**ASSIGNMENT – 3**

**Subject: CSW2 (CSE 2141)**

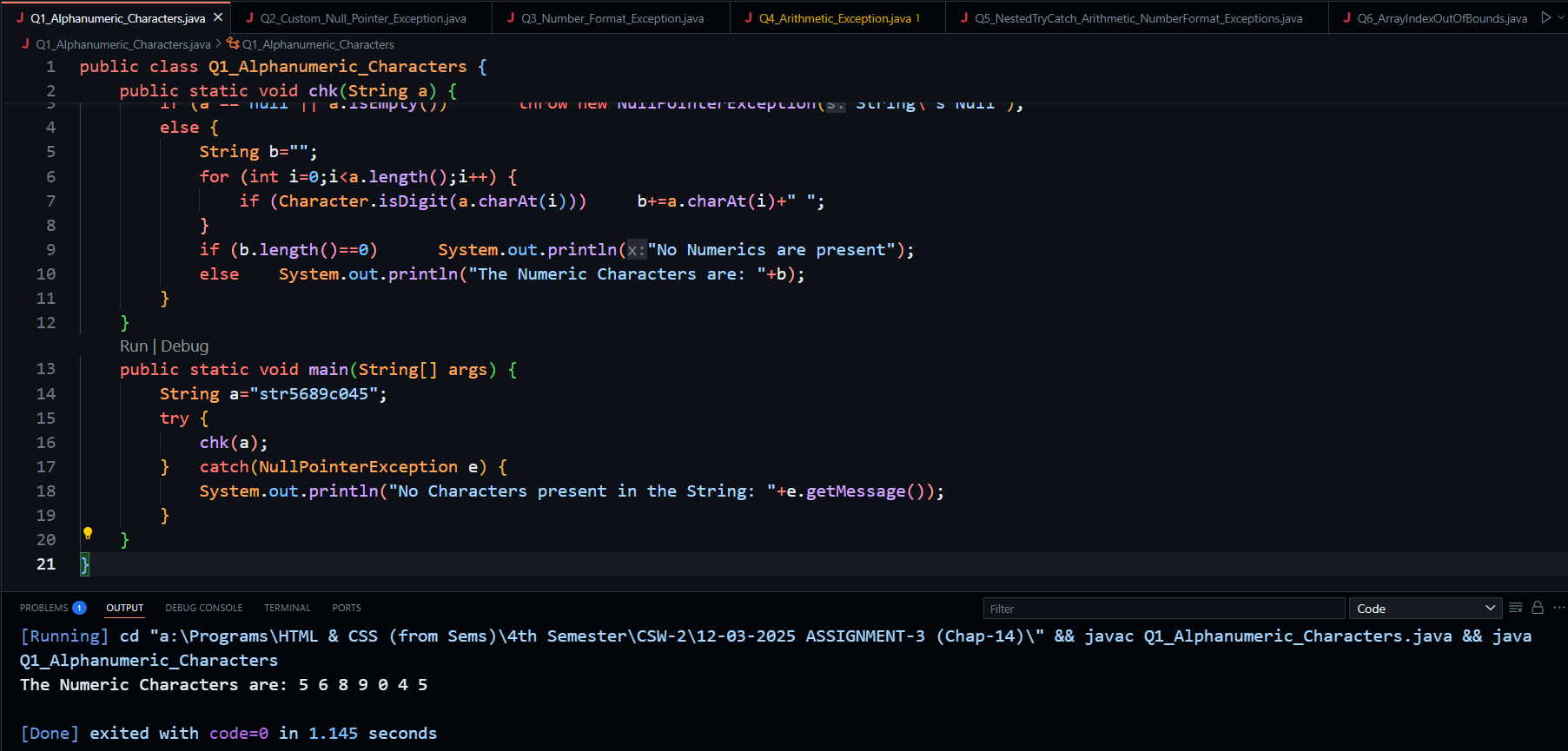
**Name: Arpit Kumar Mohanty**

**Registration Number: 2341013237**

**Section: 23412G1**

**Branch: CSE**

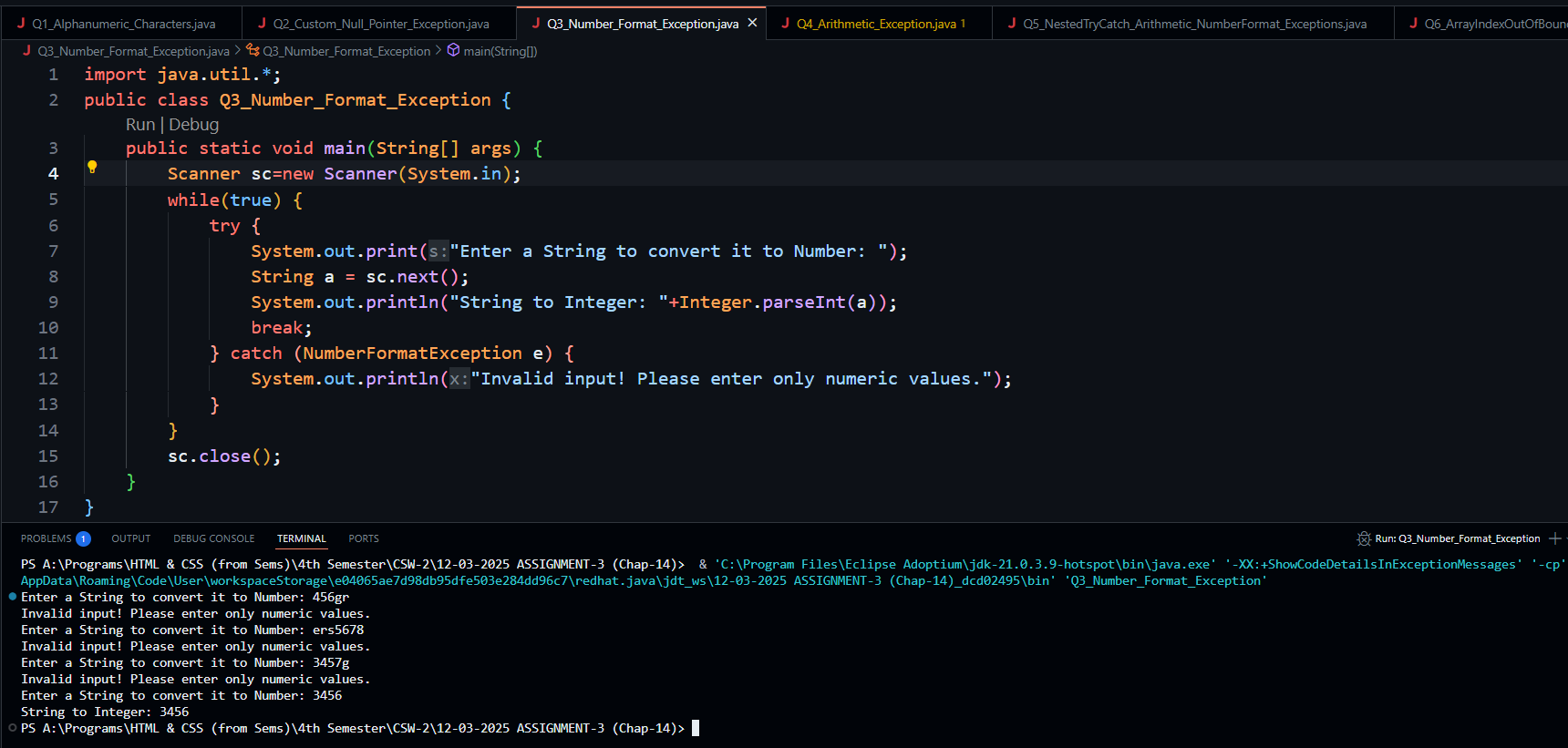
**Q1. You are given a string containing alphanumeric characters, and your task is to design a Java program that extracts and displays the numeric characters from the given string. If no numeric characters are present, the program should display an appropriate message indicating their absence. Additionally, if the input string is