- **Bad File Path**
    - **Problem**: The path `../chapter11/src/chapter11/wordcount` may cause a `FileNotFoundException`.
    - **Cause**: Relative paths depend on the IDE's working directory.
    - **Fix**: Change the path to `path/to/your/file/wordcount.txt` and ensure the file exists.
- **Poor Naming Convention**
    - **Problem**: The file doesn't follow Java's class naming convention.
    - **Cause**: Lowercase names can cause confusion.
    - **Fix**: Rename the file to `WordCount.java`.
- **FileNotFoundException Not Handled Gracefully**
    - **Problem**: File paths may not be handled properly.
    - **Cause**: Incorrect paths or missing files.
    - **Fix**: Validate the file's existence and format.
- **Not Using Try-With-Resources**
    - **Problem**: Manual resource closing can lead to issues if an exception occurs.
    - **Cause**: The `Scanner` might not close properly.
    - **Fix**: Use `try-with-resources` to auto-close resources.
- **Count Letters in Words Issues**
    - **Problem**: `word.replaceAll("[^a-zA-Z]", "")` excludes non-English characters.
    - **Cause**: Doesn't handle accented or special characters.
    - **Fix**: Adjust the regex if needed.
- **Average Word Length Precision**
    - **Problem**: Output may have excessive decimal places.
    - **Cause**: Floating-point precision isn't user-friendly.
    - **Fix**: Use `System.out.printf` to round to two decimal places.
- **Handling Empty Files**
    - **Problem**: Empty files may cause division-by-zero errors.
    - **Cause**: `wordCount == 0` leads to invalid calculations.
    - **Fix**: Add a check to prevent division by zero.
- **Performance on Large Files**
    - **Problem**: Word-by-word reading is slow for large files.
    - **Cause**: Overhead from word-by-word processing.

- **Fix**: Use `BufferedReader` for better performance.