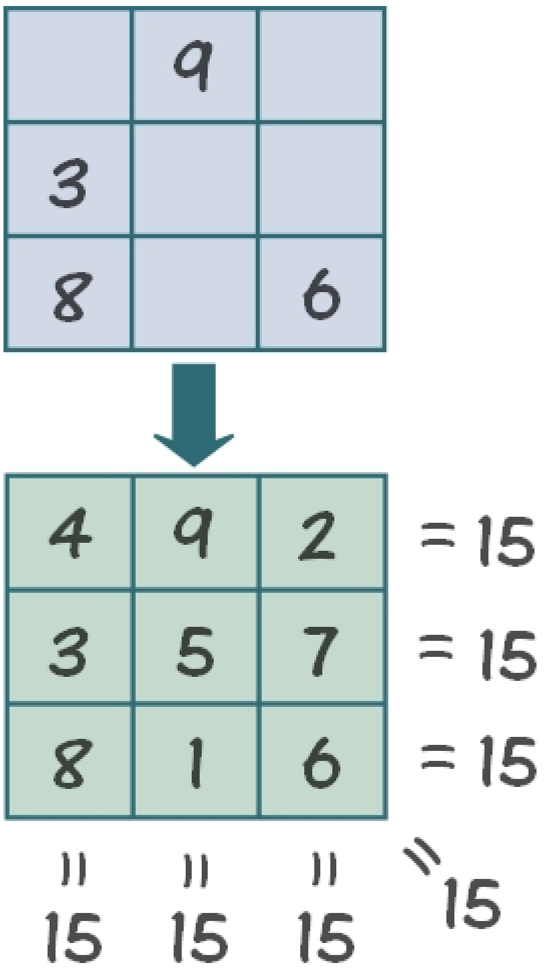
Java Project (Magic Square)

GENERAL OVERVIEW:

Magic squares is an educational games which is widely used by many primary school kids to improve their mathematical abilities. In magic squares we have to fill in a partially filled grid with numbers such that the sum of each row and each column is the same. If the user is able to achieve that, they are progressed to the next round. Solving each round successfully gives the user 5 points. The user with most number of points will be awarded as the "Champion"/"High Scorer".

The below image which has been taken from learn-with-math-games.com gives an example of a round.



FEATURES

- GUI Interface (Swing)

  As per the project requirements of Basics of programming 3 (VIIIAB00) the project will have Swing-based GUI. The app will feature a JTable component for displaying the grid (magic square) and allowing users to input numbers. A menu bar will provide options such as loading a new puzzle, saving progress and validating the puzzle.

- Validation and Logic

  After the users have input the numbers and selected the “validate the puzzle” option from the menu, the program will sum up each row and column to check if the user’s solution matches the correct answer.

- Collections Framework

  A collection (such as ArrayList or HashMap) will be used to store and manage the rows, columns, and diagonals of the magic square for easy manipulation and validation.

- Serialization

  Serialization, which allows storing of object data as bytes will be used to store the puzzle state and scores of the user into a file.

- JUnit

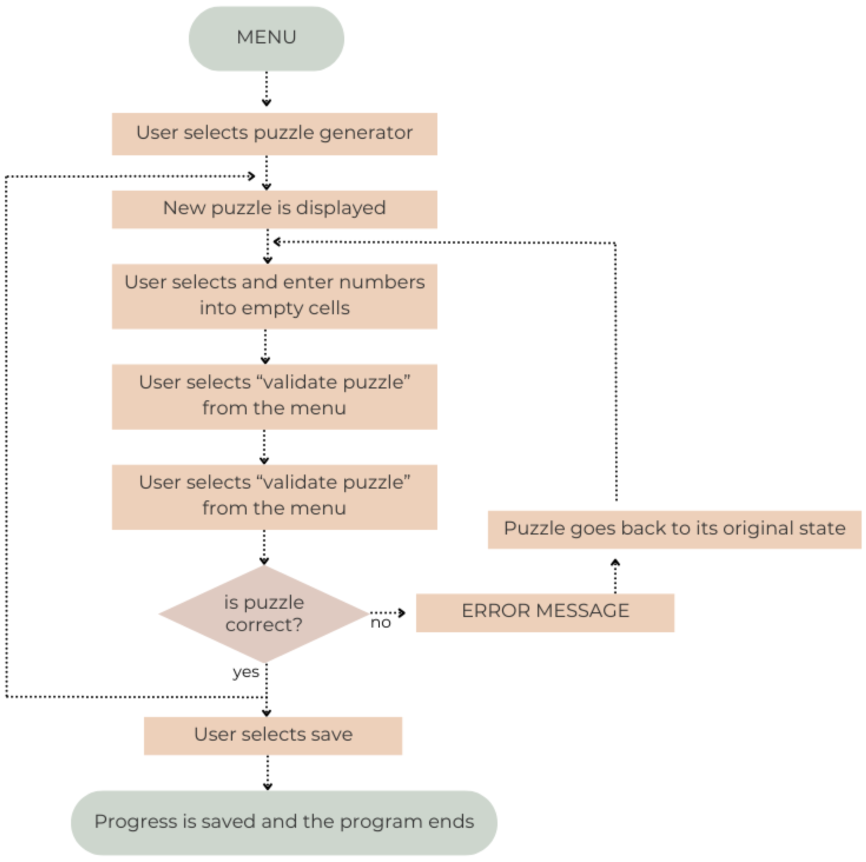
  Unit tests will be written using JUnit to ensure that critical functionalities of the app, such as the generation of magic squares, validation of the puzzle, and file input/output operations, work correctly.

