

Assignment -11 (24-09-2025)

Roll No - 107

Problem 1: Create Customer class with the relevant information like name, address, id, phone

File Edit Selection View Go Run ... Java Assignment 8

EXPLORER

- JAVA ASSIGNMENT 8
 - Animal.class
 - AnimalQ2.java
 - AnonymousQ9.class
 - AnonymousQ9.java
 - ApplianceQ3.java
 - BankingSystem.java
 - Cat.class
 - Dog.class
 - DogCatQ4.class
 - DogCatQ4.java
 - EngineQ5.java
 - GreetingQ8.java
 - ImplementationQ7.java
 - InterfaceQ6.java
 - Main.java
 - PrinterQ10.java
 - Shape.class
 - ShapeCreator.class
 - ShapeCreator\$1.class

BankingSystem.java

```
182 public class BankingSystem {
183     public static void main(String[] args) {
214         Customer customer4 = new Customer(name:"Bob Johnson", address:"789 Pine Rd, Uptown", customerId:"CUST004", phone:
215         System.out.println(x:"Test Case 4: Customer without Account");
216         customer4.display();
217         System.out.println();
218
219
220         System.out.println(x:"Test Case 5: Testing Getters and Setters");
221         Account account5 = new Account();
222         account5.setAccountType(accountType:"Current");
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Java Assignment 8_978aled5\bin' 'BankingSystem'

Banking System Demo

Test Case 1: Normal Customer with Positive Balance

=== Customer Information ===

Customer Name: John Doe

Customer ID: CUST001

Address: 123 Main St, Anytown

Phone Number: +1-555-0123

Account Details:

Account Type: Savings

Account Number: SAV001

Current Balance: \$5000.0

Minimum Balance: \$1000.0

Annual Interest: \$250.00

=====

Test Case 2: Customer with Negative Balance

=== Customer Information ===

Customer Name: Jane Smith

Customer ID: CUST002

Address: 456 Oak Ave, Downtown

Phone Number: +1-555-0456

Account Details:

Account Type: Checking

3:25 PM 9/25/2025

File Edit Selection View Go Run ... Java Assignment 8

EXPLORER

- JAVA ASSIGNMENT 8
 - Animal.class
 - AnimalQ2.java
 - AnonymousQ9.class
 - AnonymousQ9.java
 - ApplianceQ3.java
 - BankingSystem.java
 - Cat.class
 - Dog.class
 - DogCatQ4.class
 - DogCatQ4.java
 - EngineQ5.java
 - GreetingQ8.java
 - ImplementationQ7.java
 - InterfaceQ6.java
 - Main.java
 - PrinterQ10.java
 - Shape.class
 - ShapeCreator.class
 - ShapeCreator\$1.class

BankingSystem.java

```
182 public class BankingSystem {
183     public static void main(String[] args) {
214         Customer customer4 = new Customer(name:"Bob Johnson", address:"789 Pine Rd, Uptown", customerId:"CUST004", phone:
215         System.out.println(x:"Test Case 4: Customer without Account");
216         customer4.display();
217         System.out.println();
218
219
220         System.out.println(x:"Test Case 5: Testing Getters and Setters");
221         Account account5 = new Account();
222         account5.setAccountType(accountType:"Current");
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Account Details:

Account Type: Checking

Account Number: CHK002

Current Balance: \$-500.0

Minimum Balance: \$0.0

Interest Calculation Error: Cannot calculate interest on negative balance: -500.0

=====

Test Case 3: Direct Exception Handling

Exception caught: Cannot calculate interest on negative balance: -1000.0

Test Case 4: Customer without Account

=== Customer Information ===

Customer Name: Bob Johnson

Customer ID: CUST004

Address: 789 Pine Rd, Uptown

Phone Number: +1-555-0789

No account associated with this customer.

