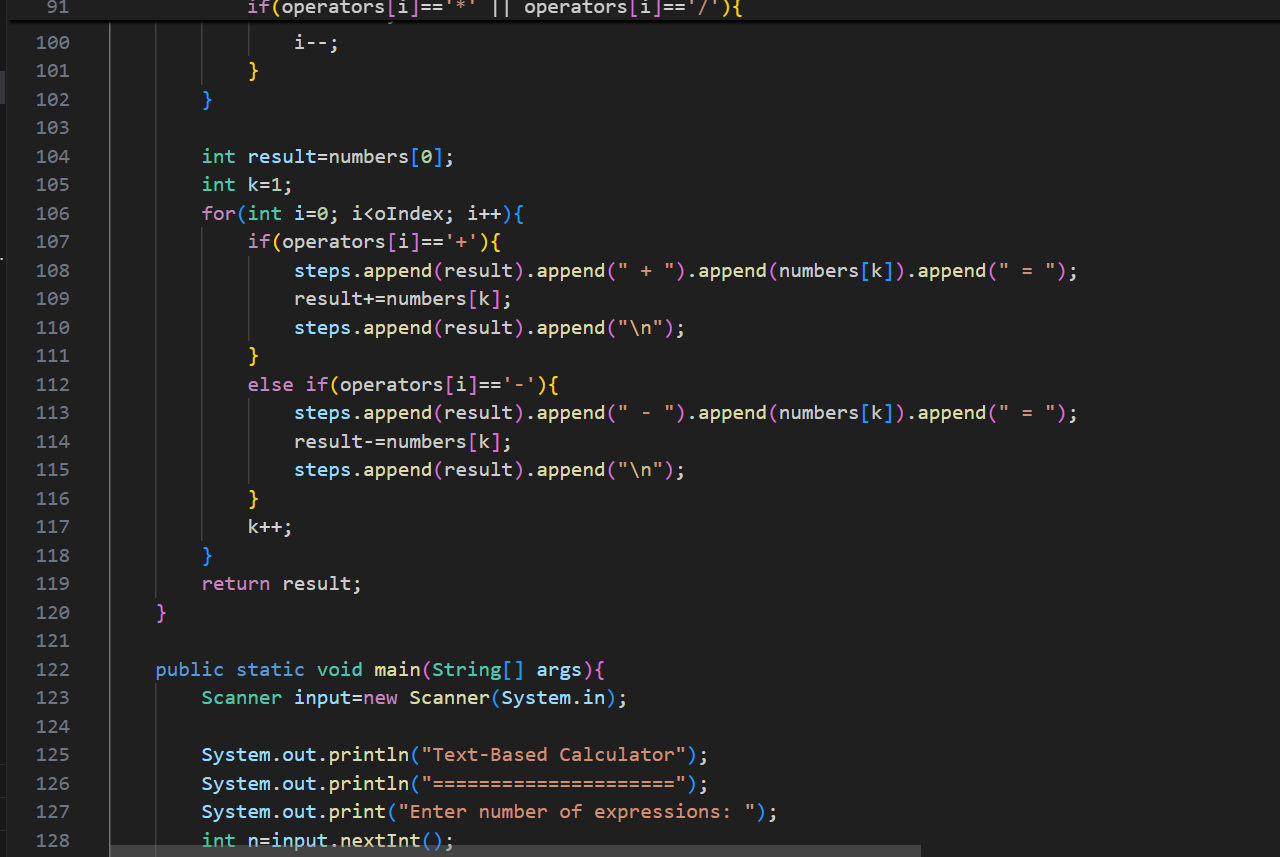
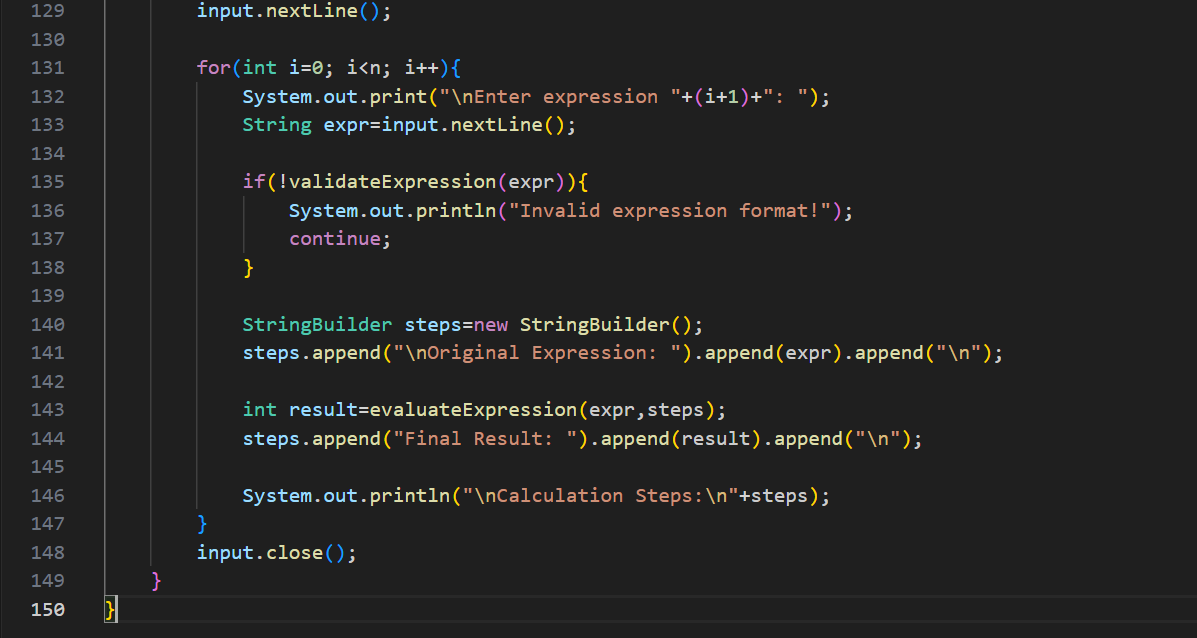
**● ii. Display column-wise statistics**

