# Reflection log

The `WordCount` class is a console-based program designed to read a text file, count the number of words, calculate the average word length, and display the results to the user. The program uses basic file handling techniques and string manipulation to achieve these objectives. The class performs the following main tasks:

1. Reads the content of a file specified by the user.
2. Counts the number of words in the file.
3. Calculates the average word length.
4. Displays the results (total word count and average word length) on the console.

**Main Method:**

The main method initializes the program, which involves:

- Defining the file path to be used.
- Using a `Scanner` to read the file and process the words.
- Counting the words and calculating their average length.
- Handling any exceptions that may arise (like file not being found).

```
public static void main(String[] args) {

    String filename = "../Chapter11/src/Mastery/source.txt";


    try {

        File file = new File(filename);

        System.out.println(file.exists());


        Scanner scanner = new Scanner(file);


        int wordCount = 0;
        int totalLength = 0;

        while (scanner.hasNext()) {
            String word = scanner.next();
            word = word.replaceAll("^[^a-zA-Z]+|[^a-zA-Z]+$", "");

            if (!word.isEmpty()) {
                wordCount++;
                totalLength += word.length();
            }
        }

        scanner.close();

        double averageLength = wordCount > 0 ? (double) totalLength /
wordCount : 0;

        System.out.println("Total words: " + wordCount);
        System.out.printf("Average word length: %.2f\n",
averageLength);

    } catch (FileNotFoundException e) {
        System.out.println("The file '" + filename + "' does not
exist.");
```

```
    }
}
```

**Constructor:**

The `WordCount` class does not have a specific constructor for this version. The logic is handled directly within the `main` method. There are no additional fields or initialization steps required in this simple program, but for future enhancements (e.g., supporting different file paths or adding additional functionality), a constructor could be introduced.

**File Handling & Logic:**

The program uses a `Scanner` object to read the file word by word. It cleans up each word by removing any punctuation or non-alphabetical characters using a regular expression (`word.replaceAll("^[^a-zA-Z]+|[^a-zA-Z]+$", "")`). This ensures that only clean words are counted.

It proceeds as follows:

1. **Word Count:** Each time a valid word is encountered, the word count is incremented, and the total length of the words is accumulated.
2. **Average Word Length:** After reading all the words, the program calculates the average word length using the formula:
   `averageLength = totalLength / wordCount`.

The results are then displayed:

- **Total word count**
- **Average word length (rounded to two decimal places)**

**Error Handling:**

The program includes basic error handling using a `try-catch` block to catch and handle the `FileNotFoundException`. This ensures that the user is informed if the file does not exist or cannot be opened. Additionally, the program checks whether the file exists using `file.exists()` before attempting to read its contents.

If the file cannot be found, the user is informed with a clear message:
`The file 'filename' does not exist.`