null or empty, the program must throw a NullPointerException with a meaningful error message.**

**Solution along with Output:  
**

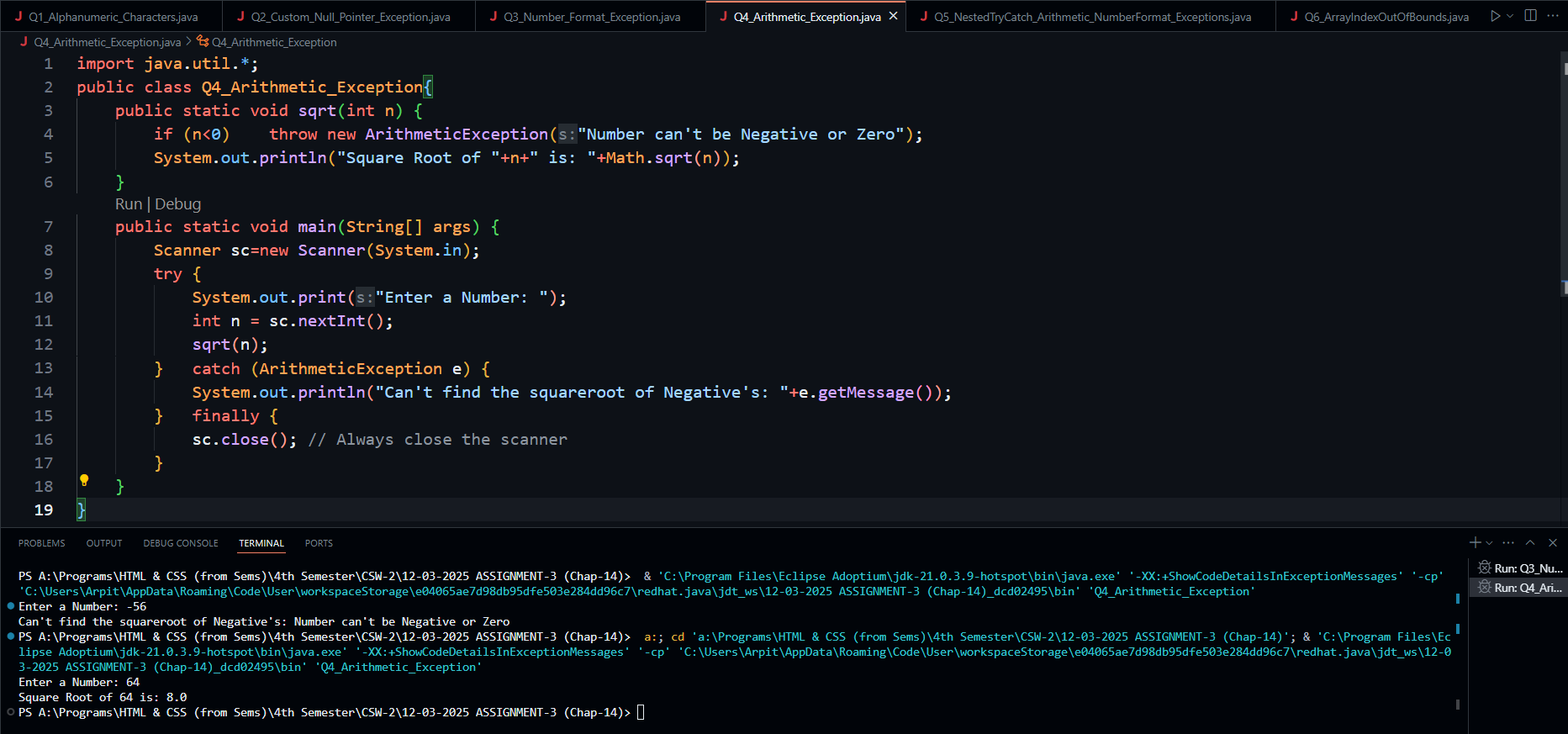
**Q2. Implement a custom exception class named CustomNullPointerException that replicates the behavior of the standard NullPointerException. However, instead of relying on default error messages or null references, this custom exception should accept a String message as a constructor argument. Your task is to create this custom exception class and demonstrate its usage in a Java program.**

