UpCurve PE 2024

assignment for day 7

Assignment #1

Handle user input

Write a Java application to accept integers in a loop. After each number is accepted, the user should be asked if he/she wishes to continue. If the user inputs "NO", then the loop should be stopped and following output should be displayed:

```
Number of inputs = X
Number of integer inputs = Y
Number of non-integer inputs = Z
Sum of all integer inputs = XX
The integer inputs = N1, N2, N3, ...
The non-integer inputs = ASD, SDF, DFG, ...
X, Y, Z, etc should be actual values, based on the inputs.
```

HINT:

- Use exception handling mechanism
- Use `java.util.Scanner` for accepting data from the user.

  ```
  Scanner s = new Scanner(System.in);
  String input = s.nextLine();
  int n = s.nextInt();
  double d = s.nextDouble();
  // ... s
  ```

Assignment #2

Sure, I'll adjust the Java assignment to use arrays instead of lists. Here's the revised version:

Text File Analyzer

Objective: Write a Java program that reads a text file, analyzes its contents, and performs various operations on arrays, implements error handling, and defines methods to accomplish the tasks.

Requirements:

1. **Array Operations:**

    - Read the contents of the text file into an array of strings. Each line of the file should be an element in the array.
    - Implement a method to find the longest and shortest lines in the file.
    - Implement a method to count the number of words in each line and store them in an array of integers.
    - Implement a method to sort the array of word count in descending order.
2. **Error Handling:**
    - Handle errors that may occur during file reading and other operations.
    - Display appropriate error messages if any operation fails.
3. **Text File Handling:**
    - Open and read the contents of a text file specified by the user.
    - Close the file after reading.
4. **Methods:**
    - Define methods for each of the tasks mentioned above (e.g., `readFile`, `findLongestLine`, `findShortestLine`, `countWords`, `sortWordCount`).

Additional Instructions:

- The program should take the filename as input from the user.
- Display the contents of the longest and shortest lines along with their line numbers.
- Display the word count for each line.
- Display the sorted word count.
- Ensure the program is well-documented with comments explaining the purpose of each method and major blocks of code.
- Test your program with different text files of varying lengths and contents to ensure it works correctly under various scenarios.

Sample Output:

```
Enter the filename: example.txt

Contents of the file:
This is a sample text file.
It contains multiple lines with different lengths.
Each line will be analyzed by your app.

Longest line:
Line 2: It contains multiple lines with different lengths.

Shortest line:
Line 1: This is a sample text file.

Word count for each line:
Line 1: 6 words
Line 2: 7 words
```

```
Line 3: 8 words

Sorted word count:
8 words
7 words
6 words
```

Function signature reference:

```
import java.io.FileNotFoundException;

public class TextFileAnalyzer {

    // Method to find the longest line in the array of strings
    public static String findLongestLine(String[] lines) {
        // Implementation to find the longest line
    }

    // Method to find the shortest line in the array of strings
    public static String findShortestLine(String[] lines) {
        // Implementation to find the shortest line
    }

    // Method to count the number of words in each line and store them in an ar
    public static int[] countWords(String[] lines) {
        // Implementation to count words in each line
    }

    // Method to sort the array of word counts in descending order
    public static void sortWordCount(int[] wordCounts) {
        // Implementation to sort word counts
    }

    // Main method to execute the program
    public static void main(String[] args) {
        // Implementation of main program logic
    }
}
```

You will need to implement the main program logic in the `main` method and the methods `findLongestLine`, `findShortestLine`, `countWords`, and `sortWordCount` as per the provided function signatures. Additionally, handle file reading, error handling, and other required functionalities accordingly.

Assignment #3

Building a Sorting Library

You are tasked with developing a sorting library in Java that provides implementations for various sorting algorithms such as Bubble Sort, Selection Sort, and Merge Sort. Your task is to design and implement the necessary classes and interfaces to create a flexible and efficient sorting system.

Requirements:

1. Define a Java interface called `Sortable` with the following method:
    - `void sort(int[] arr)`: Sorts the given array of integers in ascending order.
2. Implement classes for the following sorting algorithms, each of which should implement the `Sortable` interface:
    - Bubble Sort
    - Selection Sort
    - Merge Sort
3. Each sorting algorithm class should provide methods to perform the corresponding sorting operation on an array of integers.
4. Create a `Sorter` class that manages the sorting operations. It should have a method:
    - `void sort(Sortable algorithm, int[] arr)`: Invokes the sorting algorithm specified by the `algorithm` parameter to sort the given array `arr`.
5. Demonstrate the functionality of your sorting library in a `SortingTest` class with a `main()` method. Instantiate objects of different sorting algorithm classes, and use the `Sorter` class to sort arrays of integers using various algorithms.

Sample Output:

```
Using Bubble Sort:
Original array: [5, 2, 9, 1, 7]
Sorted array: [1, 2, 5, 7, 9]

Using Selection Sort:
Original array: [8, 3, 6, 4, 2]
Sorted array: [2, 3, 4, 6, 8]

Using Merge Sort:
Original array: [10, 6, 3, 8, 1]
Sorted array: [1, 3, 6, 8, 10]
```

Your program should demonstrate the use of interfaces to define common behavior among different sorting algorithms and showcase polymorphism in action with the sorting library system.