=====

Test Case 5: Testing Getters and Setters

=== Customer Information ===

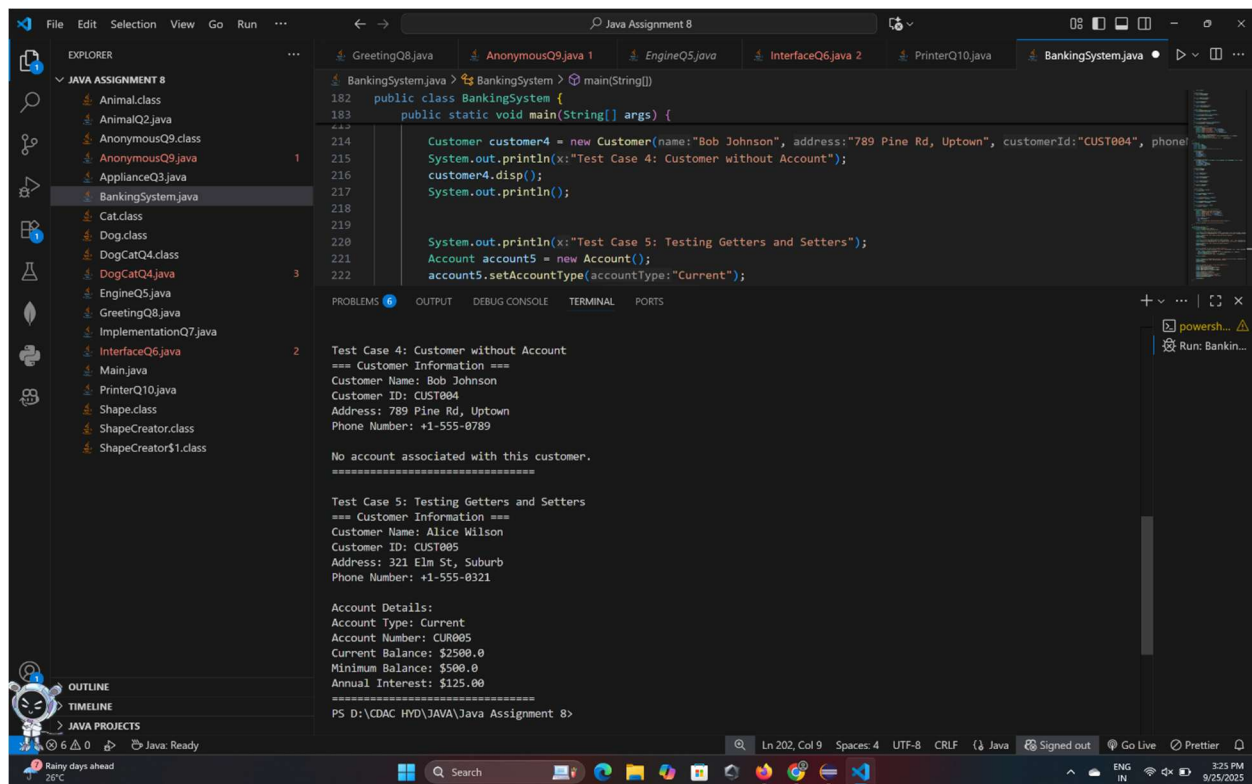
Customer Name: Alice Wilson

Customer ID: CUST005

Address: 321 Elm St, Suburb

Phone Number: +1-555-0321

3:25 PM 9/25/2025



Java IO:

1. Write a Program to read the same program file and find the no. of lines, words and characters. Write the result in in to a text file (result.txt)

The screenshot shows the Eclipse IDE with the `FileAnalyzer.java` file open. The code is as follows:

```

129     writer.println("- " + wordCount + " words total.");
130     writer.println("- " + characterCount + " characters total.");
131
132     System.out.println("Results successfully written to: " + resultFileName);
133
134     catch (IOException e) {
135         System.err.println("Error writing to result file: " + e.getMessage());
136     } finally {
137         if (writer != null) {
138             writer.close();
139         }
140     }
141
142     private static void displayResults() {
143         System.out.println();
144         System.out.println("ANALYSIS RESULTS:");
145         System.out.println("-----");
146         System.out.println("File analyzed: " + sourceFileName);
147         System.out.println("Lines: " + lineCount);
148         System.out.println("Words: " + wordCount);
149         System.out.println("Characters: " + characterCount);
150         System.out.println();
151         System.out.printf("Average words per line: %.2f\n", (double) wordCount / lineCount);
152         System.out.printf("Average characters per line: %.2f\n", (double) characterCount / lineCount);
153         System.out.printf("Average characters per word: %.2f\n", (double) characterCount / wordCount);
154         System.out.println();
155         System.out.println("Check '" + resultFileName + "' for detailed results.");
156     }
157
158 }
159
160
161

```

The console output shows the following results:

```

File Analyzer Program
=====
Results successfully written to: result.txt

ANALYSIS RESULTS:
-----
File analyzed: FileAnalyzer.java
Lines: 161
Words: 575
Characters: 6108

Average words per line: 3.57
Average characters per line: 37.94
Average characters per word: 10.62

Check 'result.txt' for detailed results.