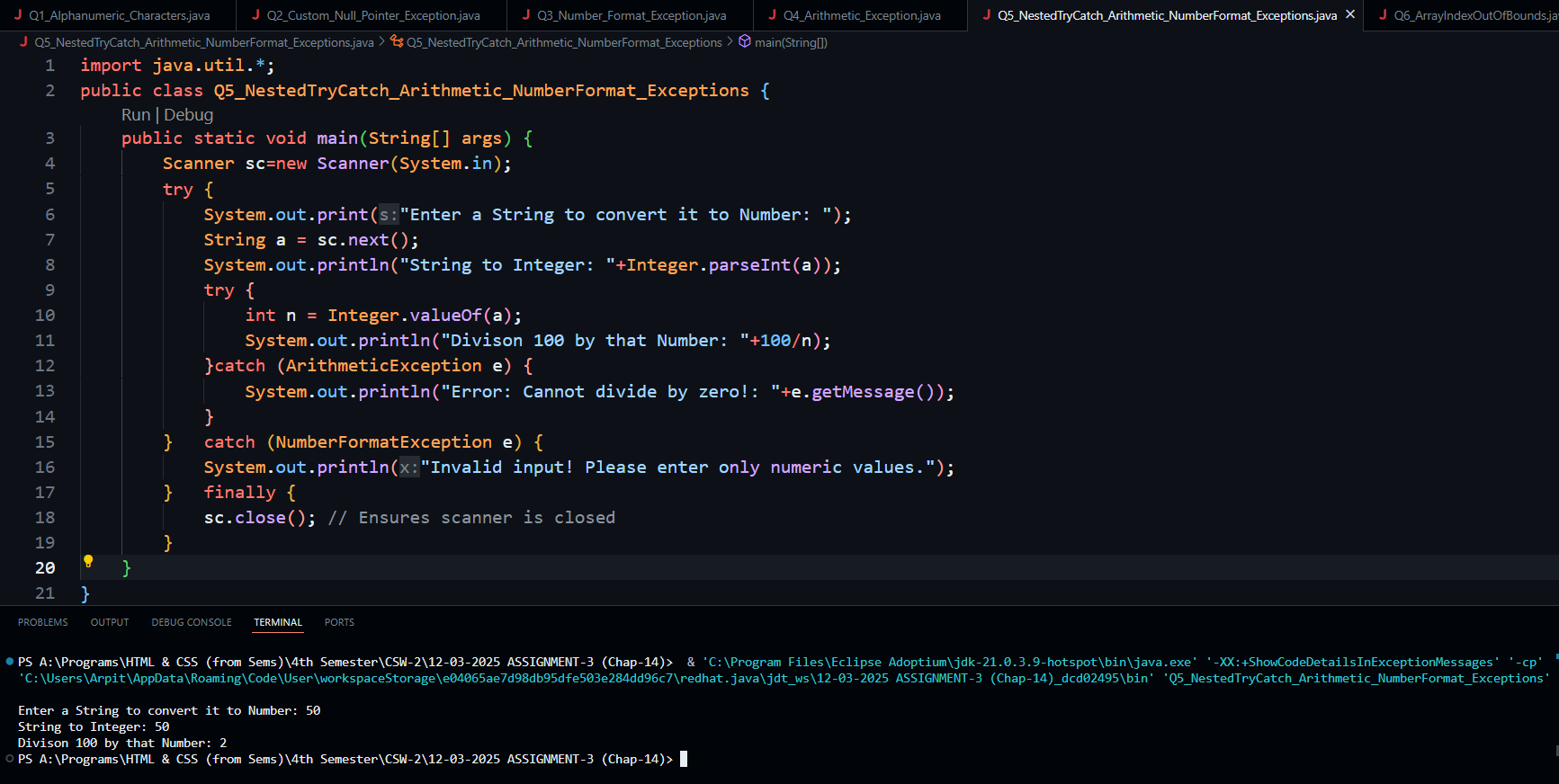
**Solution along with Output:   
**

**Q3. Create a method that accepts a string input and converts it into an integer. Use a trycatch block to handle NumberFormatException, and if an exception occurs, prompt the user to enter a valid numeric value.**

