**Tetris Solver Application Documentation**

***Introduction***

This document provides an overview and explanation of the Tetris Solver application. The application is designed to solve the problem of arranging Tetris-like pieces on a grid to form a square. The code is written in Java and follows an object-oriented approach.

***Project details***

1. **Classes**
   * **AppController**: Controls the JavaFX application and user interface.
   * **Board**: Represents the game board and handles piece placement logic.
   * **MakeASquare**: Orchestrates the solving process using parallel threads.
   * **Paralleling**: Implements the parallel execution of the solving algorithm.
   * **Piece**: Defines the Tetris-like pieces and their rotations.
   * **PiecesModel**: Manages the Tetris pieces used in the application.
2. **Details**

**AppController**

* + Manages the JavaFX user interface.
  + Initializes buttons and text fields.
  + Invokes the solving algorithm and updates the UI based on the result.

**Board**

* + Represents the game board with a specified size.
  + Utilizes a 2D array to store piece placements.
  + Handles piece rotation, placement, and board validation.

**MakeASquare**

* + Coordinates the solving process using parallel threads.
  + Divides the problem space among multiple threads.
  + Calls the **Square** method to find a solution.

**Paralleling**

* + Implements the **Runnable** interface for parallel execution.
  + Utilizes a lock mechanism to manage shared resources.
  + Executes the solving algorithm on a portion of the problem space.

**Piece**

* + Defines Tetris-like pieces with methods for rotation.
  + Includes a method to retrieve a piece after rotation.

**PiecesModel**

* + Manages the Tetris pieces used in the application.
  + Provides a method to retrieve a specific piece based on its character identifier.

1. **Code Flow**
   * The application initializes the UI in the **AppController** class.
   * The user specifies the number of each Tetris piece through the UI.
   * Upon pressing the "Solve" button, the application creates a **HashMap** of pieces and their counts.
   * The **MakeASquare** class coordinates the solving process using parallel threads.
   * Each thread, represented by the **Paralleling** class, searches for a valid arrangement of pieces on the board.
   * The solution is then displayed or an informational message is shown if no solution is found.