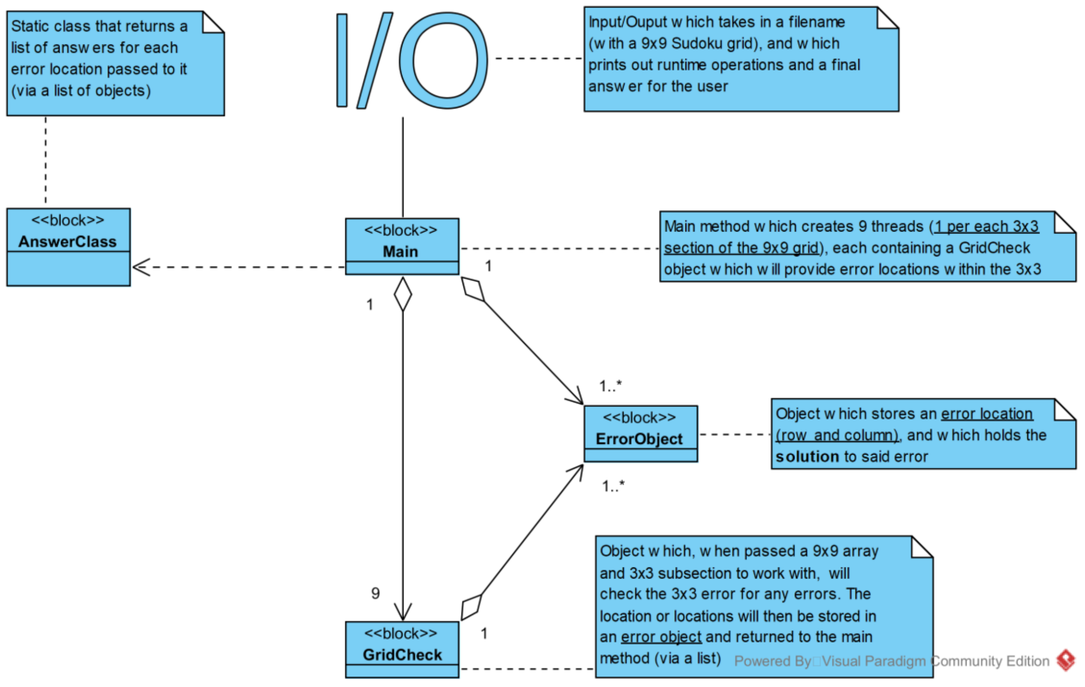
DESIGN DESCRIPTION

The approach I took towards creating a basic system for validating a given Sudoku puzzle involved **two basic assumptions**. **Firstly**, given **Sudoku grids would not be impossible to solve** (i.e. they would not have so many errors within them as to warrant finding interchangeable solutions). **Secondly**, the **grid** chosen for solving **would not have multiple numbers "swapped" out within the same, individual grid** (i.e. one 3x3 section of the sudoku grid having the numbers one through nine in it, but all swapped around). This latter assumption was the primary flaw to my solution validator, however I have attempted to put checks in place to help avert program issues if the second assumption holds false.

The main, basic workflow of the program is as follows. We use nine Linux POSIX Pthreads (inside the C++ main method thread) to search through the grid and discover the locations of errors, location referring to the row and column values. **Each thread does this by running a GridCheck method (**from its own local GridCheck instantiated object**) on a 3x3 section of the full 9x9 grid**. Put simply, it runs **nine checks for each 3x3 grid**, all at the same time **via POSIX threading**. A GridCheck consists of **checking** the **two** (technically three) **sudoku puzzles rules**: each column and row must contain all of the digits 1-9 once, and each of the nine 3x3 sub-grids must contain all of the digits 1-9 once. If the latter rule does not hold true, we check the columns and rows for each of the nine numbers within the section. If we find that one or more of these fail the first rule, we mark the location and insert into an ErrorObject list to be returned to the main method at the end of thread execution.

After all nine threads complete their discovery of errors within the 9x9 grid, we then **run** a **static method** for **answer checking**, within the static AnswerClass. This will look at each individual error location, and will **perform an appropriate algorithm** that avoids collisions with columns or rows that hold **additional** errors (i.e. if the row the error is in has another error in it, we probably shouldn't check it). Then we set the correct answer into the ErrorObject for the main method to return to the user.


Static class that returns a list of answers for each error location passed to it (via a list of objects)
I/O
Input/Ouput which takes in a filename (with a 9x9 Sudoku grid), and which prints out runtime operations and a final answer for the user
<<block>> AnswerClass
<<block>> Main
Main method which creates 9 threads (1 per each 3x3 section of the 9x9 grid), each containing a GridCheck object which will provide error locations within the 3x3
1
1
1..*
<<block>> ErrorObject
Object which stores an error location (row and column), and which holds the solution to said error
1..*
Object which, when passed a 9x9 array and 3x3 subsection to work with, will check the 3x3 error for any errors. The location or locations will then be stored in an error object and returned to the main method (via a list)
9
1
<<block>> GridCheck
Powered By: Visual Paradigm Community Edition