**Solution along with Output:   
**

**Q4. Write a Java program to find the square root of an integer number. Demonstrate the use of a try-catch block to handle ArithmeticException.  
  
Solution along with Output:**

**  
  
Q5. Demonstrate the use of a nested try-catch block. Write a Java program where the outer try-catch block handles a NumberFormatException, while the inner try-catch block handles an ArithmeticException.**

**Solution along with Output: **

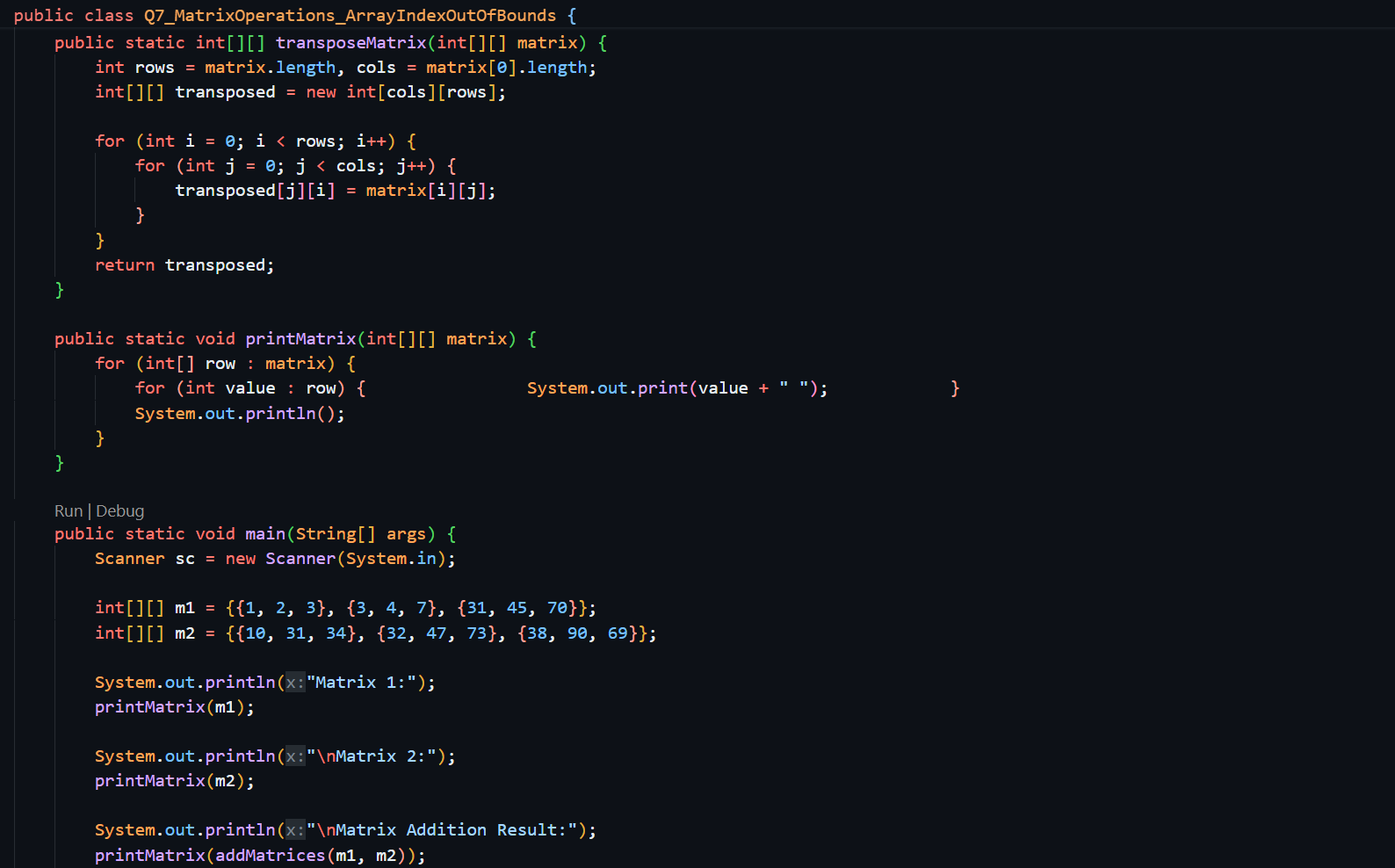
**Q6. Implement a Java program that performs complex manipulations on an array of integers, including operations such as sorting, searching, and accessing elements at various indices. Introduce scenarios where accessing elements beyond the array bounds leads to an ArrayIndexOutOfBoundsException. Handle these exceptions gracefully to ensure the program continues execution without crashing.  
  