**● iii. List data quality findings**

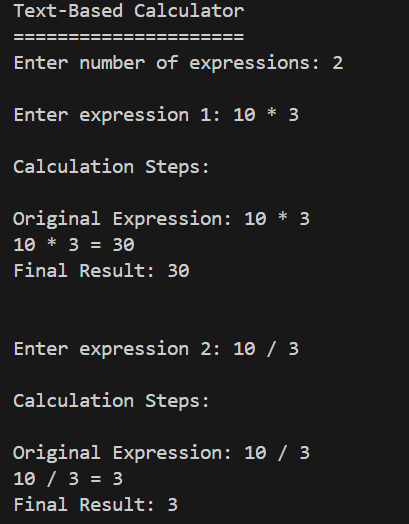
**● iv. Calculate data completeness percentage**

**g. The main function processes CSV input and generates formatted output with analysis report**

****

**** ****  

OUTPUT🡪

****

**Problem 6: Write a program to create a simple text-based file organizer that**

**categorizes and renames files based on their extensions and content**

**analysis**

**Hint =>**

**a. Take user input for multiple file names with extensions**

**b. Create a method to extract file components without using split():**

**● i. Use lastIndexOf() to find the last dot for extension**

**● ii. Extract filename and extension using substring()**

**● iii. Validate file name format and characters**

**● iv. Store file information in structured format**

**c. Create a method to categorize files by extension:**

**● i. Define categories (Documents: .txt, .doc; Images: .jpg, .png; etc.)