Peter Fabens
CS150
Professor Pfaffmann
2/20/2022

Project 1 Part 2

Design Description:

- Gap Buffer
    - cursor - represented as an int, denotes the position of an imaginary cursor which is used as a reference position in the array for entering and removing chars.
        - Methods for the cursor center around moving it.
    - char array - array comprising the string stored in the buffer, with all chars before the cursor position shunted to the beginning of the array and all characters after shunted to the end. This creates a gap that allows new chars to be entered without requiring reorganization of the array.
        - Methods for the array center around managing size and editing contents, as well as reorganizing it as cursor position changes.
- LinkedListBuffer
    - CharNode - node that holds the item in front, the item behind, and a char value
        - CursorNode - special type of CharNode that has methods for manipulating the list, as the only places where edits happen is at the cursor
    - head, tail, cursor - references to relevant nodes in the list, to allow the structure to access nodes.

- Buffer Structure
    - type - String that holds the structure's buffer type, either linked or gap.
    - StructNode - holds a text buffer, linked or gap, and a reference to the next node
    - cursor - int holding the index of the line the cursor is on
    - Methods which allow manipulation of the structure, the backbone of which is get_node(), which gets a node in the list at a specified index.
- Document IO
    - doc_file - File which holds the reference for a particular file. This file is what is referred to here as the 'open' file despite not always being considered open by the JVM. It is only open in the sense that it is used for all IO until a new file is opened.
    - doc_in - Scanner which takes in text from the open file.
    - doc_out - PrintStream which prints the contents of the document to the open file.
        - Because PrintStream is best for continually printing and less good for saving a bunch of text and overwriting the previous text, this PrintStream doesn't stay open between uses. It is reinitialized as a new PrintStream each time store_file() is called, as store_file() needs to rewrite the file instead of merely adding to it.
- Document

        - type - String that holds the structure's buffer type, either linked or gap.
        - Mediates interaction between the Buffer Structure and the Document IO, carbon copying much of the functionality from the Buffer Structure and allowing the Buffer Structure's data to be passed to the Document IO and vice versa, scanning the input file and and populating the Buffer Structure with it.
- Editor
    - type - String that holds the structure's buffer type, either linked or gap.
    - Integrates document and data structure with a system command line user interface, using commands

New Methods:

- Gap Buffer
    - cursorString()
        - Identical to toString() except that empty spaces in the Gap Buffer are represented as hyphens, necessary as a low-tech debugging tool for the cursor movement methods.
    - move_cursor(int move)
        - The cursor movement methods are largely identical with small differences, so this is a generalized version. Necessary for avoiding about 30 lines of highly repetitive code, and for ensuring flexibility.
    - set_cursor(int index)
        - Less user friendly version of move_cursor(), which doesn't return a boolean or try to protect against NullPointerExceptions by checking an input parameter. Necessary in some cases where setting the cursor location is more useful than relative movement.
    - grow(int e)
        - Grows the array by the specified size.
            - While this method is not called often, it reduces code clutter and allows better generality in the future in case it is needed in the future.
- LinkedListBuffer
    - cursorString()
        - Same function as for GapBuffer, puts a special character at the index of the cursor to show easily where it is.
- Buffer Structure
    - cursorString()
        - Mirrors the cursorString() of the line the cursor is currently on.
    - get_node(int index)
        - Returns the node at the specified index in the list structure. Avoids a ton of duplicate loops whenever a particular node is needed.
    - get_new_type_obj()
        - Returns a new object of the type specified in the type variable.

- Document IO
    - write_file(String text)
        - Writes text to the open file. Necessary for writing any data to files, since files cannot be accessed outside Document IO.
    - file_name()
        - Returns name of open file, necessary for the Document to retrieve it.
- Document
    - None
- Editor
    - format_filename(String input)
        - Takes a string and formats it to be an acceptable filename. Avoids duplicate code, as this is necessary more than once.
    - print_cursor()
        - Prints out a version of the current line that has been formatted to show the position of the cursor as a ^ on the line below. Called after almost every command, so avoids a lot of duplicate code.
    - execute()
        - Takes user input and executes commands based on it.
        - Additional Commands:
            - h
                - Help, prints list of commands
            - rm
                - Removes characters behind cursor
            - p
                - Prints the entire document at once for easy viewing.

Ambiguity:

- Gap Buffer/LinkedListBuffer
    - None
- Buffer Structure
    - The biggest source of ambiguity in the Buffer Structure was deciding when to implicitly move the cursor as a part of another method. For example, when inserting a line above the cursor line, everything below the new line is shifted downwards, including the line the cursor is on. The question then becomes, should the cursor stay at the same *index*, or should it stay on the same *line*? I decided on the latter, so that when removing lines or adding new ones above the cursor, the cursor index will move to stay with the same line of text.
    - A similar issue arises from removing the current line, where it must be decided whether to move to the line above or the line below. I decided on the line above.
- Document IO
    - I was slightly confused by how the load_file() and close_file() methods were intended to be implemented, as I initially thought the intent was to open the PrintStream and Scanner within those methods. Since the PrintStream had to be

reopened each time the file was saved, this didn't seem like a logical method structure. I chose to have a File object for storing the reference to the file, which is set in the open_file() method. The close_file() method simply closes any existing Scanner and PrintStream, and removes their references.

- Editor
    - It was ambiguous to me what the 's' command should do, whether it should make up a generic file name or prompt for one. I chose to prompt.
    - I wasn't sure whether the main method should just error out if no args were provided or if it ought to just default to one option, but I chose to make it default to LinkedListBuffer if anything besides "gap" is entered for the type.