model

UW C++ Course Project 1

Author: Hugh Tay

Date: 2016-03-25

Table of Contents

Overview

In this project there are 2 classes, the Solver class and the InOut class, to allow for separation of duties for functions with a role towards solving the Sudoku puzzle, and functions with a role towards handling user inputs/outputs.
.

1. Class Diagrams

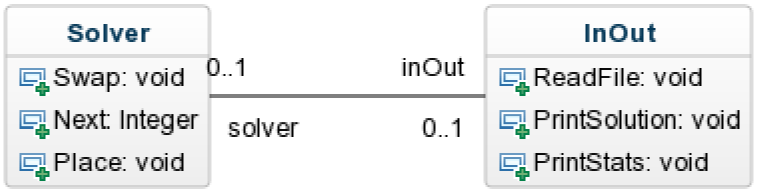

Figure 1. class-diagram Diagram

Referenced Elements

- Class InOut — see "InOut" definition
- Class Solver — see "Solver" definition

1.1. Model model

In this project there are 2 classes, the Solver class and the InOut class, to allow for separation of duties for functions with a role towards solving the Sudoku puzzle, and functions with a role towards handling user inputs/outputs.
.

1.1.1. Class InOut

This class will be used to handle inputs from file streams (using a program-specified file name) and check that the format of the data contents follow the prescribed format, as well as outputs to the standard (console) stream for users to view the solution. .

Attributes

- ReadFile : void[1] — see "void" definition

  Description

  Reads the file, checks the format of the contents, sends the parsed contents to the Solver class if format valid, and informs user of problems in format if invalid. .

- PrintSolution : void[1] — see "void" definition

  Description

  Prints the solution to the standard ouput stream in plain text. .

- PrintStats : void[1] — see "void" definition

  Description

  Prints statistics that might be informative to the user in terms of (computational) difficulty of Sudoku puzzle presented. .

Associations

- solver : Solver [0..1] — see "Solver" definition

1.1.2. Class void

I have no idea why this UML software thinks *void* is a Class rather than a return type.
.

1.1.3. Class Solver

The Solver class will initialize the data structures needed for solving the puzzle in the constructor (the InitEntry() function in the original source code), while deleting the copy-constructor and operator= as it's not expected that anyone would need a deep-copy of the data structures since all the solving is done in 1 class. .

Attributes

- Swap : void[1] — see "void" definition

  Description

  Performs a swap between 2 data structures. .

- Next : Integer[1]

  Description

  Finds the next square on the grid that needs solving. .

- Place : void[1] — see "void" definition

  Description

  Places a solved number in the correct square on the grid. .

Associations

- inOut : InOut [0..1] — see "InOut" definition