CSCI3901 – Assignment2-Problem1

Name: Pratheep Kumar Manoharan

Banner Id: B00837436

Objective:

To write test cases for the SUDOKU program.

Assumptions:

1. setPossibleValues method will return false in case if it receives space or carriage returns as input or in between the valid inputs. It will not trim the space in the front, in the end or in the middle.
2. setCellValue method will overwrite the existing content in the cell.
3. setCellValue will validate the content whether its valid or not and store the value or return false without setting the cell value. Validations like already available in row, column or small grid.
4. solve method will solve the puzzle even if the puzzle is empty (setCellValue is not even called once) by taking the possible values from setPossibleValues method.
5. toPrintString method will print the puzzle even there is no content in the puzzle. All cell will have the character that is passed to the method as parameter.

Test case grid:

S. No	Test Case	Expected result	Actual result
General cases			
1	Compile the program.	There should not be any compilation errors.	
Constructor method test cases			
Input validation test cases			
1	Give a positive integer as input to the method.	It should be taken as valid input.	
2	Give a negative integer as input.	A negative integer is not a valid input. The method should print an error message.	
3	Give zero as input.	Zero is not a valid input. The method should print an error message.	
4	Give null as input	Null is not a valid input. The method should print an error message.	
5	Give a floating number with decimal places as input.	Floating number with decimal places is not a valid input. The method should print an error message.	
Boundary test cases			
1	Give no input to method call.	We need size of the sudoku. So, the method should print an error message.	
2	Give 2 or more inputs to the method.	The method should print an error message.	
3	Give input as 2147483647 (maximum int value).	It should be taken as a valid input.	
setPossibleValues method test cases			
Input validation test cases			
1	Give values as null.	Null is not a valid input. The method should return false.	
2	Give values as empty string.	Empty string is not a valid input. The method should return false.	
3	Give space or carriage returns (\n) as input.	The method should return false.	
4	Give space or carriage returns (\n) in between valid input.	The method should return false.	
Boundary test cases			
1	Give no input to the method call.	The method should return false.	

2	Give 2 or more inputs.	The method should return false.	
3	For a 3 * 3 sudoku, give input as "12345678" less than the expected values.	The method should return false.	
4	For a 3 * 3 sudoku, give input as "abcdefghijk" more than the expected values.	The method should return false.	
5	For a 3 * 3 sudoku, give input as "123456789", the expected values.	The method should return true.	
setCellValue method test cases			
Input validation test cases			
1	Give a negative integer as input in x and other inputs as valid ones.	A negative integer is not a valid input. So, the method should return false.	
2	Give zero as input in x and other inputs as valid ones.	Zero is not a valid input. So, the method should return false.	
3	Give null as input in x and other inputs as valid ones.	Null is not a valid input. So, the method should return false.	
4	Give a value that is more than (n*n) in x and other inputs as valid ones.	n*n is the maximum value that is valid, and this case is invalid. So, the method should return false.	
5	Give a floating number with decimal places as input in x and other inputs as valid ones.	Floating number with decimal places is not a valid input. So, the method should return false.	
6	Give a negative integer as input in y and other inputs as valid ones.	A negative integer is not a valid input. So, the method should return false.	
7	Give zero as input in y and other inputs as valid ones.	Zero is not a valid input. So, the method should return false.	
8	Give null as input in y and other inputs as valid ones.	Null is not a valid input. So, the method should return false.	
9	Give value that is more than (n*n) in y and other inputs as valid ones.	n*n is the maximum value that is valid, and this case is invalid. So, the method should return false.	
10	Give a floating number with decimal places as input in y and other inputs as valid ones.	Float is not a valid input. So, the method should return false.	
11	Give a negative integer as input in letter and other inputs as valid ones.	A negative integer is not a valid input. So, the method should return false.	
12	Give null as input in letter and other inputs as valid ones.	Null is not a valid input. So, the method should return false.	
13	Give value that is more than (n*n) in letter and other inputs as valid ones.	n*n is the maximum value that is valid, and this case is invalid. So, the method should return false.	

14	Give a floating number with decimal places as input in letter and other inputs as valid ones.	Floating number with decimal places is not a valid input. So, the method should return false.	
15	Give y and letter as invalid inputs and x as valid input	Method should return false. (invalid input may be any of the above cases like Zero, float, outside the 1-(n*n), etc.,	
16	Give x and letter as invalid inputs and y as valid input	Method should return false. (invalid input may be any of the above cases like Zero, float, outside the 1-(n*n), etc.,	
17	Give x and y as invalid inputs and letter as valid input	Method should return false. (invalid input may be any of the above cases like Zero, float, outside the 1-(n*n), etc.,	
18	Give x, y and letter as invalid inputs	Method should return false. (invalid input may be any of the above cases like Zero, float, outside the 1-(n*n), etc.,	
Boundary test cases			
1	Give only 1 input as a parameter to the method call.	The method should return false.	
2	Give only 2 inputs as the parameter to the method call.	The method should return false.	
3	Give 4 or more inputs as the parameter to the method call.	The method should return false.	
Control flow test cases			
1	Give the letter that is already in the same row and other inputs as valid ones.	One value can exist only one time in the row. So, the method should return false. (as per assumption).	
2	Give the letter that is already available in the same column and other inputs as valid ones.	One value can exist only one time in the column. So, the method should return false. (as per assumption).	
3	Give the letter that is already available in the same smaller grid and other inputs as valid ones.	One value can exist only one time in the smaller grid (n*n). So, the method should return false. (as per assumption).	
4	Assuming the sudoku accepts only integers, try a char input in the letter variable.	It should not be taken as a valid input. So, the method should return false.	
5	Assuming the sudoku accepts only characters, try an integer input in the letter variable.	It should not be taken as a valid input. So, the method should return false.	
6	Give a valid input in x (which is less than n*n), y (which is less than n*n), and letter which is not in the row, column or smaller grid.	It should be taken as valid input. So, the method should return true.	