**

**● ii. Use string comparison methods to match extensions**

**● iii. Count files in each category**

**● iv. Identify unknown file types**

**d. Create a method using StringBuilder to generate new file names:**

**● i. Create naming convention based on category and date**

**● ii. Handle duplicate names by adding numbers**

**● iii. Ensure generated names follow proper file naming rules**

**● iv. Validate that new names don't contain invalid characters**

**e. Create a method to simulate content-based analysis:**

**● i. For text files, analyze content for keywords**

**● ii. Suggest subcategories based on content (Resume, Report, Code, etc.)**

**● iii. Calculate file priority based on name patterns and content**

**● iv. Use ASCII values to validate content characters**

**f. Create a method to display file organization report:**

**● i. Show original filename, category, new suggested name**

**● ii. Display category-wise file counts in tabular format**

**● iii. List files that need attention (invalid names, unknown types)**

**● iv. Show organization statistics and recommendations**

**g. Create a method to generate batch rename commands:**

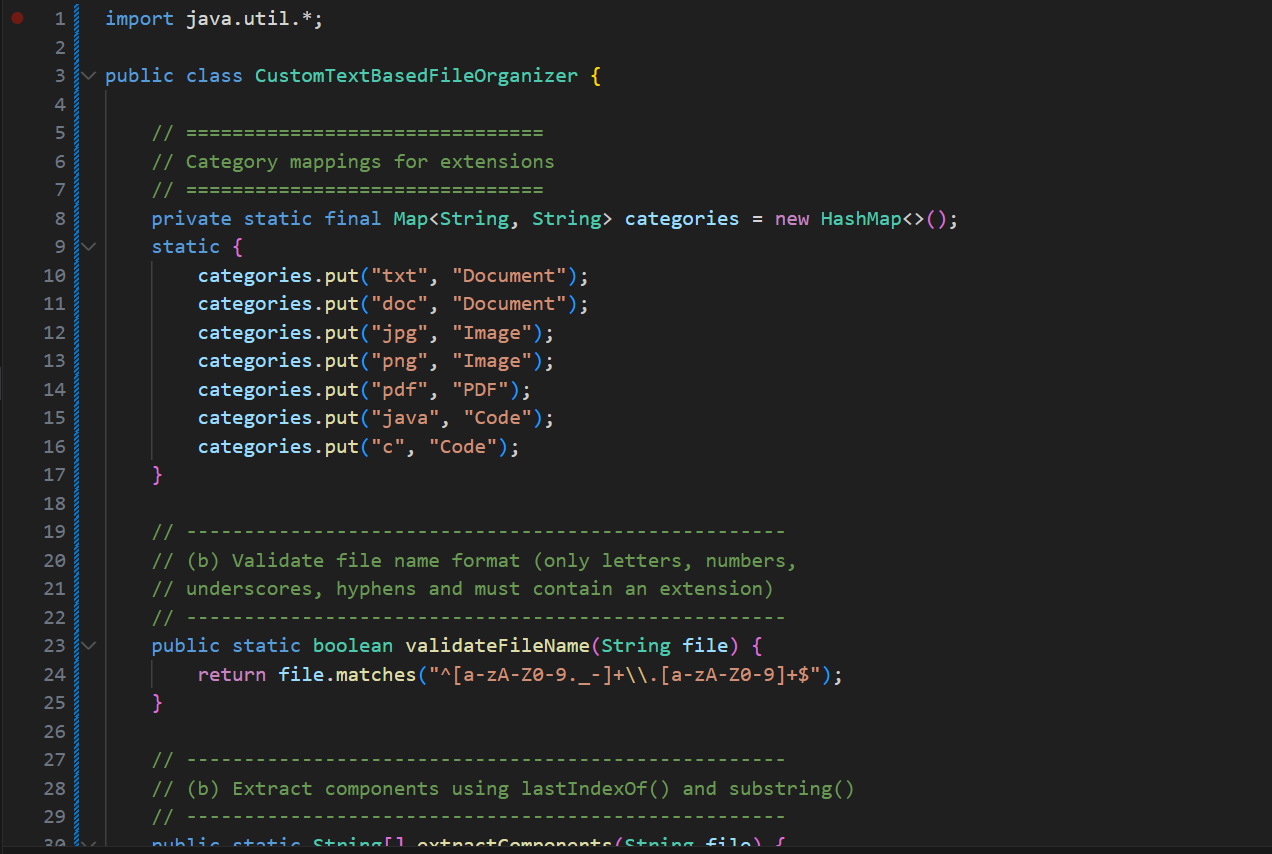
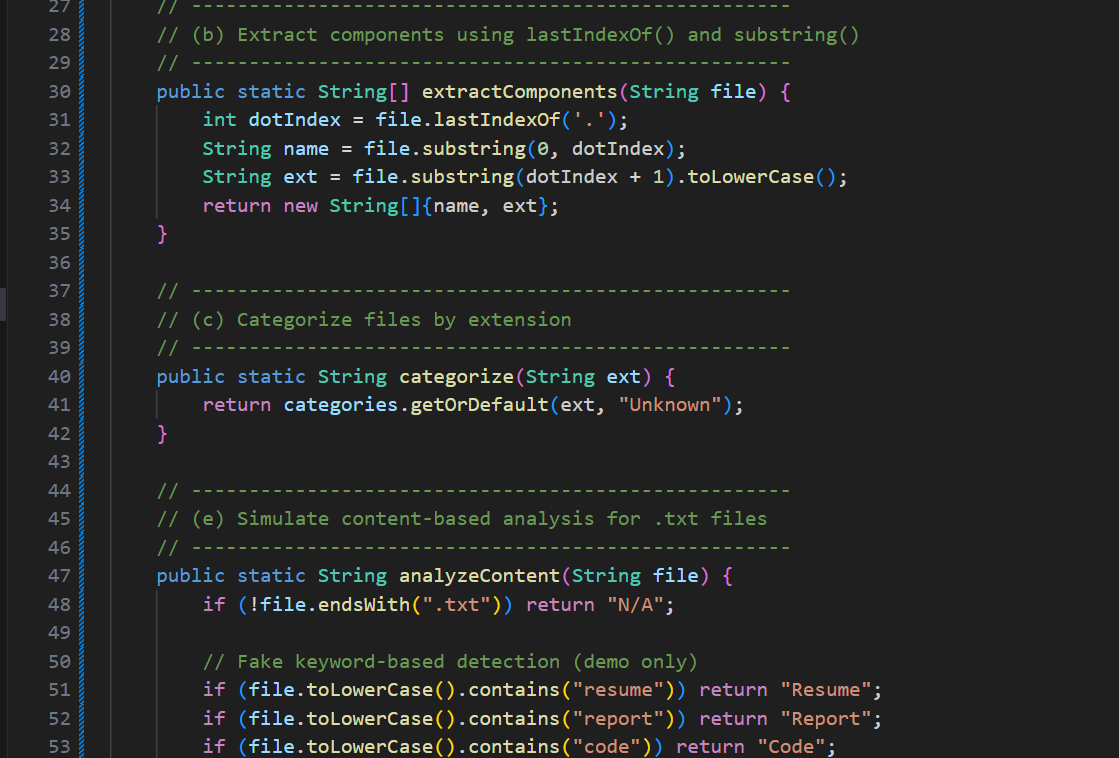
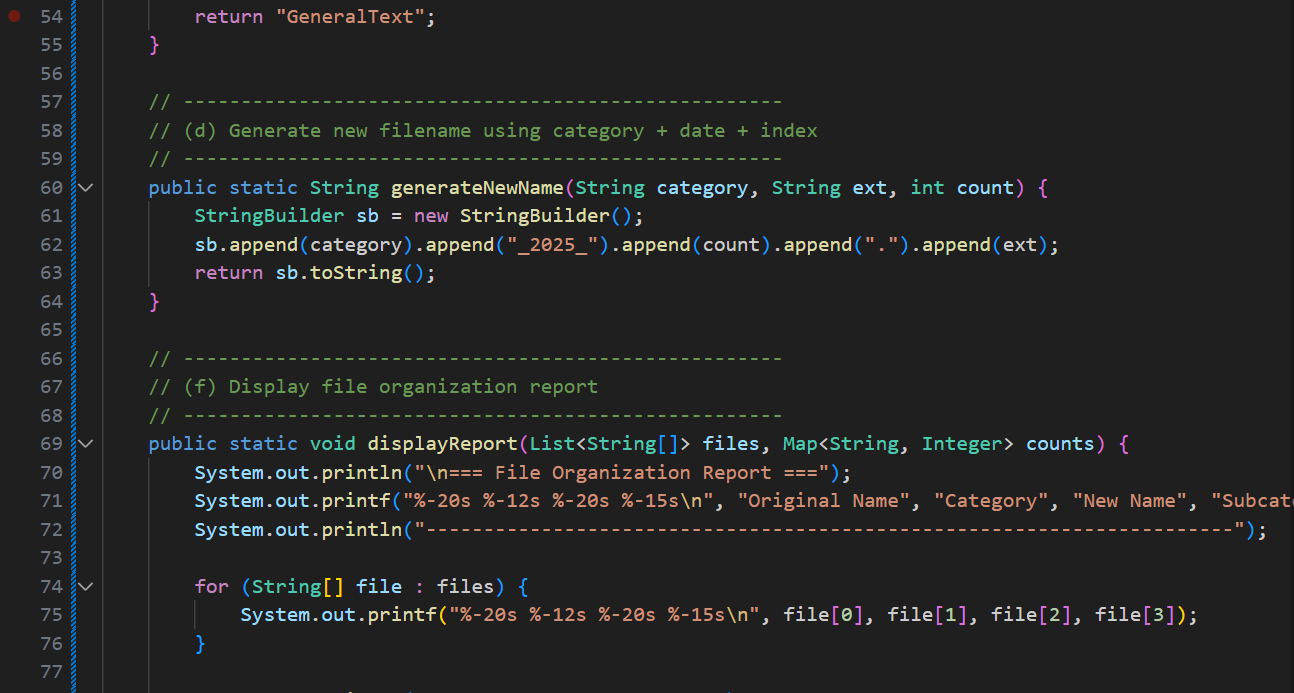
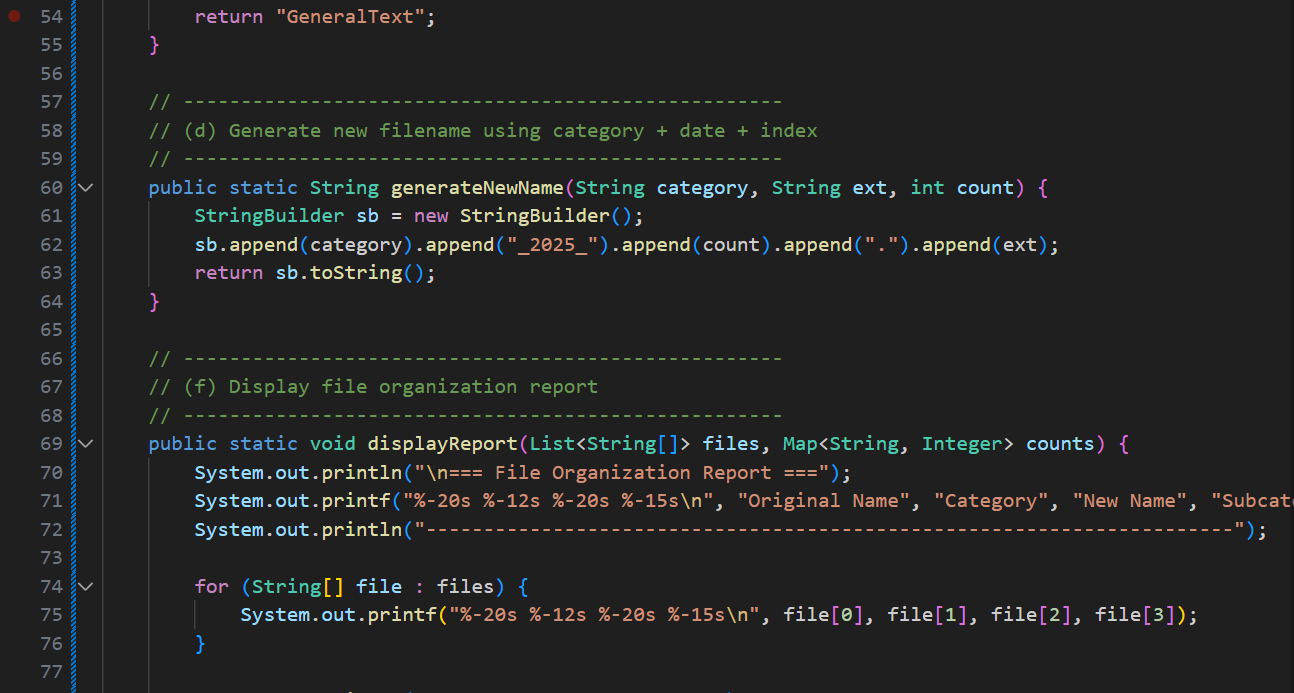
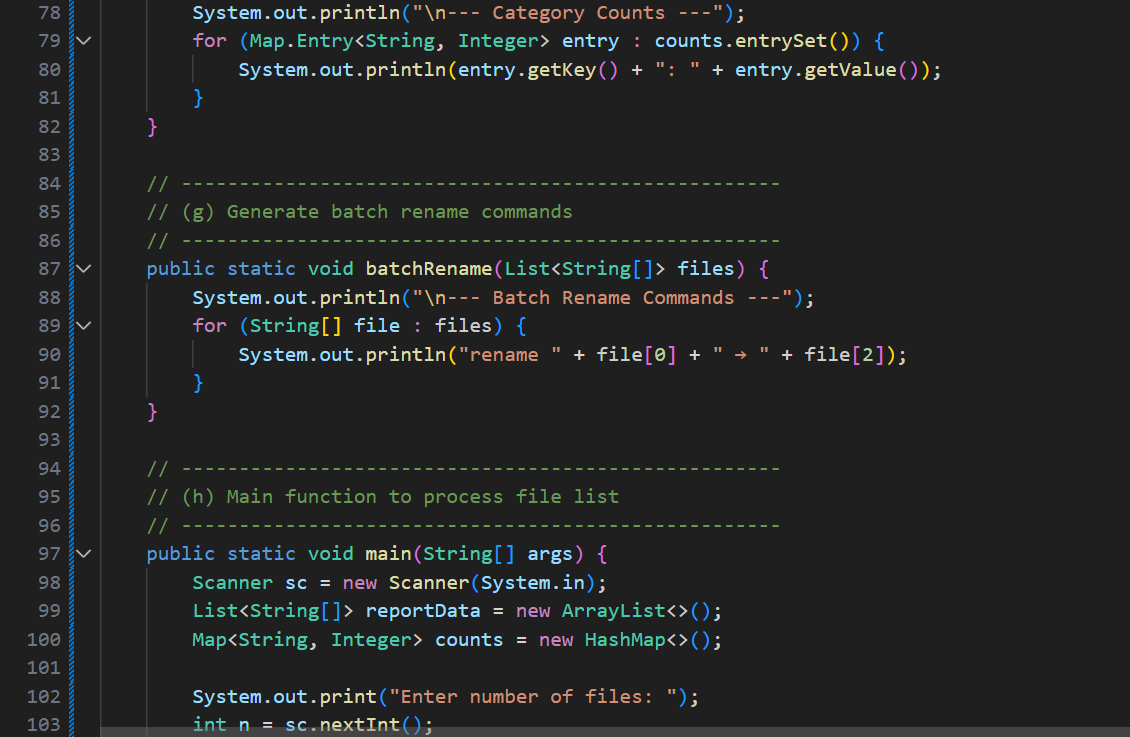
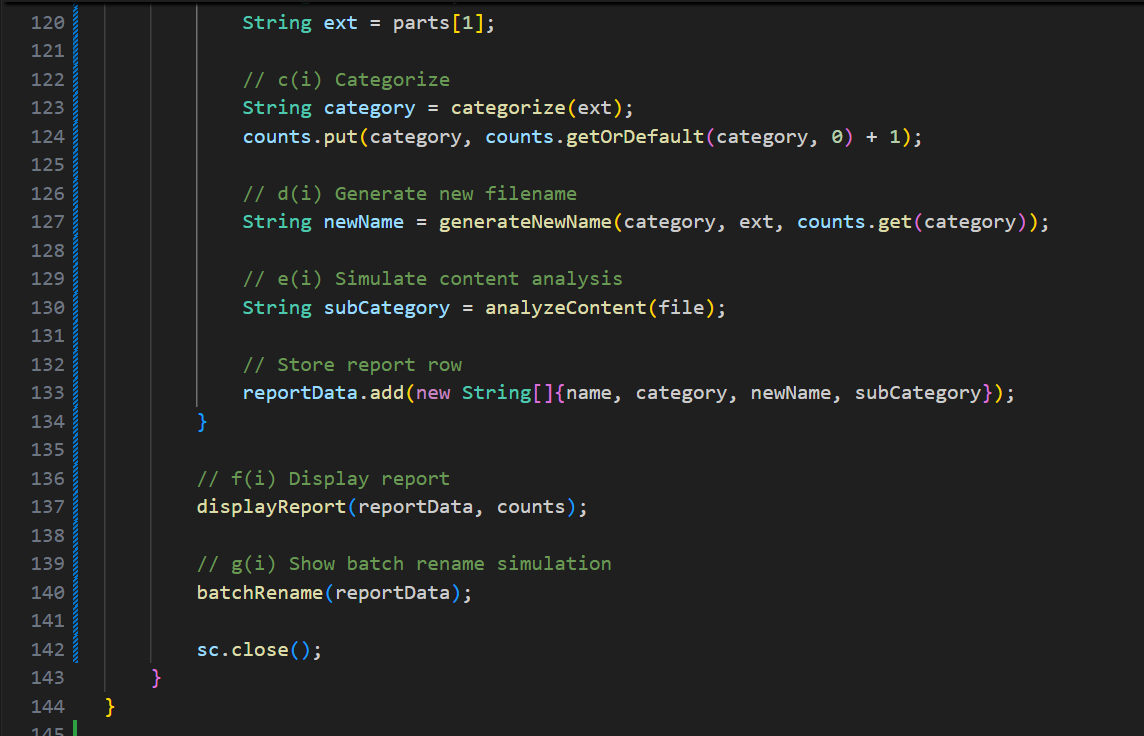
**● i. Create command strings for renaming operations**