Use Cases

1. Starting new puzzle
   - The user selects the option to start a new puzzle. The app uses the “Puzzle Generator” to create a fresh, incomplete magic square.
   - The new table will appear in the JTable grid for the user to solve.
2. Entering a new number
   - The user selects an empty cell in JTable grid by clicking and then enters a number from their keyboard. If the user enters a letter, it’s corresponding ASCII value will be entered into the grid.

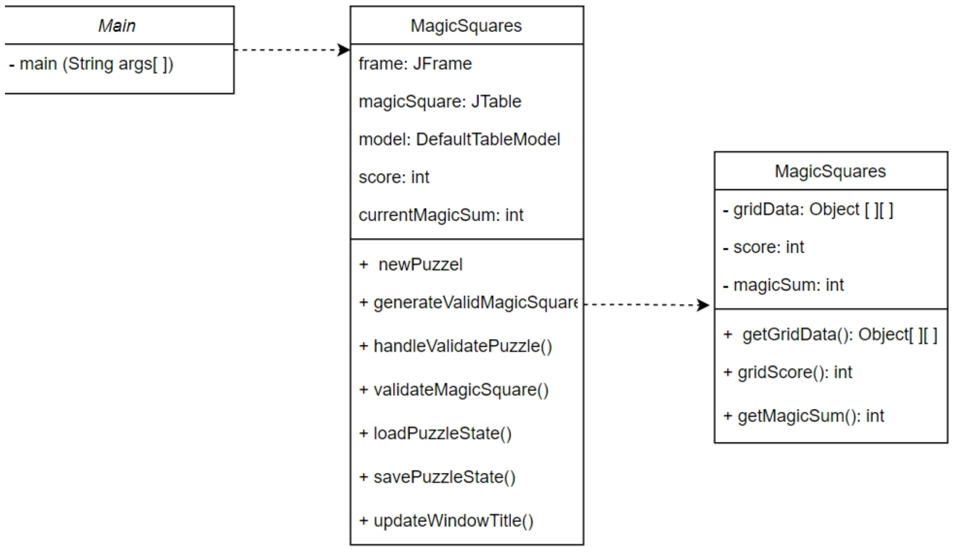
3. Validating
   - User will select “Validate Puzzle” from the menu and if the solution is correct, the program will display a congratulations message and will proceed to the next Magic Square.
   - If the solution is incorrect, the magic square will restart and an error message will be displayed
4. Saving Puzzle
   - The user will select “Save” from the menu. The puzzle and score will be saved into a file with a pop up message indicating a successful saving.
5. Exiting
   - The user will select “Exit” from the menu and before the app is closed, it will prompt the user with a message indicating that all unsaved progress will be lost. It will give user the option to “exit anyway” or “save then exit” the program.

