CS 253 (Spring 2024)
Assignment 4: Autocomplete

Goals

- Use the trie structure to implement an auto-complete function.
- Test your autocomplete implementation for accuracy and speed.

Most virtual keyboards provide the option of auto-complete, which gives a list of candidate words that you wish to type given a prefix you enter. For instance, when you type "sh", it gives you a list of candidate words such as "she", "shell", "ship", etc.

```
"sh" -> ["she", "ship", "shell", ...]
```

If you select a word from the candidates, it should learn your selection as the top candidate for that prefix. For instance, if you select "ship" from the example above, the next time you enter the same prefix "sh", it should give a list of candidates where the first item is "ship" instead of "she".

```
"sh" -> ["ship", "she", "shell", ...]
```

Your task it to write a program that gives a list of candidate words for any prefix and learns the selected candidates.

Details

- Create a class called `Autocomplete`
  - This class should extend the abstract class `AbstractAutocomplete`, which extends the class `Trie`.
  - The value type of the generic `T` can be a collection of strings (e.g. `List<String>`).
  - You must create a constructor that takes two parameters, `dict_file` and `max`, and calls its super constructor, which reads all words in the dictionary file (e.g., `dict.txt`).
- Override the `getCandidates()` method that takes a prefix and returns a list of candidate words matching the prefix.
  - The maximum number of candidates in teh list must be the return value of the `getMax()` method.
  - The most recently selected candidate should appear as the 1st item, the 2nd most recently selected candidate should appear as the 2nd candidate, and so on.
  - The rest of the candidate list should be filled with the shortest words matching the prefix.
  - If there is more than one candidate with the same lengths, they should be sorted alphabetically.
  - Make sure the same candidate does not appear more than once.
- Override the `pickCandidate()` method that takes a prefix and a selected candidate, and saves the information.
  - This is the most recently selected candidate for that particular prefix. It must appear as the first time in the candidate list when the same prefix is entered next time.

Notes

- Do not change the dictionary file. If you find anything peculiar about the dictionary file, please let me know so everyone works on the same copy of the dictionary file.
- Please test the program yourself. I will evaluate your program using my tests and measure the performance (both speed and accuracy).
- Take a look at Map if you are not familiar with method in the standard library.
- If you are having trouble with implementing `getCandidates()`, think about how to traverse the trie given any node.
- If you are having trouble with implementing `pickCandidate()`, take a look at `Trie.put()`
- You are not allowed to use any type of Map to store candidates that do not exist in the dictionary.
- All picked candidates must be treated as real words.
- White spaces are not allowed as input. You should trim all input strings.

Honor Code
The assignment is governed by the College Honor Code and Departmental Policy. Please remember to have the following comment included at the top of the file.

Note: This assignment was also given in Prof. Choi's section of CS 253. We will be comparing submissions between our sections for code plagiarism.

```
/*
THIS CODE WAS MY OWN WORK, IT WAS WRITTEN WITHOUT CONSULTING ANY
SOURCES OUTSIDE OF THOSE APPROVED BY THE INSTRUCTOR. _Your_Name_Here_
*/
```

Submission:
Submit your completed `Autocomplete.java` file to Gradescope.

Grading:

- Correctness and robustness: your autocomplete methods works correctly for any sequence of inputs and selections. (60 pts.)
- Efficiency: Your methods run quickly. In particular `getCandidates()` and `pickCandidate()` should not have a noticeable delay when executing. (30 pts.) [Note: You cannot get points for speed, if your program doe not work correctly.]
- Code clarity and style (10 pts.)