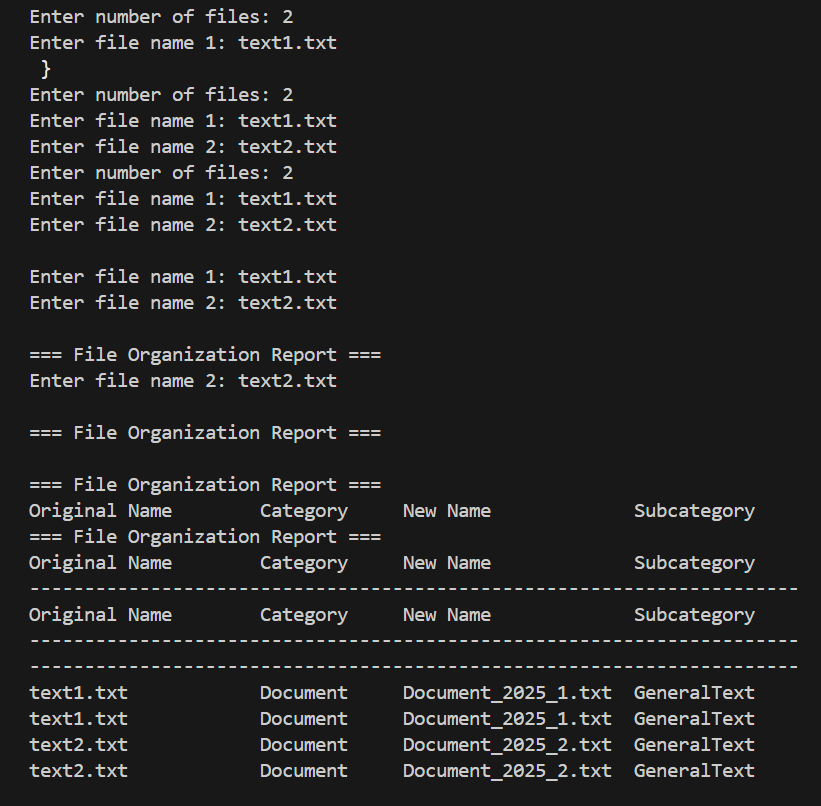
**● ii. Show before/after comparison**

**● iii. Calculate storage organization improvement**

**h. The main function processes file list and generates comprehensive organization plan with**

**statistics**

**** ****    

**OUTPUT🡪** **** 