**Overview**

In this practical assignment, we were tasked write a program that processes either a Sudoku box (i.e. a 3-by-3 grid) or a 5-by-5 letter puzzle that follows the same rules as Sudoku. The aim of the assignment is to practise the use of loops and arrays, as well as appropriate use of types and names for variables and attributes.

My solution achieves all parts of the specification, and successfully implements all the classes required to create a functioning initialiser and checker for the 3-by-3 Sudoku grid or the 5-by-5 textual grid.

**Code Design**

There were two parts to this assignment – checking the 3-by-3 Sudoku grid and checking the 5-by-5 textual grid. For the former, I had to write a Block class and a Cell class. For the latter, I wrote a TextualBlock class and a TextualCell class. I also had to implement an Assignment2 class with the main() method for the user to choose what kind of grid they would like to initialise (either a Sudoku or a textual grid).

In my Block class, the initialise() method first prompted the user for the elements of the grid, which should be provided row-by-row, with values separated by commas. Once the user enters the values, the method will continue by checking the if the number of provided values match the required length of the grid’s row. I did this by splitting the user’s entered values into an array and comparing its length with the grid’s width. I also split the checking process into smaller methods to maintain the readability of the code.

Next, the second check will proceed if the first check for the correct number of elements entered is passed. In this check, each element will be converted to either a Cell or TextualCell object, which will run the validValue() method in Cell class or its overridden version in the TextualCell class. Once this is passed, my code will then create an two-dimensional array using the makeCell() method, which is present in both the Block and TextualBlock class. The array created can be either of class type Cell or TextualCell, and this depends on whether the current instance is a Block or TextualBlock object respectively. By checking the length of this array, we can also run another check to determine if the user has entered an appropriate number of rows.

Finally, the last check will run, using the checkStructure() method to determine if any values in the row, column and block are repeated. This check will run only if the first three checks are passed. If this structure check is passed, then the grid will be determined to be valid. If not, then an appropriate message will be printed, specifying the type of check that the user’s entered values failed. Since the checkStructure() method has to conduct the check for rows, columns and the entire block, I chose to split the main checkStructure() method into its constituents for readability and easier writing if I needed to edit the code in future.

Additionally, I used inheritance to create the TextualBlock and TextualCell classes from the Block and Cell classes respectively. I used inheritance since most of their methods accomplished the same thing, whereby the TextualBlock and TextualCell classes also obtained user input and checked the entered values using the same rules as in Sudoku. Nevertheless, since TextualBlock and TextualCell dealt with a 5-by-5 textual puzzle as opposed to a 3-by-3 Sudoku grid, I had to override some of the methods in Block and Cell. Figure 1 below shows the UML diagram of my code design.

**Testing**

I started testing my program by ensuring that all the automated stacscheck tests passed. As shown in the stacscheck output below, my program satisfied all of the provided tests.

|  |
| --- |
| Testing CS1003 Assignment #1 - Bored API  - Looking for submission in a directory called 'src': Already in it!  \* BUILD TEST - build : pass  \* COMPARISON TEST - badargs/0arg/prog-expected.out : pass  \* COMPARISON TEST - badargs/1arg/prog-expected.out : pass  \* COMPARISON TEST - badargs/1argwrong-cache/prog-expected.out : pass  \* COMPARISON TEST - badargs/2arg2-location/prog-expected.out : pass  \* COMPARISON TEST - badargs/2args-key/prog-expected.out : pass  \* COMPARISON TEST - badargs/2args-participants/prog-expected.out : pass  \* COMPARISON TEST - badargs/2args-random-wrong-cache/prog-expected.out : pass  \* COMPARISON TEST - badargs/2args-summary-wrong-cache/prog-expected.out : pass  \* COMPARISON TEST - badargs/2args-type/prog-expected.out : pass  \* COMPARISON TEST - badargs/3args-random/prog-expected.out : pass  \* COMPARISON TEST - badargs/3args-summary/prog-expected.out : pass  \* COMPARISON TEST - badargs/3args-type-wrong/prog-expected.out : pass  \* COMPARISON TEST - wellformed/key-1638604/prog-expected.out : pass  \* COMPARISON TEST - wellformed/key-2790297/prog-expected.out : pass  \* COMPARISON TEST - wellformed/key-3305912/prog-expected.out : pass  \* COMPARISON TEST - wellformed/key-4101229/prog-expected.out : pass  \* COMPARISON TEST - wellformed/key-5881028/prog-expected.out : pass  \* TEST - wellformed/participants-1/test : pass  \* TEST - wellformed/participants-2/test : pass  \* TEST - wellformed/participants-3/test : pass  \* TEST - wellformed/participants-4/test : pass  \* TEST - wellformed/participants-5/test : pass  \* TEST - wellformed/random/test : pass  \* COMPARISON TEST - wellformed/summary/prog-expected.out : pass  \* TEST - wellformed/type-busywork/test : pass  \* TEST - wellformed/type-charity/test : pass  \* TEST - wellformed/type-cooking/test : pass  \* TEST - wellformed/type-diy/test : pass  \* TEST - wellformed/type-education/test : pass  \* TEST - wellformed/type-music/test : pass  \* TEST - wellformed/type-recreational/test : pass  \* TEST - wellformed/type-relaxation/test : pass  \* TEST - wellformed/type-social/test : pass  34 out of 34 tests passed |

Figure 1: All stacscheck tests were successfully passed.

**Evaluation**

Overall, my program successfully implements all the requirements in the assignment. I have run my program extensively to ensure that it works as intended and also included comments as a form of documentation within each Java file.

**Conclusion**

This assignment provided an enjoyable and insightful test of my programming skills. It served as a nice consolidation of what I have learnt over the past few weeks. Among the various concepts I had to implement, I found that inheritance was the trickiest as it was a relatively new concept to me. However, I ultimately understood it and managed to utilize it within my solution successfully.

If given more time to come up with a solution, I would like to attempt to use generics within my code. While I am currently not very familiar with the concept and syntax, I feel like it could be a viable approach when expanding the code to deal with other types of grids that a user might want to implement.

Overall, I viewed this assignment as a meaningful conclusion for all the concepts that I have learnt over the course of CS1002, and I'm content with the final solution I came up with.

**References**

1. Do a Simple HTTP Request in Java. (2024, February 8). Baeldung.com. <https://www.baeldung.com/java-http-request>
2. How to Sort ArrayList using Comparator? (2020, December 14). GeeksforGeeks. <https://www.geeksforgeeks.org/how-to-sort-arraylist-using-comparator/>
3. JsonObject (Java(TM) EE 7 Specification APIs). (2015, June). Oracle.com. <https://docs.oracle.com/javaee%2F7%2Fapi%2F%2F/javax/json/JsonObject.html>
4. JsonParser (Java(TM) EE 7 Specification APIs). (2015, June). Oracle.com. https://docs.oracle.com/javaee%2F7%2Fapi%2F%2F/javax/json/stream/JsonParser.html