Puzzle Documentation

Dylon Rajah

December 14, 2022

Project's GitHub Repository can be found here: https://github.com/adonofrio23/Software-Engineering-Sudoku

1 Program Purpose

This Program organizes all the data pertaining to the puzzle board and allows other parts of the game to access its data safely. The program communicates directly with the cell class and is used by the algorithms and initialized by the game engine. When other programs need to access the data.

The puzzle is used to represent a regular matrix where every entry is a Cell object. The puzzle will hold the collection of cells in the same orientation as the original matrix. The puzzle program is able to set a cell into the matrix, return a cell object from the puzzle, get and set the size of the puzzle, get and set the difficulty of the puzzle, as well check the validity of the puzzle at any state.

2 Private Variables

The puzzle has variables ”size”, ”difficulty”, and ”cells” that are private variables and cannot be accessed directly from other programs.

2.1 size (int)

The size variable represents the size of the puzzle and it is stored as an int vairable (ex. if size = 4, puzzle is 4x4).The size is passed from the game engine and is decided by the user. The size variable does not change throughout the cycle of a game but it can be during the game. The size variable is set through the ”SetSize()” function and it will be retrieved through the ”GetSize()” function.

2.2 difficulty (int)

The difficulty variable represents the difficulty of the puzzle and is stored as an int variable. The difficulty is passed from the game engine and is decided by the user. The difficulty variable does not change throughout the cycle of a game but it can be during the game. The difficulty variable is set through the ”SetDifficulty()” function and it will be retrieved through the ”GetDifficulty()” function.

3 Public Methods

The puzzle has 7 functions that are accessible from anywhere in the program.

3.1 IsValid()

The IsValid() function takes in coordinates of a cell and checks if that cell is valid. For a cell to be valid, the solution parameter and the value parameter must be the same (more information about the cell parameters can be found in the Cell documentation). If the cell is valid, it will return true. If it is invalid it will return false.

The coordinates of a single cell is passed in. We use those coordinates to pull the cell object from the puzzle. After we have the cell object, we are able to extract the value and solution using the GetSolution() and GetValue() functions (more information about these functions can be found in the Cell documentation). The it checks if those values are the same and returns accordingly.

Parameters:
int row: Represents the row index of the cell we want to retrieve from the puzzle.
int col: Represents the column index of the cell we want to retrieve from the puzzle.

Return Value: Returns a boolean.

3.2 SetCell()

The SetCell() function takes in a cell object and the SetCell() function will place the cell into the puzzle at the location specified in that cell object. There are other programs that check the validity of the parameters so we simply insert into the puzzle and return nothing.

A cell object is passed into the function. The function extracts the row and column from the cell using the GetRow() and GetCol() from the Cell class. The it indexes the location in the puzzle and inserts the cell object into that location.

Parameters:
Cell cell: A fully populated Cell object.

Return Value: Nothing (void)

3.3 SetSize()

The SetSize() function takes in an int variable that represents the size of the puzzle that the user selected. The function has no return value, but it will update the private "size" variable in the puzzle class.

Parameters:
int sizeSet: An int variable representing the size of the puzzle to be set.

Return Value: Nothing (void)

3.4 GetSize()

The GetSize() function doesn't have any parameters. It simply returns the value of the size variable that is held privately in the puzzle.

Parameters:
None

Return Value: int

3.5 SetDifficulty()

The SetDifficulty() function takes in an int variable that represents the difficulty of the puzzle that the user selected. The function has no return value, but it will update the private "difficulty" variable in the puzzle class.

Parameters:
int diff: An int variable representing the difficulty of the puzzle to be set.

Return Value: Nothing (void)

3.6 GetDifficulty()

The GetDifficulty() function doesn't have any parameters. It simply returns the value of the difficulty variable that is held privately in the puzzle.

Parameters:
None

Return Value: int

3.7 GetCell()

THe GetCell() function takes in a row index and a column index and returns a pointer to that cell in the puzzle.

Parameters:
int row: Row index of cell int col: Column index of a cell

Return Value: Pointer to a Cell object.