7	Give a valid input and that cell is already filled.	The method should return true by overwriting the existing content of the cell. (as per assumption).	
solve method test cases			
Control flow test cases			
1	Call the method with valid content in SUDOKU.	The method should return true if the SUDOKU is resolved.	
2	Call the method with non resolvable content in SUDOKU.	The method should return false if the SUDOKU is not resolved.	
3	Call the method with no content provided we already called the constructor to set size and setPossibleValues to set values.	The method should return true by resolving the SUDOKU by taking the possible values from setPossibleValues method. (as per assumption).	
toPrintString method test cases			
Input validation test cases			
1	Give a character that is already in the sudoku.	The method should not accept this input as valid and should throw a message. As it may confuse with the actual content.	
2	Give carriage return as an input parameter.	The method should not accept this input as valid and should throw a message. As it may confuse with the actual content.	
Control flow test cases			
1	Called the constructor to set the size and called the setPossibleValues to set the possible values. Now call the toPrintString method.	The method will print the character passed as method parameter in all the cells and print the SUDOKU.	
2	Called the constructor to set the size, called the setPossibleValues to set the possible values and called the setCellValue method to set few cell values. Now call the toPrintString method.	check all the below points in the printed string: a. the number of rows should be n*n. b. the number of columns should be n*n. c. check the cell values set using setCellValue method. d. check whether the remaining cells are filled with the character passed as the input parameter. (as per assumption).	

3	Called the constructor to set the size, called the setPossibleValues to set the possible values, called the setCellValue method to set few cell values and called the solve method. Now call the toPrintString method. (consider the solve method returned true).	check all the below points in the printed string: a. the number of rows should be n*n. b. the number of columns should be n*n. c. an element can appear only once in a row. d. an element can appear only once in a column. e. an element can appear only once in a smaller grid.	
4	Called the constructor to set the size, called the setPossibleValues to set the possible values, called the setCellValue method to set few cell values and called the solve method. Now call the toPrintString method. (consider the solve method returned false).	check all the below points in the printed string: a. the number of rows should be n*n. b. the number of columns should be n*n. c. check the cell values set using setCellValue method. d. check whether the remaining cells are filled with the character passed as the input parameter.	
Data flow test cases			
1	Call the constructor method first to set the size of the sudoku, call the setPossibleValues to find the set of values that can be set into the sudoku and then setCellValue to set the cell values of sudoku before solving. After setCellValue solve or toPrintString methods to solve or print respectively. (Normal order)	setPossibleValues should return true if the input is valid or false if the input is invalid. setCellValue should return true if the input is valid or false if the input is invalid. solve should return true if the puzzle is valid or false if the puzzle is invalid. toPrintString should print the content.	
2	Call the setPossibleValues method first. setPossibleValues depend on the constructor method.	The method should return false. As we don't know the size of sudoku.	
3	Call the setCellValue method first. setCellValue depend on constructor and setPossibleValues methods.	The method should return false. As we don't know the size of sudoku and valid content.	
4	Call the solve method first. solve method depend on both constructor and setPossibleValues methods.	The method should return false. As we don't know the size of sudoku and valid input of the sudoku.	

5	Call toPrintString method first. toPrintString method depends on the constructor method.	The method should return null. As we don't know the size of the sudoku.	
6	Call the constructor first and call the setCellValue. setCellValue depend on the setPossibleValues.	The method should return false. As we don't know the valid set of inputs of the sudoku.	
7	Call the constructor method first and call the solve method.	The method should return false. As we don't know the valid set of inputs of the sudoku.	
8	Call the constructor method first and call the toPrintString method.	The method should print the inputted character in all places as sudoku is with no values.	
9	Call the constructor, setPossibleValues and solve methods in sequence.	The method should return true by solving the sudoku with possible inputs it has.	
10	solve and toPrintString are independent of each other. We can call these 2 after setting size, possible values, and cell value.	They should return results appropriately.	
11	setCellValue and toPrintString are independent methods of each other. We can call these 2 after setting size and possible values. After setting the value in each cell, we can print the SUDOKU, so it is as expected.	They should return results appropriately.	