**Word Statistics Project Documentation**

**Overview**

Project 2: Word statistics:

Write a program that reads all text files form a specific directory and return word statistics (number of words

per file/directory, longest word, shortest word, number of “is”, “are” and “you”).

- The program should have a simple GUI

- The input of the program is a directory

o It should then search for all text files that reside in that directory

o There should be an option to check for text files in subdirectories

- While the program is processing text files, it should display them (file names) and the up-to-date

statistics. (Statistics should be updated in run time)

**Features**

1. **Directory Selection:** Users can select a directory using a file chooser to analyze all files within that directory.
2. **Include Subdirectories:** Users can choose whether to include subdirectories in the analysis.
3. **Multi-threaded Processing:** Each file is processed on a separate thread to improve performance and responsiveness.
4. **Real-time Display:** Word statistics are updated and displayed in real-time as each file is processed.
5. **GUI Display:** The results are presented in a table format within the GUI, providing information such as file name, word count, occurrences of specific words ("is," "are," "you"), longest word, and shortest word.
6. **Global Statistics:** The GUI also includes global statistics for the entire dataset, displaying the longest and shortest words across all processed files.

**Classes and Components**

**1. WordStatistics Class (GUI)**

* **Attributes:**
  + **directoryTextField**: JTextField for displaying the selected directory path.
  + **includeSubdirectoriesCheckBox**: JCheckBox for enabling/disabling subdirectory inclusion.
  + **browseButton**: JButton for opening a file chooser to select a directory.
  + **startButton**: JButton for initiating the file processing.
  + **resultTable**: JTable for displaying the word statistics.
* **Methods:**
  + **initUI()**: Initializes the GUI components and sets the look and feel.
  + **browseForFile()**: Opens a file chooser for selecting a directory.
  + **startProcessing()**: Initiates the file processing by creating a new thread.

**2. FileProcessor Class (Thread)**

* **Attributes:**
  + **directoryPath**: The path of the directory to be processed.
  + **includeSubdirectories**: Flag indicating whether to include subdirectories.
  + **resultTableModel**: DefaultTableModel for updating the GUI with word statistics.
* **Methods:**
  + **run()**: Implements the Runnable interface and initiates the file processing.
  + **processFiles(File directory)**: Recursively processes files in the specified directory.
  + **processFile(File file)**: Processes a single file, updating word statistics in real-time.
  + **updateGui(...)**: Updates the GUI with word statistics, including both per-file and global statistics.

**3. WordStatisticsData Class**

* **Attributes:**
  + **longest**: String representing the longest word across all files.
  + **shortest**: String representing the shortest word across all files.
  + **ahmed**: Boolean flag indicating whether Ahmed's special conditions are met.
* **Methods:**
  + **getLongest()**, **setLongest(String)**: Getter and setter for the longest word.
  + **getShortest()**, **setShortest(String)**: Getter and setter for the shortest word.
  + **isAhmed()**, **setAhmed(boolean)**: Getter and setter for Ahmed's special condition.

**Usage**

1. **Run the Application:**
   * Compile and run the **WordStatistics** class.
   * The GUI will appear, allowing users to select a directory and configure processing options.
2. **Select Directory:**
   * Click the "Browse" button to choose a directory for analysis.
3. **Configure Options:**
   * Use the checkbox to include or exclude subdirectories from the analysis.
4. **Initiate Processing:**
   * Click the "Start" button to begin processing files concurrently.
5. **View Results:**
   * The GUI will display real-time word statistics for each processed file.
   * Global statistics for the entire dataset are also shown.
6. **Note on Ahmed's Conditions:**
   * Ahmed's special conditions are met during processing and are reflected in the global statistics.

**Team Work**

Ahmed Saed – Ahmed Medhat – Ahmed Tarek – Ahmed Alaa – Ahmed Khaled.

Each Member of the team coded a function, then one of us created the GUI then we put it all together.

**Conclusion**

The Word Statistics project provides a practical example of multi-threaded file processing with a GUI, offering users a convenient way to analyze word occurrences in a directory. The application demonstrates the use of Java Swing for creating a responsive and interactive user interface, while the use of threads enhances performance by concurrently processing multiple files.