Solution along with Output:**

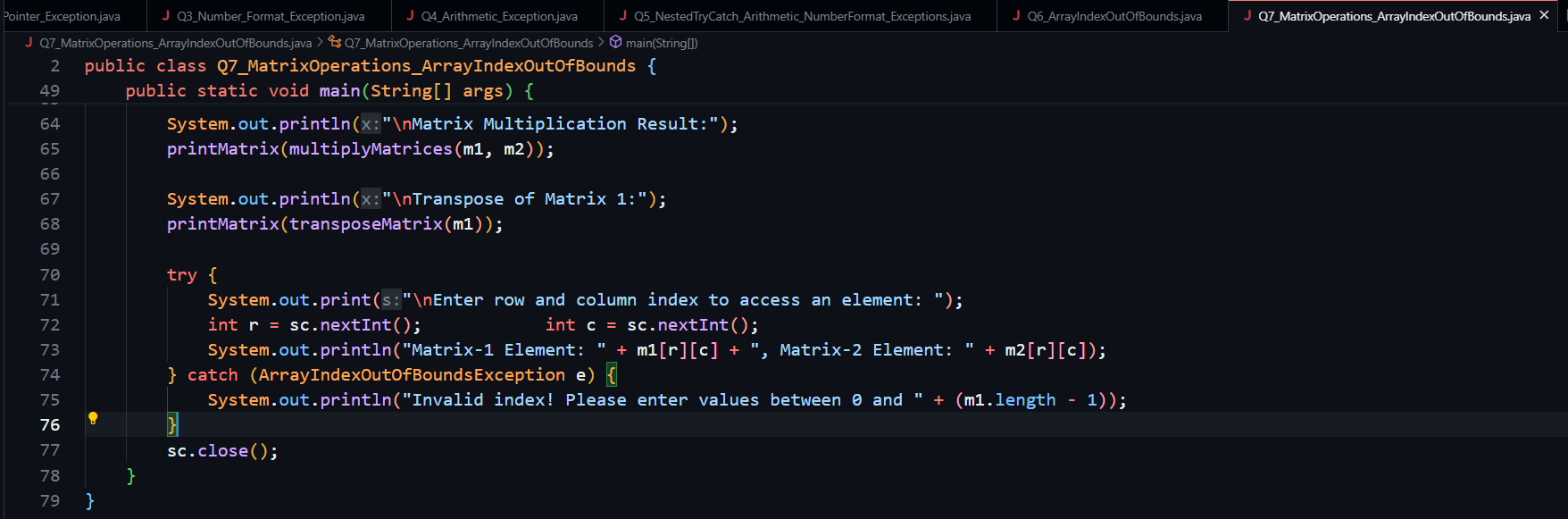
****

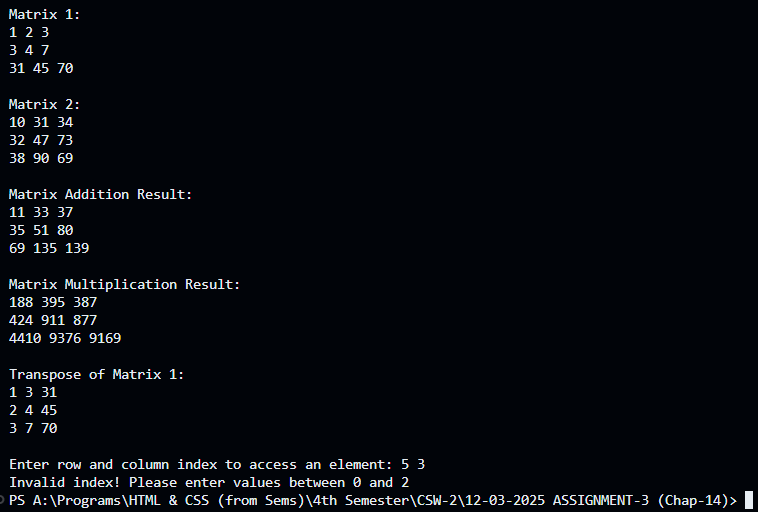
**  
  
Q7. Design a Java program to perform matrix operations such as addition, multiplication, and transpose. Introduce scenarios where accessing elements beyond the matrix bounds results in an ArrayIndexOutOfBoundsException. Handle these exceptions effectively and provide meaningful error messages that clearly indicate the nature of the exception**