```

2. Write a program to read the same program file and write it to other file with the lines number added before each line, starting from 1.

The screenshot shows the Eclipse IDE with the `LineNumberAdder.java` file open. The code is as follows:

```

138     int totalLines = 0;
139
140     try (BufferedReader reader = new BufferedReader(new FileReader(SOURCE_FILE));
141          PrintWriter writer = new PrintWriter(new FileWriter(OUTPUT_FILE))) {
142
143         writeHeader(writer);
144
145         String line;
146         while ((line = reader.readLine()) != null) {
147
148             writer.printf("%3d: %s\n", lineNumber, line);
149             lineNumber++;
150             totalLines++;
151         }
152
153         writeFooter(writer, totalLines);
154         System.out.println("Total lines processed: " + totalLines);
155         return SUCCESS;
156     } catch (FileNotFoundException e) {
157         System.err.println("Error: Source file '" + SOURCE_FILE + "' not found.");
158         return FAILURE;
159     } catch (IOException e) {
160         System.err.println("Error during file operation: " + e.getMessage());
161         return FAILURE;
162     }
163
164 }
165
166
167
168
169
170

```

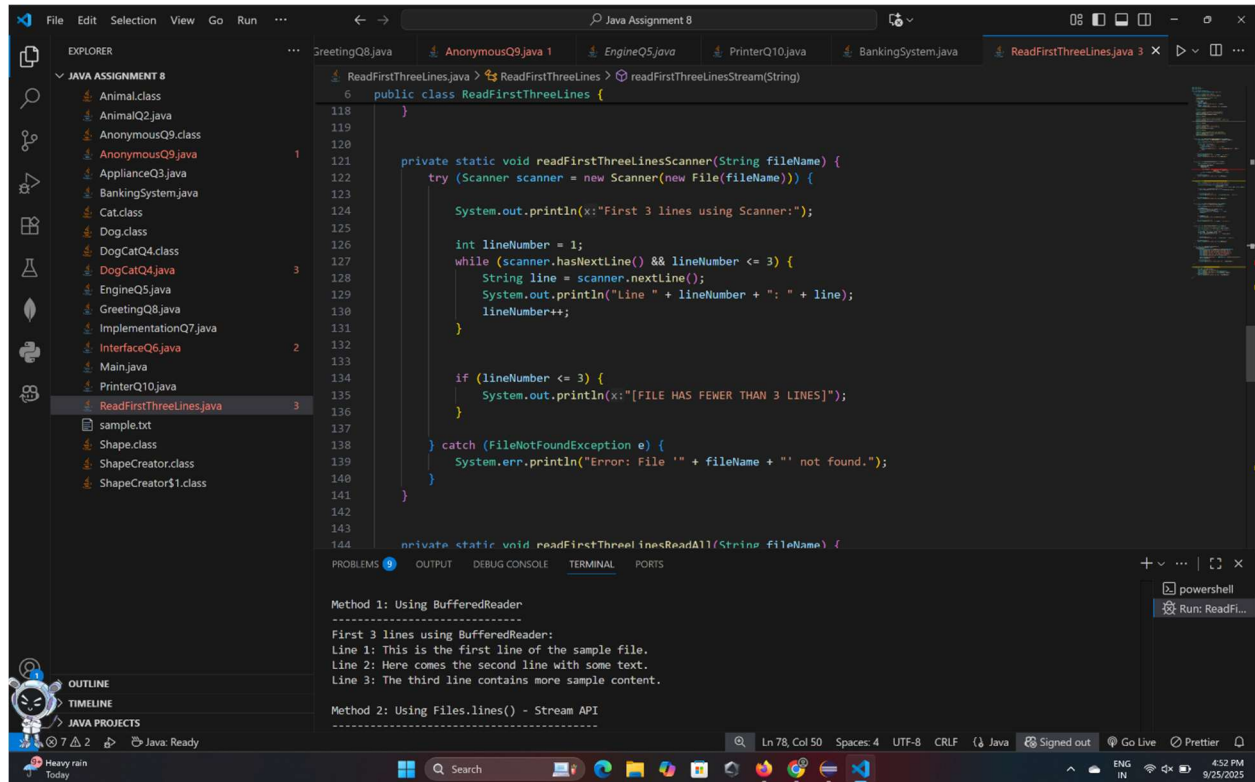
The console output shows the following results:

```

Processed 9 lines...
Processed 19 lines...
Processed 29 lines...
Processed 39 lines...
Processed 49 lines...
Processed 59 lines...
Processed 69 lines...
Processed 79 lines...
Processed 89 lines...
Processed 99 lines...
Processed 109 lines...
Processed 119 lines...
Processed 129 lines...
Processed 139 lines...
Processed 149 lines...
Processed 159 lines...
Processed 169 lines...
Total lines processed: 169
SUCCESS: Line numbers added successfully!
Check the file 'LineNumberedProgram.txt' for the result.

