Assignment 2

Team Members:

1. Dhrumil Amish Shah

Date:

2021-01-31

—

Subject:

Software Development Concepts

—

Professor:

Matthew Amy

Sudoku Puzzle

A Sudoku puzzle is a square grid of dimension **n x n** where each grid is of dimension **n x n**. In total there are **n^2 x n^2** squares. A slight generalization of Sudoku is to use collection of **n^2** distinct items instead of the numbers from **1 to n^2**. Example, we can use letters [a, i] for a **9 x 9** Sudoku. Any set of **n^2** unique characters will be allowed.

Test Cases

➪ Input Validation Cases

1. public Sudoku(int size)

   - A constructor for a Sudoku grid with **size^2 x size^2** cells.

   - Send a negative size.
     - Throws exception. (Invalid size)
   - Send a 0 size.
     - Throws exception. (Invalid size)
   - Send a 1 size.
     - Throws exception. (Invalid size)

   => Allowed: **[2, n)** where **n** is maximum int size. This will create a Sudoku grid with **size^2 x size^2**. (Operation success)

2. public boolean setPossibleValues(String values)

   - Takes a string of length **size^2** where the characters of values are the unique items allowed in the cells. Returns **false** if any error occurred and **true** otherwise.

   - Send a null string.
     - Returns false. (Not allowed)
   - Send an empty string.
     - Returns false. (Not allowed)
   - Send a string of size less than **size^2**.
     - Returns false. (Not allowed)
   - Send a string of size greater than **size^2**.
     - Returns false. (Not allowed)
   - Send a string with duplicate/repeated characters.
     - Returns false. (Not allowed)

   => Allowed: **values** is a string of length **size^2** with **size^2 unique characters.** (Operation success)

3. public boolean setCellValue(int x, int y, char letter)
   - Sets the value in cell (x, y) to the character letter.
     - Send a negative x value.
       - Returns false. (Not allowed)
     - Send a x value greater than **(size^2 – 1)**.
       - Returns false. (Not allowed)
     - Send a negative y value.
       - Returns false. (Not allowed)
     - Send a y value greater than **(size^2 – 1)**.
       - Returns false. (Not allowed)
     - Send invalid character in letter parameter.
       - Returns false. (Not allowed)

     => Allowed: **0 <= x, y < size^2** and **valid letter** character**.** (Operation success)

4. public boolean solve()
   - Solve the puzzle, returning **true** if a solution was found and **false** otherwise.

     => No input validation test.

5. public String toPrintString(char emptyCellLetter)
   - Return a string representing the current state of the grid. The grid is returned with no spaces between cells, the character emptyCellLetter used for any empty cells, and lines separated by carriage returns (\n).
     - Send an empty character (\0)
       - Returns null. (Empty char is not allowed – Sending \0 as emptyCellLetter will not create the grid properly)
     - Send '\n' string.
       - Returns null. (Empty char is not allowed – Sending \n as emptyCellLetter will not create the grid properly)
     - Send any unique symbol used for puzzle. (**size^2 unique symbols**).
       - Returns null.

     => Allowed: **Valid emptyCellLetter** character that is not \0, \n or any character from unique symbols. (Operation success)

⇨ Boundary Cases

1. public Sudoku(int size)
   - Set size 2 to create grid and sub grid of size 2 x 2 and cells 4 x 4.
     - Creates a Sudoku of size 2 x 2.
   - Set size 3 to create grid and sub grid of size 3 x 3 and cells 9 x 9.
     - Creates a Sudoku of size 3 x 3.

2. public boolean setPossibleValues(String values)
   - **values** already read once (setPossibleValues already called once with unique characters).
     - Returns false. (Not allowed)

3. public boolean setCellValue(int x, int y, char letter)
   - Set a valid letter at **x = 0** and **y = 0**.
     - Returns true. (Allowed)
   - Set a valid letter at **x = (size^2 – 1)** and **y = (size^2 – 1)**.
     - Returns true. (Allowed)
   - Set a valid letter at **x = 0** and **y = (size^2 – 1)**.
     - Returns true. (Allowed)
   - Set a valid letter at **x = (size^2 – 1)** and **y = 0**.
     - Returns true. (Allowed)
   - Set letter in any outer row and column.
     - Returns true. (Allowed)

4. public boolean solve()
   - Solve an empty grid.
     - Returns false. (Solution not found)
   - Solve when there are not enough starting values.
     - Returns false. (Solution not found)
   - Solve an invalid unsolved grid.
     - Returns false. (Solution not found)
   - Solve an invalid filled grid.
     - Returns false. (Solution not found)
   - Solve an invalid grid when only one cell is remaining.
     - Returns false. (Solution not found)

- Solve a valid unsolved grid.
  - Returns true. (Solution found)
- Solve a valid filled grid.
  - Returns true. (Solution found)
- Solve a valid grid when only one cell is remaining.
  - Returns true. (Solution found)

5. public String toPrintString(char emptyCellLetter)
   - Print an empty grid.
     - Returns a grid with all cells filled with emptyCellLetter.
   - Print a false solved grid. (Solution not found)
     - Returns a false solved grid.
   - Print a true solved grid. (Solution found)
     - Returns a solved grid.

⇨ Control Flow Cases

1. public Sudoku(int size)

   => Covered in above cases.

2. public boolean setPossibleValues(String values)

   => Covered in above cases.

3. public boolean setCellValue(int x, int y, char letter)
   - Set in a cell which is already filled.
     - Returns false. (Not allowed)
   - Set letter in a cell which is duplicate in the same row.
     - Returns false. (Not allowed)
   - Set letter in a cell which is duplicate in the same column.
     - Returns false. (Not allowed)
   - Set letter in a cell which is duplicate in the same sub grid.
     - Returns false. (Not allowed)

4. public boolean solve()
   - Solve when only one letter is remaining to be filled in a row.
     - Returns true if solution is found otherwise false.
   - Solve when only one letter is remaining to be filled in a column.

        - Returns true if solution is found otherwise false.
    - Solve when only one letter is remaining to be filled in a sub grid.
        - Returns true if solution is found otherwise false.
    - Solve when a row contains a duplicate value.
        - Returns false.
    - Solve when a column contains a duplicate value.
        - Returns false.
    - Solve when a sub grid contains a duplicate value.
        - Returns false.
    - Solve where multiple solution exists for a grid.
        - Returns false.

5. public String toPrintString(char emptyCellLetter)
    - Some covered under boundary cases
    - Print a partially solved grid.
        - Returns a grid with all empty cells filled with emptyCellLetter.

⇨ Data Flow Cases

1. Call setCellValue() before calling setPossibleValues().
    - Returns false.
2. Call solve() before calling setPossibleValues().
    - Returns false.
3. Call solve() before calling setCellValue().
    - Returns false.
4. Call toPrintString() before calling setPossibleValues().
    - Returns a grid with all cells filled with emptyCellLetter.
5. Call toPrintString() before calling setCellValue().
    - Returns a grid with all cells filled with emptyCellLetter.
6. Call toPrintString() before calling solve().
    - Returns a grid with all empty cells filled with emptyCellLetter.
7. Call setPossibleValues(), then call setCellValue() multiple times with valid letter and again call setPossibleValues() with different values.
    - Returns false.(Not allowed)