**Solution along with Output:**

****

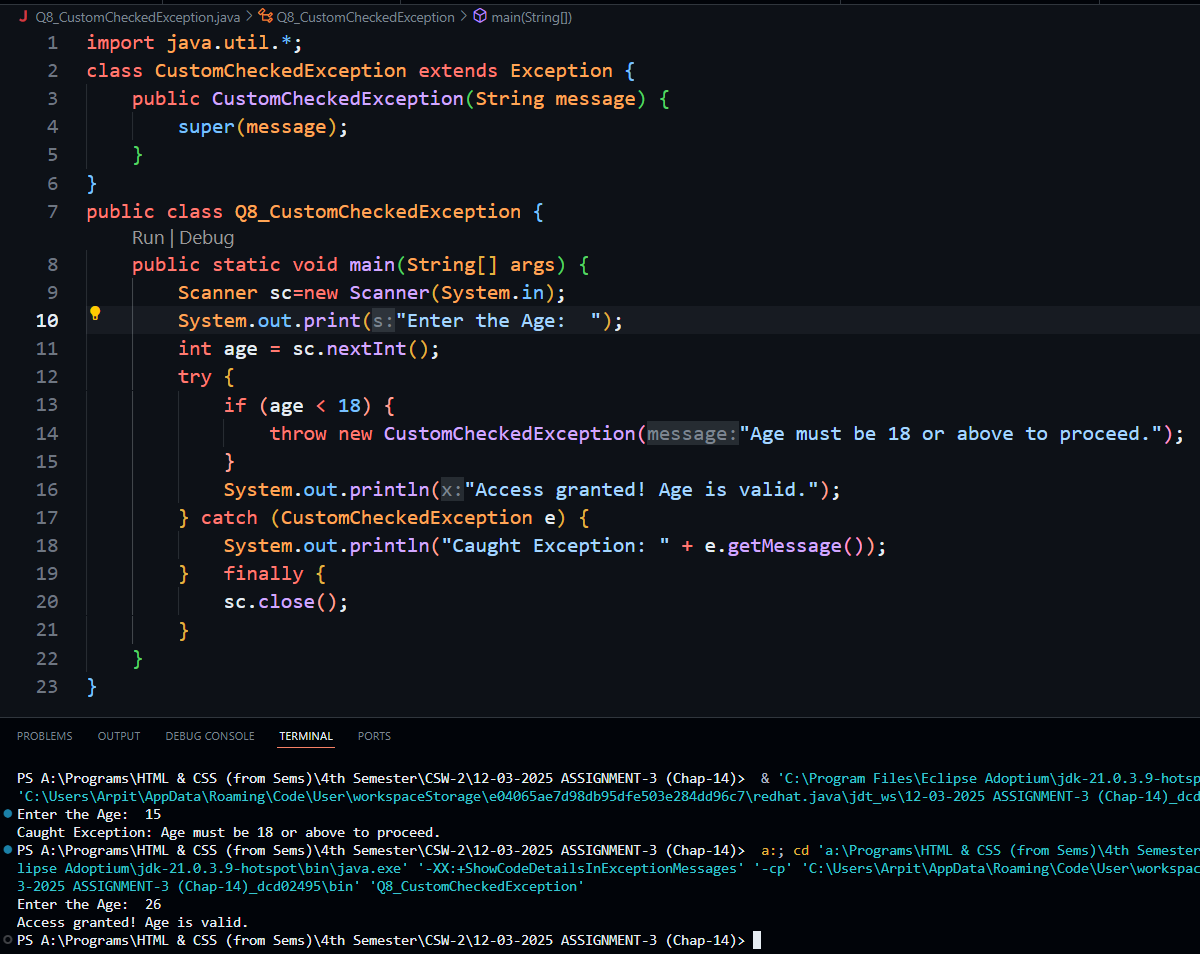
****

****

****

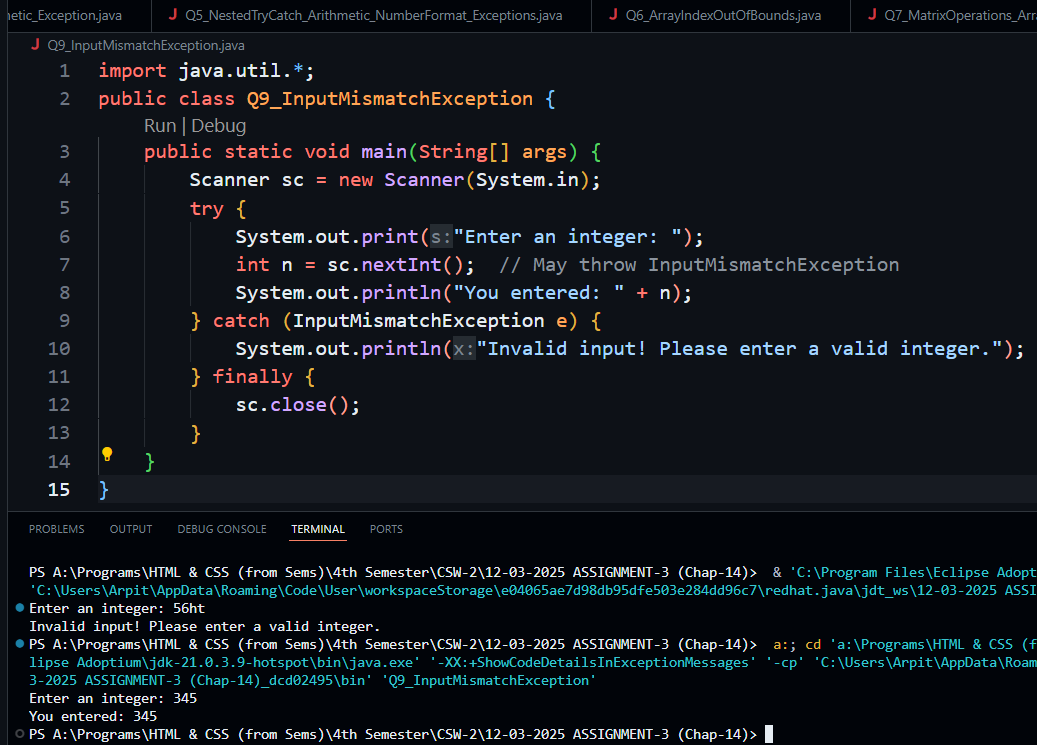
**Q8. Create a custom-checked exception class named CustomCheckedException. Use this exception in your program to handle a specific error condition and demonstrate its usage with a try-catch block.**

