Continue to work with your partner on this assignment. Do not forget to switch roles often.

We are through with manipulating tetrads in the `Tetrad` class. In fact, we are no longer concerned with tetrads themselves but rather with the rows and columns in the `grid` made up of individual blocks.
Recall that the `Grid` class represents a 2-dimensional grid where the game will be played. It will allow us to place blocks into the grid at specific locations. For your reference, below is a summary of the constructor and other methods used in the `Grid` class.

`Grid` class

```
public Grid(int rows, int cols)
public int getNumRows()
public int getNumCols()
public Block get(Location location)
public Block put(Location location, Block block)
public boolean isValid(Location location)
public Block remove(Location location)
```

Exercise 3.0: Full House

An important task is to clear the `grid` of any rows the user completes (fills full of blocks). The first order of business is to go ahead and complete the `Tetris` private helper method `isCompletedRow`.

```
// precondition:  0 <= row < number of rows
// postcondition: Returns true if every cell in the given row is occupied;
//                false otherwise.
private boolean isCompletedRow(int row)
```

Iterate through each `grid` column in the given row (how can you ask the `grid` how many columns there are?), making a temporary new `Location` object for the row and column in question. Call `grid.get(location)` to see if there is a block there or not. If a given location ever returns `null` (nothing there), then the row is not completed. If you get through the entire row and every location has a block, the row is completed.

Run `TetrisTest` to verify that you have completed this exercise before moving on.

Exercise 3.1: Death Row

Next, given a row that is completed, we need to clear that row and then move down (by one row) all of the blocks above that row. Complete the following private helper method `clearRow` in the `Tetris` class.

```
// precondition:  0 <= row < number of rows;
//                given row is full of blocks
// postcondition: Every block in the given row has been removed
//                and every block above row has been moved down one row.
private void clearRow(int row)
```

There are two tasks here. The first is to clear the given row of blocks. As you did for `isCompletedRow`, loop through each column in the `grid`, and create a new `Location` object for the given row and column. Ask the `grid` for the block at that location, and then have the block `removeSelfFromGrid`. (No need to test if a block might be missing because a precondition of the method is that the given row is full of blocks.)

The second task is to loop through every row above the given row to move down the blocks from above. Work backward, one row at a time, starting at the row you are in and moving up. Remember that the row numbers will decrease as you move upwards. Loop through each column in the row, and have each block in the row above `moveTo` the same location below (you will have to make a new `Location` object for the location below). Beware that a block that does not exist cannot move anywhere. In other words, you first have to check to see if a block exists before you can have it move. Also, do not forget to stop before the top row because attempting to have the row about the top row move down will go out of bounds.

This can be a challenging method to write correctly, so think about what you want to do before you write the code for this method.

Run `TetrisTest` to verify that you have completed this exercise before moving on.

Exercise 3.2: Clean House

Now use the above helper methods to complete the following `Tetris` helper method. Remember that you want to start with the bottom row and work your way up to the top. Do not forget that when a row is cleared and everything above moves down one row, you need to check the same row again (the one that just moved down) to see if it should also be cleared.

```
// postcondition: All completed rows have been cleared.
private void clearCompletedRows()
```

Run `TetrisTest` to verify that you have completed this exercise before moving on.

Exercise 3.3: Last Request

Last of all, complete the private helper method `topRowsEmpty` in the Tetris `class.`

```
// returns true if top two rows of the grid are empty;
// false otherwise
private boolean topRowsEmpty()
```

This method checks each column of the top two rows of the grid to see if they are all empty. If the top two rows are not empty, a new tetrad cannot be (legally) added to the game, and the game is over. This method is checked each time a new tetrad is added to the game.

Run `TetrisTest` to verify that you have completed this exercise. Please submit a screenshot of the tester success message for Part – 3.

Your Tetris game is now ready to play. Go ahead and run `Tetris`. A random tetrad should appear at the top of your `Tetris` window.

- You should be able to move it around with the arrow keys.
- Pressing the up arrow will rotate the tetrad clockwise.
- Pressing the space bar will drop it to the floor.
- When a tetrad hits the floor (or another tetrad), a new tetrad will spawn from the top.
- You score 10 points for each new tetrad added to the game.
- The game speeds up with each new tetrad.
- The game is over whenever there is no more room to add a new tetrad.

You will find that the game gets hard quickly. How high can you score before you lose?

Good luck, and have fun!