```

3. Write a Java program to read first 3 lines from a file.



The screenshot shows an IDE with a project named "Java Assignment 8". The Explorer panel on the left lists several Java files, with "ReadFirstThreeLines.java" selected. The main editor displays the code for this class. The code defines a public class "ReadFirstThreeLines" with two methods: "readFirstThreeLinesScanner" and "readFirstThreeLinesReadAll". The "readFirstThreeLinesScanner" method uses a "Scanner" to read the first three lines of a file. It prints the first three lines using "System.out.println". The "readFirstThreeLinesReadAll" method is currently empty. The Output panel at the bottom shows the results of running the program, displaying the first three lines of the file "sample.txt".

```
118 }
119
120
121 private static void readFirstThreeLinesScanner(String fileName) {
122     try (Scanner scanner = new Scanner(new File(fileName))) {
123
124         System.out.println("First 3 lines using Scanner:");
125
126         int lineNumber = 1;
127         while (scanner.hasNextLine() && lineNumber <= 3) {
128             String line = scanner.nextLine();
129             System.out.println("Line " + lineNumber + ": " + line);
130             lineNumber++;
131         }
132
133         if (lineNumber <= 3) {
134             System.out.println("FILE HAS FEWER THAN 3 LINES");
135         }
136     } catch (FileNotFoundException e) {
137         System.err.println("Error: File " + fileName + " not found.");
138     }
139 }
140
141 private static void readFirstThreeLinesReadAll(String fileName) {
142
143
144 }
```

Method 1: Using BufferedReader

First 3 lines using BufferedReader:
Line 1: This is the first line of the sample file.
Line 2: Here comes the second line with some text.
Line 3: The third line contains more sample content.

Method 2: Using Files.lines() - Stream API

4. Write a Java program to find the longest word in a text file

File Edit Selection View Go Run ... Java Assignment 8

EXPLORER

- Java Assignment 8
 - Animal.class
 - AnimalQ2.java
 - AnonymousQ9.java
 - ApplianceQ3.java
 - BankingSystem.java
 - Cat.class
 - Dog.class
 - DogCatQ4.class
 - DogCatQ4.java
 - EngineQ5.java
 - GreetingQ8.java
 - ImplementationQ7.java
 - InterfaceQ6.java
 - LongestWordFinder.java
 - Main.java
 - PrinterQ10.java
 - ReadFirstThreeLines.java
 - sample_text.txt
 - Shape.class
 - ShapeCreator.class
 - ShapeCreator\$1.class

LongestWordFinder.java

```
6 public class LongestWordFinder {
289     private static void createSampleFileIfNotExists() {
293         try {
294             PrintWriter writer = new PrintWriter(new FileWriter(DEFAULT_FILE));
295             writer.println("The quick brown fox jumps over the lazy dog.");
296             writer.println("Java programming is extraordinarily powerful and versatile.");
297             writer.println("Supercalifragilisticexpialidocious is a very long word!");
298             writer.println("Short words: a, an, the, is, in, on, at, to, of, for.");
299             writer.println("Programming languages include: Java, Python, JavaScript, C++.");
300             writer.println("This file contains various words of different lengths.");
301             writer.println("Some technical terms: algorithm, implementation, optimization.");
302             writer.println("Antidisestablishmentarianism is another extremely long word.");
303
304             System.out.println("Sample file '" + DEFAULT_FILE + "' created for demonstration.");
305             System.out.println();
306         } catch (IOException e) {
307             e.printStackTrace();
308         }
309     }
310 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Sample file 'sample_text.txt' created for demonstration.

Analyzing default file: sample_text.txt

Method 1: Using BufferedReader

Longest word: "supercalifragilisticexpialidocious"

Length: 34 characters

Total words analyzed: 61

Method 2: Using Stream API

Longest word: "supercalifragilisticexpialidocious"

Length: 34 characters

Method 3: Using Scanner

Longest word: "supercalifragilisticexpialidocious"

Length: 34 characters

Total words analyzed: 61

Method 4: Find All Longest Words

Maximum word length: 34 characters

Ln 288, Col 5 Spaces: 4 UTF-8 CRLF Java Signed out Go Live Prettier

Top Stories Bar... of Boty...

File Edit Selection View Go Run ... Java Assignment 8

EXPLORER

- Java Assignment 8
 - Animal.class
 - AnimalQ2.java
 - AnonymousQ9.java
 - ApplianceQ3.java
 - BankingSystem.java
 - Cat.class
 - Dog.class
 - DogCatQ4.class
 - DogCatQ4.java
 - EngineQ5.java
 - GreetingQ8.java
 - ImplementationQ7.java
 - InterfaceQ6.java
 - LongestWordFinder.java
 - Main.java
 - PrinterQ10.java
 - ReadFirstThreeLines.java
 - sample_text.txt
 - Shape.class
 - ShapeCreator.class
 - ShapeCreator\$1.class

LongestWordFinder.java

