Text Encoder/Decoder

@author (Your Name)
@version Java 19

Description

This is a console-based application that allows the user to **encode and decode text files** using a dictionary of word-to-code mappings provided in a CSV file. The application is menu-driven, providing options to set file paths, switch between encoding and decoding modes, and execute the chosen process. The rationale for this design was to make the tool both **flexible and user-friendly**, while ensuring robust error handling.

The project makes use of **arrays, file I/O utilities, exception handling, and string manipulation**, all topics covered in lectures and labs. For example, the decision to load mappings into arrays was influenced by the lecture on collections and array management, while the use of *try/catch* reflects error handling practices demonstrated in labs.

To Run

From console at *.jar* file directory:
```
java -cp ./encoderdecoder.jar Runner
```

Then navigate through console options to set desired files.

If no mapping file, input file, or output file is specified, the program attempts to use default paths (*./encodings-10000.csv*, *./test.txt*, *./out.txt*). If these are missing, the user is prompted to enter custom file paths.

Features

- **Specify Mapping File** - Input the path to the CSV mapping file containing word-to-code definitions. Default: *./encodings-10000.csv*
- **Specify Input/Output Files** - Input path for text file to encode/decode, and output file path for results.
- **Switch between Encoding and Decoding Modes** - Menu option allows toggling between modes. Encoding replaces words with numerical codes, decoding reconstructs original text from codes.
- **Robust File Handling** - Application uses Java NIO (Files) for efficient reading/writing. Exceptions are caught and displayed gracefully.
- **Progress Feedback** - A progress meter prints updates for each line processed, helping track performance on large files.

Reflection

The design prioritizes **separation of concerns**:
- *Encoder* and *Decoder* handle logic for conversion.
- *Menu* manages user interaction.
- *FileUtility* encapsulates file I/O.
- *MappingLoader* is responsible for parsing the CSV mapping file.

This modularity was directly influenced by **object-oriented programming principles** discussed in lectures. By separating each responsibility, the code is more maintainable and easier to extend.