Diagram:

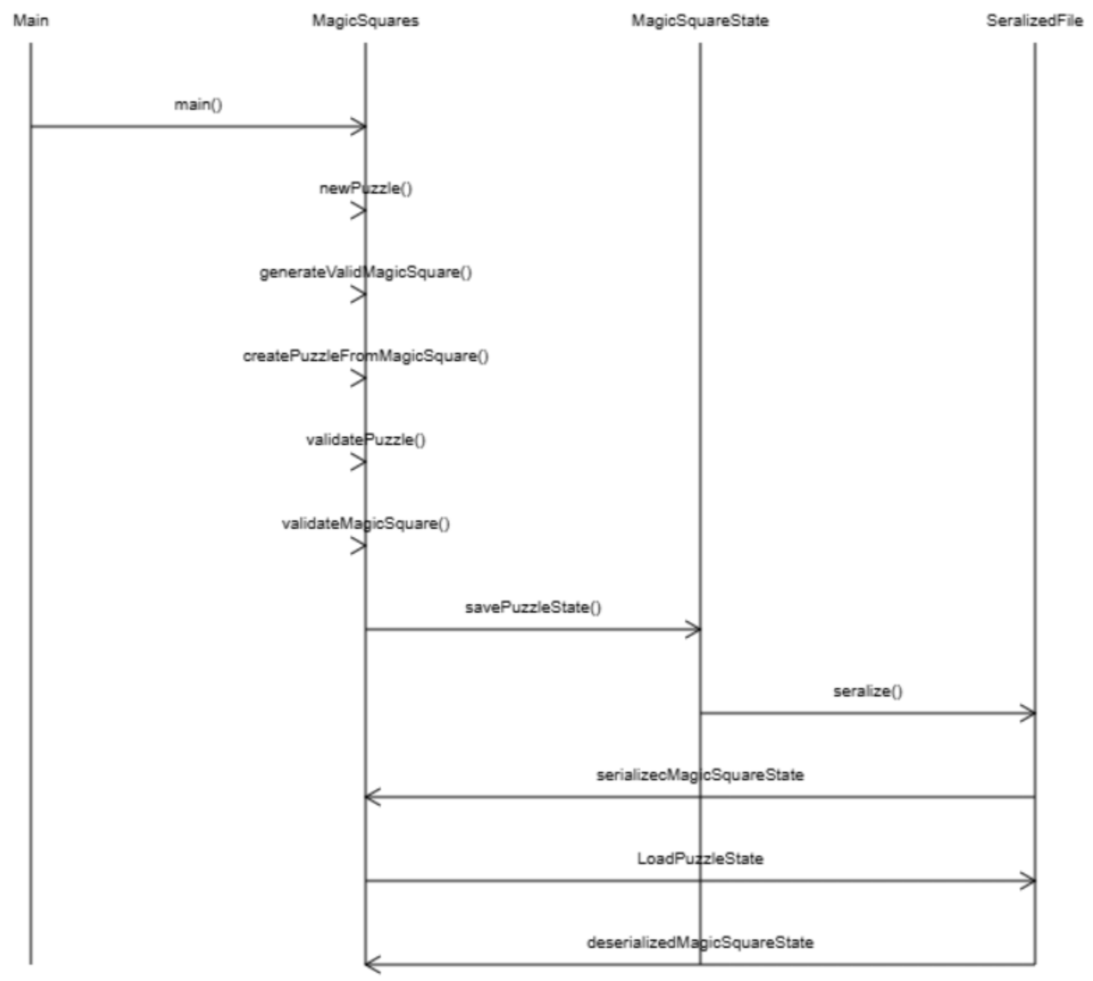


MENU
User selects puzzle generator
New puzzle is displayed
User selects and enter numbers into empty cells
User selects “validate puzzle” from the menu
User selects “validate puzzle” from the menu
is puzzle correct?
no
ERROR MESSAGE
Puzzle goes back to its original state
yes
User selects save
Progress is saved and the program ends

Class Diagram



Sequence Diagram



Test case description

- testValidateMagicSquare()
    - This test case verifies that the magic square validation logic correctly identifies whether a given magic square satisfies the expected conditions (i.e., the sum of all rows, columns, and diagonals must match the magic sum).
    - The test creates a valid 3×3 magic square with magic sum being15.
    - The validateMagicSquare() method is called, which checks whether the sum of the rows, columns, and diagonals matches the expected magic sum.
    - The square is modified by changing o

- ne value (making it invalid), and the test asserts that the validation fails (i.e., false is returned).

- testSerializationAndDeserialization()
    - This test case verifies that the serialization and deserialization of the puzzle state works correctly. The test checks whether the grid data, score, and magic sum are correctly saved to a file and restored when the puzzle is loaded.
    - The test creates a MagicSquares object and generates a magic square.
    - It sets a custom score (10) and magic sum (30).
    - The puzzle state (grid, score, magic sum) is serialized into a file and then deserialized from the file.
    - The test compares the deserialized data (grid, score, magic sum) to the original data to ensure consistency.