```
6 public class LongestWordFinder {
289     private static void createSampleFileIfNotExists() {
293         try {
294             PrintWriter writer = new PrintWriter(new FileWriter(DEFAULT_FILE));
295             writer.println("The quick brown fox jumps over the lazy dog.");
296             writer.println("Java programming is extraordinarily powerful and versatile.");
297             writer.println("Supercalifragilisticexpialidocious is a very long word!");
298             writer.println("Short words: a, an, the, is, in, on, at, to, of, for.");
299             writer.println("Programming languages include: Java, Python, JavaScript, C++.");
300             writer.println("This file contains various words of different lengths.");
301             writer.println("Some technical terms: algorithm, implementation, optimization.");
302             writer.println("Antidisestablishmentarianism is another extremely long word.");
303
304             System.out.println("Sample file '" + DEFAULT_FILE + "' created for demonstration.");
305             System.out.println();
306         } catch (IOException e) {
307             e.printStackTrace();
308         }
309     }
310 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Total words analyzed: 61

Method 4: Find All Longest Words

Maximum word length: 34 characters

Longest word: "supercalifragilisticexpialidocious"

Method 5: Detailed Word Analysis

=== DETAILED WORD ANALYSIS ===

Total words: 61

Unique words: 49

Shortest word: "a" (1 chars)

Longest word: "supercalifragilisticexpialidocious" (34 chars)

Average word length: 6.03 characters

=== WORD LENGTH DISTRIBUTION ===

1 chars:	3 words (4.9%)
2 chars:	11 words (18.0%)
3 chars:	7 words (11.5%)
4 chars:	12 words (19.7%)
5 chars:	7 words (11.5%)
6 chars:	1 words (1.6%)
7 chars:	4 words (6.6%)

