MultiThreaded Indexing API

Ben Murray – G00275740

This project intakes a user-specified text file and outputs an index of words found in that text mapped to their dictionary definitions and a list of pages on which each word is found. The program parses the users file in a conventional way, reading line by line and breaking each line into words. The program also leverages Virtual Threads to increase efficiency where applicable. The provided dictionary and common word files are parsed in this way, with a new Virtual Thread being created for each line streamed in.

All file parsing classes fall into a hierarchy defined in the program, with the interface Parsator as the root of the hierarchy, and progressively less abstract classes branching from there, demonstrating inheritance (specification and specialisation).

A divide and conquer approach was applied to the rest of the indexing process, with classes adhering to the SRP. For example, dedicated classes were created for filtering sets of words generated by parsing classes, and building a final mapping of words to their definitions and page lists. The goal of this, as well as using an overarching Indexer class to delegate tasks privately, was to demonstrate encapsulation, while leaving enough functionality exposed for the user to extend the program if required.

Extra functionality is included, and offered to the user once the program has built the index. An output menu with the following options is presented:

- Print index words in sorted order to console (with user defined number of words per row).
- Print words in reverse sorted order to console.
- Print total number of unique words in index to console.
- Print list of most frequently found words in text to console (user defined length of list).

Instructions:

User is prompted to enter paths to their chosen text file, dictionary file, common words file and the file they wish the index to be written to. User is notified of complete, and total processing time in milliseconds. The user is then prompted to select options for extra output as described above. Upon completion, the program closes.

Note: Successfully tested on 2 separate Windows machines, functions as expected. Attempted testing on Ubuntu using WSL, but latest available version of JDK (17) was not compatible with Virtual Thread features.

References:

- https://stackoverflow.com/questions/5648336/how-select-first-n-items-in-java-treemap
- https://softwareengineering.stackexchange.com/questions/331909/whats-the-complexity-of-javas-string-split-function
- VirtualThreadFileParser.java – Available on Moodle