**Solution along with Output:**

****

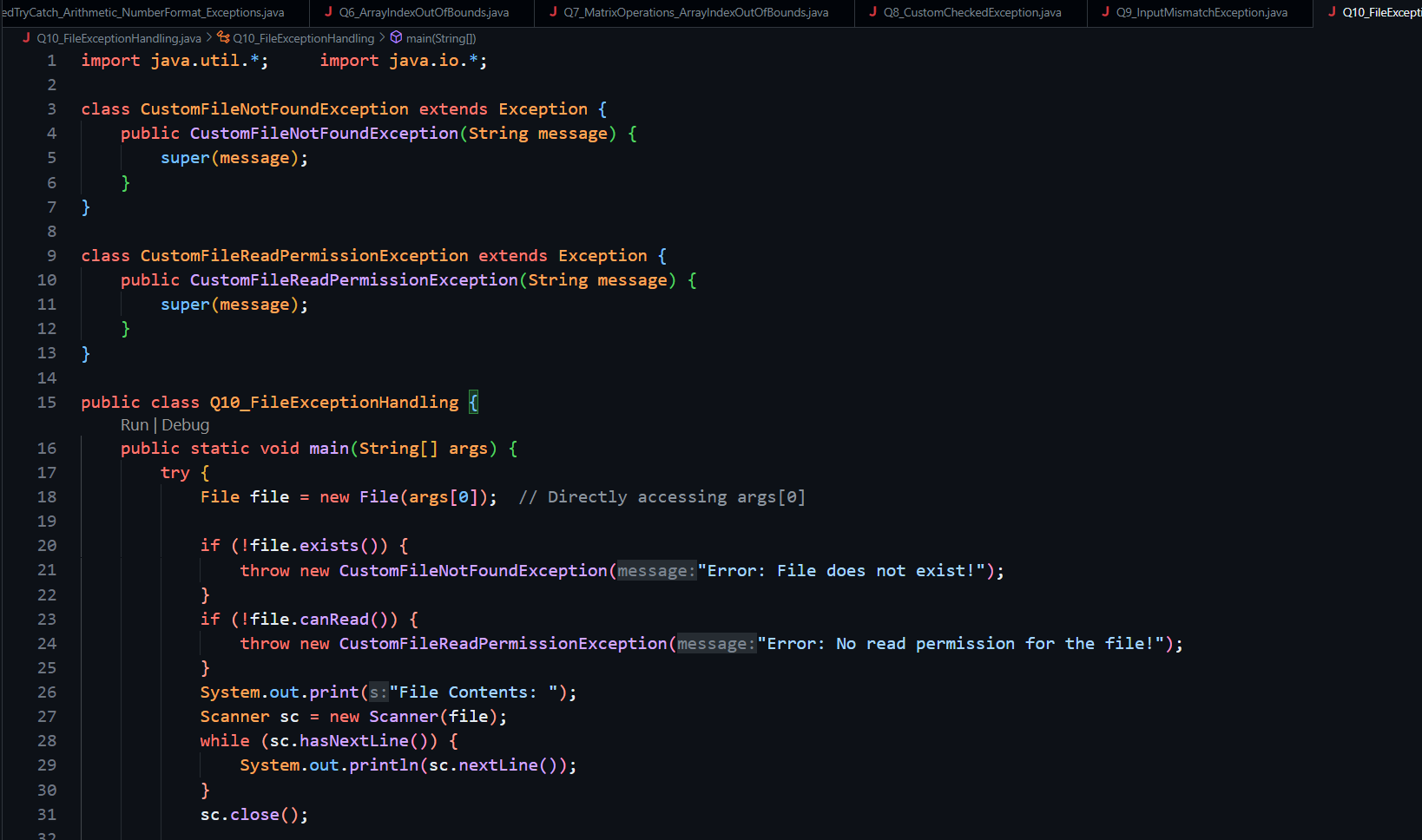
**Q9. . Implement a method that reads an integer from the user and handles InputMismatchException using a try-catch block.**