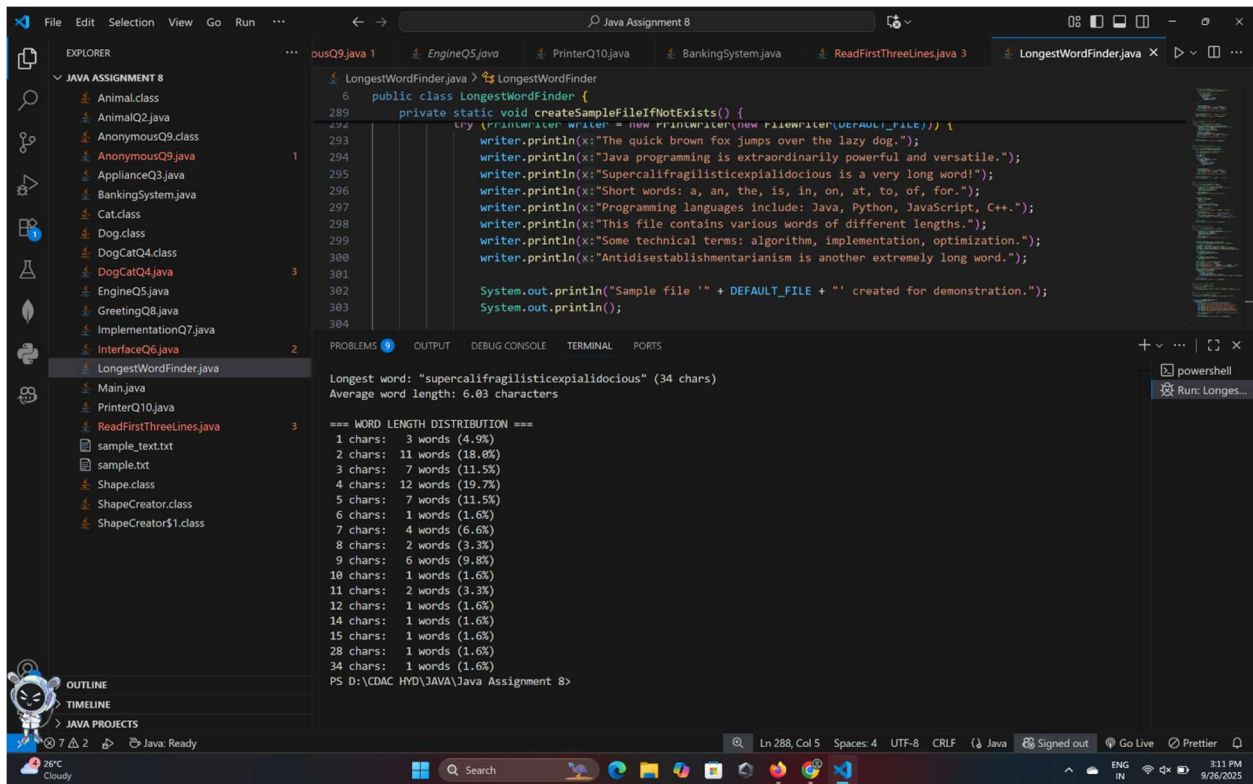
OUTLINE

TIMELINE

JAVA PROJECTS

Ln 288, Col 5 Spaces: 4 UTF-8 CRLF Java Signed out Go Live Prettier

26°C Cloudy



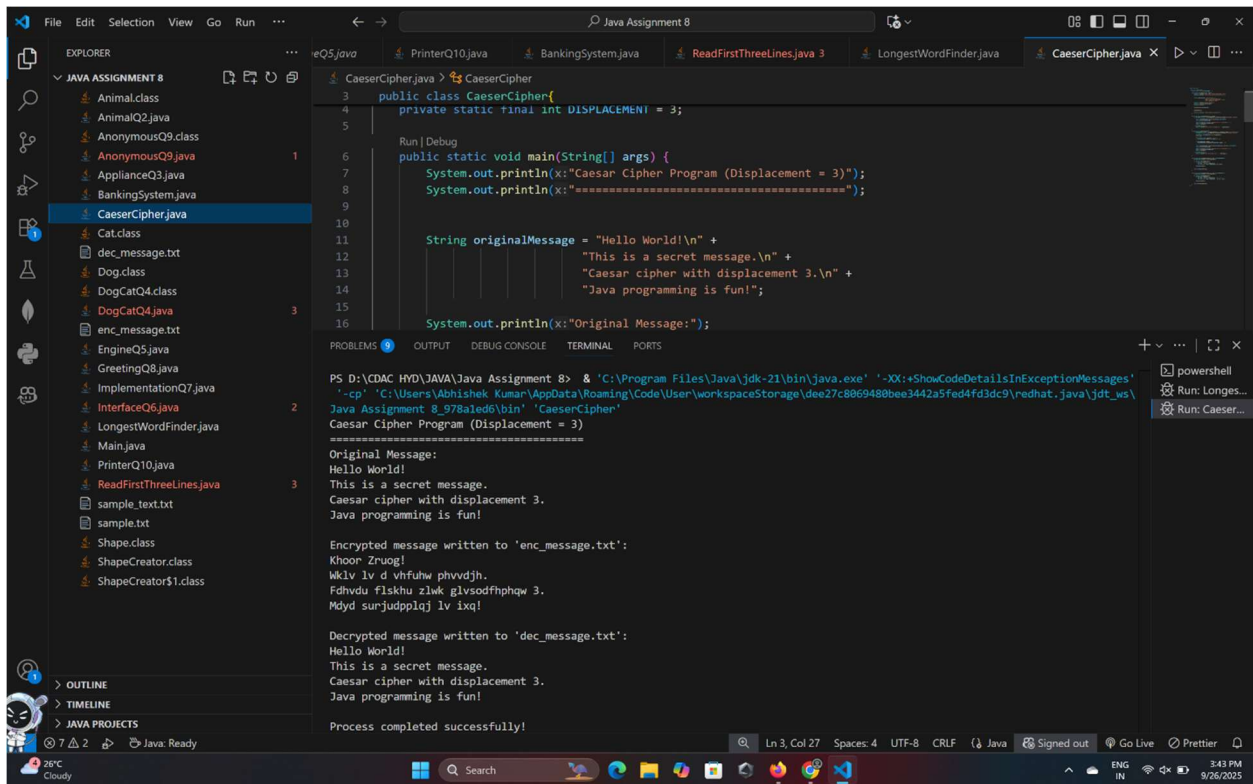
5. Write a program to implement Caesar cipher using files.

