Simranbanu Roshansha Diwan B00833562

Assignment 4–External Documentation

Overview

This program loads puzzle from given input. While loading puzzle, positions on grid which are available are given and the words which are needed to be stored are given as input. It will solve puzzle by putting words in available grid positions.

Files

- FillInPuzzle.java: This class has methods to load puzzle, solve puzzle, print puzzle and a method which returns number of choices made so far which are back tracked.

Inputs

Program takes available grid positions, and word list as input.

Output

-Solved puzzle is the output

Key algorithms and design elements

Functions

1) loadPuzzle

-Input: BufferedReader object.
-It will get puzzle grid details as input. It will also get words to be filled in as input.
-Representation of grid: Puzzle 2-D array is used to store puzzle state. It has space where word can't be filled. It has '-' where puzzle is supposed to have words.
-Words are stored in one ArrayList. It will be sorted alphabetically.
-It will store details of empty slots of grid. (Details means– starting indexes x and y, size of slot, direction of slot)
-It will return true if the grid is successfully loaded. Otherwise, it will return false.

2) solve

- It solves the puzzle. We have variable choices to keep track how many backtracking is made.

- To have efficiency,

*Stack is used.

*Character level check is only done if length of slot and word matches. So, it won't check any position with spaces. It will directly start check from the slot position upto length.

1. Repeat until word arrayList size>0

2. It will take first word from words arrayList.

3. It will try to fit word in available slots.

4. If it fits then store that word and information about where it is stored in stack and remove that word from arrayList. Goto step 1

5. If it doesn't fit that means previously made series of choices does not yield to the solution. So pop from the history stack (details about where last word was placed) and FilledWords stack. Add that word to ArrayList of words. Again put '-' where word was fitted incorrectly so it will make that place available to new coming words. Don't put '-' where two words were crossing and and were filled. Goto step 1. Choices++.

6. Check if any place in grid has '-'. If so, that means grid is not solved so return false. Otherwise, return true.

3) print

- It will print the grid.

4) choices

- It will return how many choices made so far which are undone.

5) copyMatrix

- It will have two matrix as parameter. It will copy second matrix into first matrix.

6) fillHorizontal

- It will have grid positions, length of current word to put, direction, current word, and pos (it is index of arrays which have grid current position details)
-It will try to fit word horizontally.
-It will put word if the place has '-' or current letter which is on that position on grid matches letter of word to be put. Return 1.
- If there is any conflict then it will not put word at that place. And return 0.

7) fillVertical

- It will have grid positions, length of current word to put, direction, current word, and pos (it is index of arrays which have grid current position details)
-It will try to fit word vertically.
-It will put word if the place has '-' or current letter which is on that position on grid matches letter of word to be put. Return 1.
- If there is any conflict then it will not put word at that place. And return 0.

Assumptions

- Puzzle is case insensitive.
- The words in the puzzle fill from top to bottom or left to right.

Choices

- If a user calls solve method before loadPuzzle method then solve method will return true.