CSCI 3901 Assignment 4

Name : Yogish Honnadevipura Gopalakrishna(B00928029)

Overview

This program solves the puzzle from the given data set. The data set consists of words and the instructions on where the words are meant to be placed. This program tries all the permutation and combination of words to try to fit in the puzzle. If there is any feasible combination, then the puzzle will be "**filled"** and "**true"** will be returned. Otherwise, the program fills the "**period"** in the places where there should have been words and returns "**false"**. The program will also count the number of backtracks the program has made to complete the puzzle.

Files and external data

There are two main files:

- FillnPuzzle.java -> This file contains the following methods
    - loadPuzzle() - This method is used to read the data from the bufferreader and store it in the program
    - Solve() - This method first places periods to the places where there are words or characters expected. Then it will call the method puzzleSolver to solve the puzzle. If the puzzle is solved, it returns true, else it returns false.
    - puzzleSolver() - This method actually solves the puzzle. Every time when an word is fit in the puzzle, it will use recursion or backtracking to solve the puzzle
    - print() - This method prints the puzzle into the console like an actual puzzle
    - choices() - This method returns the number of backtracks that the program made while trying to solve the puzzle.
- Helper.java -> This file contains the following methods
    - fillPeriods() - This method is used to fill period into the puzzle by reading the data stored.
    - deepCopy() - This method is used to deep copy an 2D array which will be used as an backup while backtracking in the program

Data structures and their relations to each other

- Stored String of words(Ex :Plush) as an ArrayList in an attribute called “**words**”.
- Stored the information related to the words(Ex :0 0 5 h) as an ArrayList in an attribute called “**wordsDescription**”.
- Stored puzzle data in a 2D array in an attribute called “**Puzzle**”.
- Stored data regarding availability of words in an boolean array in an attribute called “**wordAvailabilityArray**”.

Choices

- Created a Helper class for deepCopying the puzzle and a method which places period inside the puzzle

Key algorithms and design elements

The algorithm and design used for Solving the puzzle is as follows:

1. loadPuzzle method stores the data related to the puzzle which will be later used in solve method
2. The solve method first places the period in the space where characters are expected.
3. A Boolean array is created which tells the program if the word is available to use or not in the program
4. PuzzleSolver method is called to solve the puzzle.
5. The 2D array or the puzzle is backed up or copied using an String array Object namely “backUp”.
6. For every space present(ex : 0 0 5 h) try to fit any word that is not existing in the puzzle. If a word is fit, then recursively call the method until the puzzle is filled
7. If the puzzle is not solved, then backtrack using the backUp variable, and also increment the count of the variable “**numberOfBacktracks**”.
8. Check if the puzzle is solved between step 6 and step 7, if the puzzle is solved or if after all the combinations if the puzzle remains unsolved, return the puzzle back to the solve() method.
9. If the puzzle is solved, return true else return false.

Limitations

- The program might do a lot of backtracks if the dataset is very large.

Test Cases

FillInPuzzle Class

loadPuzzle(BufferedReader stream)

Input validations:
- Stream is null
- Check if input is String

Data Flow:
- Calling loadpuzzle at the start
- Calling loadPuzzle after solve

solve()

Data Flow:
- Calling solve before loadPuzzle
- Calling solve after loadPuzzle

print(PrintWriter outstream):

Input validations:
- outstream is null

Data Flow:
- Calling print before loadPuzzle and solve
- Calling print after loadPuzzle and solve

choices():

Data Flow:
- Calling choices before solve
- Calling choices after solve