Write to the file (enc_message.txt) with using caesar cipher with

the displacement value = 3.

Read the file (enc_message.txt) and decode the Cipher text and write it into

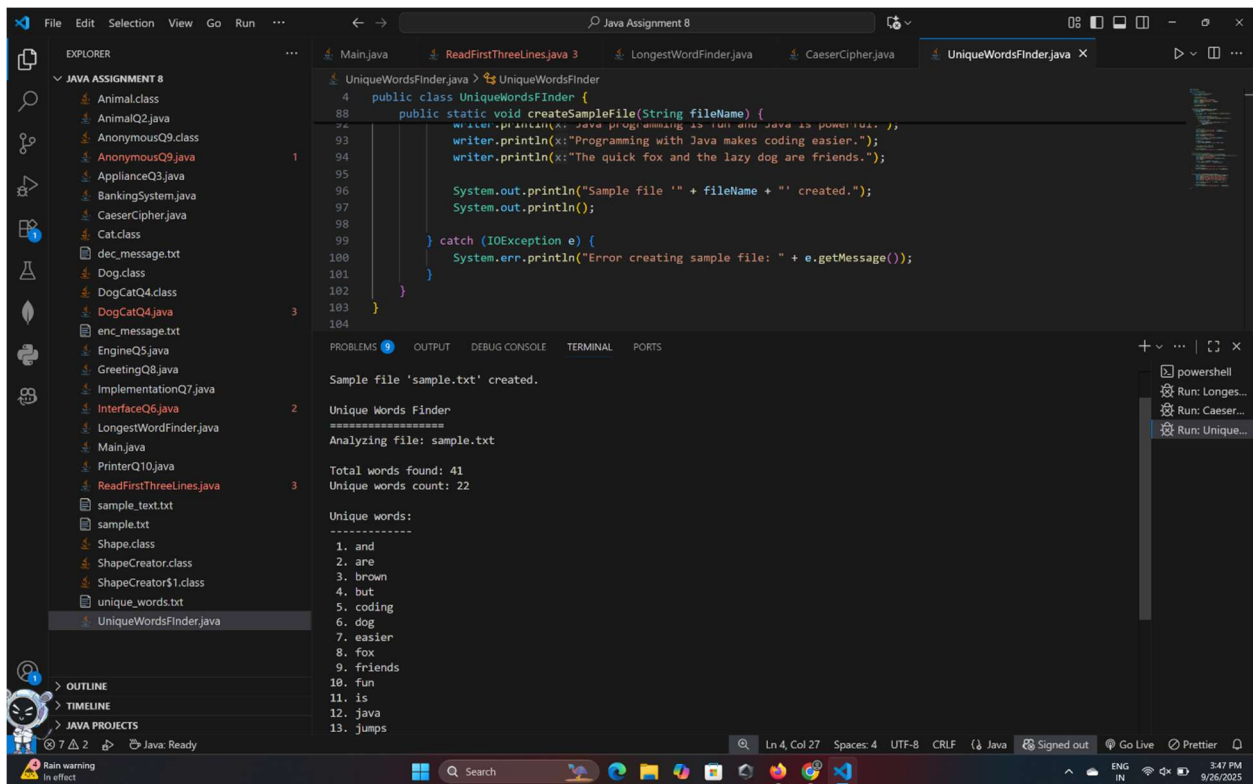
a file (dec_message.txt)



```
CaesarCipher.java > CaesarCipher
3 public class CaesarCipher {
4     private static final int DISPLACEMENT = 3;
5
6     Run | Debug
7     public static void main(String[] args) {
8         System.out.println(x:"Caesar Cipher Program (Displacement = 3)");
9         System.out.println(x:"=====");
10
11         String originalMessage = "Hello World!\n" +
12             "This is a secret message.\n" +
13             "Caesar cipher with displacement 3.\n" +
14             "Java programming is fun!";
15
16         System.out.println(x:"Original Message:");
17     }
18 }
```

PS D:\CDAC HYD\JAVA\Java Assignment 8> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' -cp 'C:\Users\Abhishek Kumar\AppData\Roaming\Code\User\workspaceStorage\dee27c886948bbee3442a5fed4fd3dc9\redhat.java\jdt_ws\Java Assignment 8_978baed6\bin' 'CaesarCipher' Caesar Cipher Program (Displacement = 3) ===== Original Message: Hello World! This is a secret message. Caesar cipher with displacement 3. Java programming is fun! Encrypted message written to 'enc_message.txt': KhooH Zruog! WkIv Iv d vhfuhw phvvdjh. Fdhvdu flskhu zIwK glvsodfhphqw 3. Mdyd surjudpplqj Iv ixq! Decrypted message written to 'dec_message.txt': Hello World! This is a secret message. Caesar cipher with displacement 3. Java programming is fun! Process completed successfully!

