CSCI 3901 Software Development Concepts Assignment 2 External Documentation

Problem 1

Overview

Write Test Cases for Sudoku Class

Input Validation Test Cases

Public Sudoku (int size)

- Null Value passed
  Get the error message
- Boolean value passed
  Get the error message
- String value passed
  Get the error message
- Float value passed
  Get the error message
- Object passed
  Get the error message

Public Boolean setPossibleValues (String values)

- Empty String passed
  Return false
- Null value passed
  Return false
- Int value passed
  Compilation error
- Object Passed
  Compilation error
- Duplicate value passed
  Return false
- Character value passed
  Compilation error

Public Boolean setCellValue (int x, int y , char letter)

- Different numbers of parameter passed
  Compilation error
- Null Character passed to letter value
  Return false
- Character passed with single space

Return true

- Character data type passed to all paramaters
  Return true
- Integer data type passed in all 3 paramters
  Compilation error
- Letter value passed which is not in the values String
  Return False

Public Boolean solve()

- Arguments passed while calling the method
  Compilation Error

Public String toPrintString(char emptyCellLetter)

- Null value passed
  Get the error message
- Different data type value passed
  Compilation error

Boundary Tests Cases:

Public Sudoku (int size)

- -1,0,1 passed
  Gives Exception
- Greater than the range of int is passed
  Compilation error

Public Boolean setPossibleValues(String values)

- String greater or lesser than the (size*size) is passed
  Return false
- String with defined size with unique characters passed
  Return true
- String with less size of (size*size) is passed
  Return false

Public Boolean setCellValue(int x, int y, char letter)

- 0, or negative data passed in x or y
  Return false
- Values of x and y is greater than the size defined
  Returns false

Public Boolean solve()

- Solve the Sudoku with one or more empty grid
  Return false
- Solve the Sudoku with size in limit the size defined for gird
  Return false

Public String toPrintString(Char emptyCellLetter)

- Called the function before solving sudoku
  Unsolved Sudoku would be printed
- Called the function after Solving Sudoku
  Solved Sudoku string would be printed

Control Flow Test Cases:

Public Sudoku(int size)

- Constructor is called and the sudoku is created

Public Boolean setPossibleValues(String values)

- Getting the string values

Public Boolean setCellValue(int x, int y, char letter)

- Adding duplicate value in any column
  Return false
- Adding duplicate value in any row
  Return false
- Adding duplicate values in sub grid
  Return false
- Calling the function on same cell
  Return false
- Any row or columns contains unique characters from the value string
  Return true
- Any sub grid contains unique characters from the value string
  Return true

Public Boolean solve()

- Solve when there is only one sub grid is left to solve
- Solve when there is duplicate value in row or column
- Solve when there is duplicate value in sub grid.

Public String toPrintString(char emptyCellLetter)

- Print before the Sudoku is solved(partially solved Sudoku will be printed)

Data Flow Cases

- Call Solve() before calling the setCellValues(int x, int y, char letter)
- Call Solve() before calling the setPossibleValues(String values)
- Call toPrintString(char emptyCellLetter)
- Call setCellValue (int x, int y, char letter) before calling the setPossibleValues(String values)
- Call solve()
- Call toPrintString(char emptyCellLetter) before calling the solve()
- Call setCellValue (int x, int y, char letter) before calling the solve()

Problem 2

Overview

The program generates an unbalanced binary search tree with the functionality to quickly search for frequently accessed items. Most searched items will automatically be placed toward the tree's root.

Files and external data

The program consists of 3 files:

• SearchTree.java – class which contains the program's main functionality, with tree restructuring code.

• Node.java – a class that has the blueprint of the Node.

• TestCases.java – a helper class for the input and boundary test cases for the search tree class.

Data structures and their relations to each other

The program constructs the binary search tree using Linked List data structure, The value of the key is stored in the class object called node, the object will store the value of the key as a string, The reference of the parent node in the tree, the reference of the left and right child of the node.

The whole tree structure program uses recursion to add, find and to traverse in a tree. There is an end condition defined to return from the function call. The Search Tree class constructor will define the root of the tree as null

When the add function is called, it will take the key as a parameter, and create the node of the tree with node value as string key. Also it will store the reference of the parent, left child, right child in node class attribute. For the reference it will store the search count, and depth of the node as a class attribute. After creation of the node it will be added in the tree with appropriate location according to the lexicographical order of the string. If there any error occurs (null or empty string provided) then it will return false

Find method works on the recursion as it takes the key as string, then the find method will search that key in the tree with recursion function call, if the match is found then it will count the depth of the node from the root and return it. Also this find method will perform the tree restructuring if the found node has the more search count than the parent’s one. In the restructuring, the found node will be moved to one level up and the parent node of the found node will replace the found node, so it will go down one level. While performing the tree restructuring, the linking of the

parent, left child, and right child will be taken care. Node reset function will take the root node and traverse into the tree and reset each node's search count to 0. A print Function will iterate over each node value in the tree and print the key and the depth of the tree.

Classes co-relations

To make code more modular The program is separated with three different java files, SearchTree, Node, and TestCases java file. SearchTree class will have main method and the program execution will start from there. The object of the SearchTree class will be created, and the method would be called on that objects.

There is a separate Node class file, which provides the template of the node object which can be created with the help of this class. Nodes are created in the add function of the searchTree Class.

To test the program, a separate TestCases java file is provided, which performs input validations, and the boundary test cases validation.

Assumptions

- No assumptions

Limitations

- Case sensitivity is not checked, so keys with different case sensitivity will be treated as the same