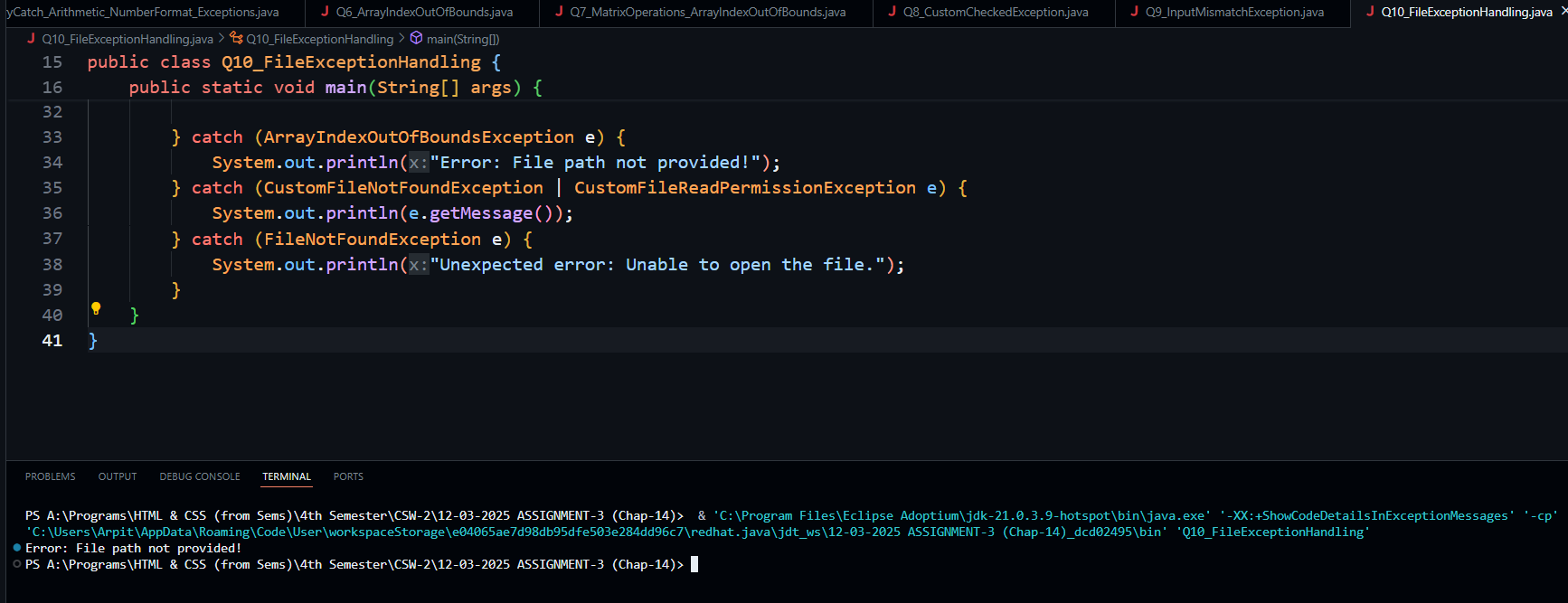
**Solution along with Output:**

****

**Q10. Implement a Java program that reads a file path from the command-line argument and attempts to read its contents. If the file path is null or points to a non-existent file, throw a custom FileNotFoundException. If the file exists but cannot be read due to permission issues, throw a custom FileReadPermissionException. Your task is to create these custom exception classes and handle them appropriately in your program.**

**Solution along with Output:**

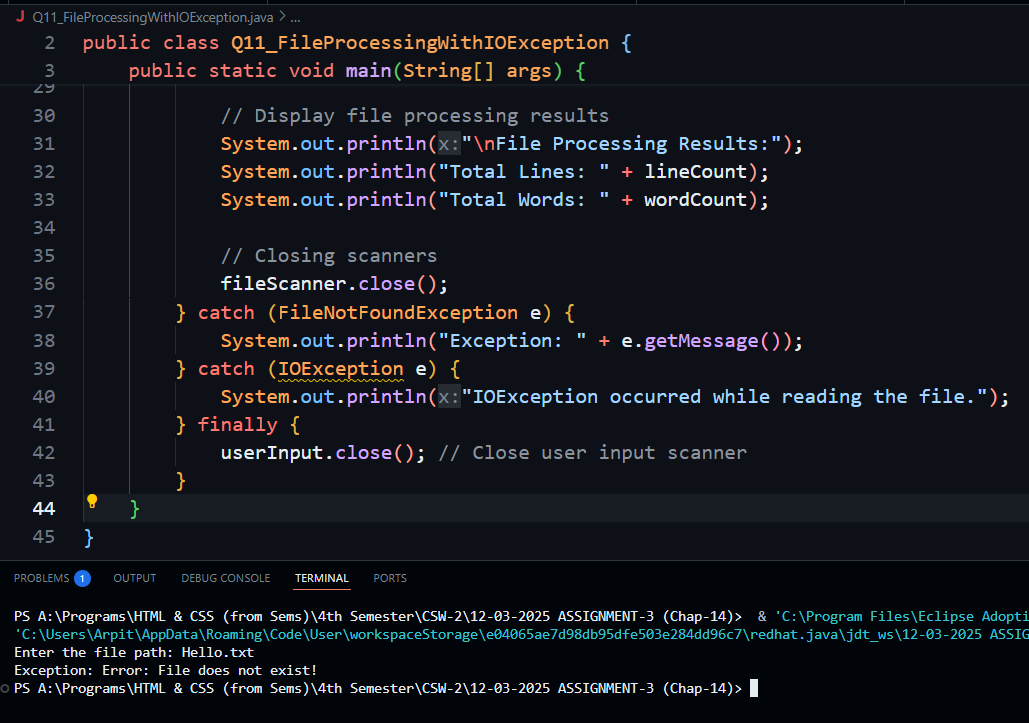
****

****

**Q11. Write a program that reads data from a file and performs some processing. Handle checked IOException by using try-catch block to catch and handle the exception.**

**Solution along with Output:**

****

****