6. Write a program to find unique words in file



```
Main.java > ReadFirstThreeLines.java 3 > LongestWordFinder.java > CaesarCipher.java > UniqueWordsFinder.java
4 UniqueWordsFinder.java > UniqueWordsFinder
4 public class UniqueWordsFinder {
88     public static void createSampleFile(String fileName) {
89         try {
90             FileWriter writer = new FileWriter(fileName);
91             writer.println(x:"Programming with Java is powerful.");
92             writer.println(x:"The quick fox and the lazy dog are friends.");
93
94             System.out.println("Sample file '" + fileName + "' created.");
95             System.out.println();
96
97         } catch (IOException e) {
98             System.err.println("Error creating sample file: " + e.getMessage());
99         }
100     }
101
102     public static void findUniqueWords(String fileName) {
103         try {
104             FileReader reader = new FileReader(fileName);
105             Scanner scanner = new Scanner(reader);
106             Set<String> uniqueWords = new HashSet<>();
107             while (scanner.hasNextLine()) {
108                 String line = scanner.nextLine();
109                 String[] words = line.split("\\s+");
110                 for (String word : words) {
111                     uniqueWords.add(word.toLowerCase());
112                 }
113             }
114             scanner.close();
115
116             System.out.println("Total words found: " + uniqueWords.size());
117             System.out.println("Unique words count: " + uniqueWords.size());
118
119             System.out.println("Unique words:");
120             for (String word : uniqueWords) {
121                 System.out.println(word);
122             }
123         } catch (IOException e) {
124             System.err.println("Error reading sample file: " + e.getMessage());
125         }
126     }
127 }
```

Sample file 'sample.txt' created.

Unique Words Finder

Analyzing file: sample.txt

Total words found: 41

Unique words count: 22

Unique words:

- 1. and
- 2. are
- 3. brown
- 4. but
- 5. coding
- 6. dog
- 7. easier
- 8. fox
- 9. friends
- 10. fun
- 11. is
- 12. java
- 13. jumps

7. Write a program to find duplicate words in a file

File Edit Selection View Go Run ... Java Assignment 8

EXPLORER

- JAVA ASSIGNMENT 8
 - Animal.class
 - AnimalQ2.java
 - AnonymousQ9.class
 - AnonymousQ9.java
 - ApplianceQ3.java
 - BankingSystem.java
 - CaesarCipher.java
 - Cat.class
 - dec_message.txt
 - Dog.class
 - DogCatQ4.class
 - DogCatQ4.java
 - duplicate_words.txt
 - DuplicateWordsFinder.java
 - enc_message.txt
 - EngineQ5.java
 - GreetingQ8.java
 - ImplementationQ7.java
 - InterfaceQ6.java
 - LongestWordFinder.java
 - Main.java
 - PrinterQ10.java
 - ReadFirstThreeLines.java
 - sample_text.txt
 - sample.txt
 - Shape.class
 - ShapeCreator.class
 - ShapeCreator\$1.class
 - unique_words.txt
 - UniqueWordsFinder.java

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

=====

Analyzing file: sample.txt

Total words: 57
Unique words: 24
Duplicate words: 12

Duplicate words (sorted by frequency):

1. the	(appears 9 times)
2. and	(appears 5 times)
3. dog	(appears 4 times)
4. fox	(appears 4 times)
5. is	(appears 4 times)
6. lazy	(appears 4 times)
7. quick	(appears 4 times)
8. java	(appears 3 times)
9. brown	(appears 2 times)
10. fun	(appears 2 times)
11. programming	(appears 2 times)
12. was	(appears 2 times)

Results saved to 'duplicate_words.txt'

PS D:\CDAC HYD\JAVA\Java Assignment 8>

Ln 75, Col 13 Spaces: 4 UTF-8 CRLF Java Signed out Go Live Prettier

26°C Cloudy 3:51 PM 9/26/2023