Approach

The implementation follows a modular design, dividing responsibilities across different classes:

- FileReaderManager: Reads words from a file into a list.
- CharacterFrequencyCounterManager: Generates a unique frequency-based key for each word.
- AnagramGrouperManager: Groups words based on their frequency key.
- FileWriterManager: Writes the grouped anagrams back to a file.

Maintainability

- The modular approach enhances maintainability, making it easy to update or extend specific functionalities independently.
- Methods are well-named and concise, making the code readable and easy to debug.
- Exception handling ensures proper resource management, such as closing files after use.

Scalability

- The HashMap in AnagramGrouperManager class ensures efficient lookups and insertions.
- Using a StringBuilder in CharacterFrequencyCounterManager class avoids excessive string concatenation, improving performance.

Performance

- The use of a character frequency array (int[26]) instead of sorting makes key generation O(n) rather than O(n log n).
- File operations use buffered reading and writing for better performance.

External Libraries

No external libraries were used, ensuring simplicity and portability.

Scalability Considerations

To support large datasets I would:

- Store words and their frequency keys in a database to avoid memory overflows.
- Store files in cloud services to handle large data more efficiently.
- Store words in compressed formats, to reduce I/O overhead.