Assignment 4 – 2048

1. General Info

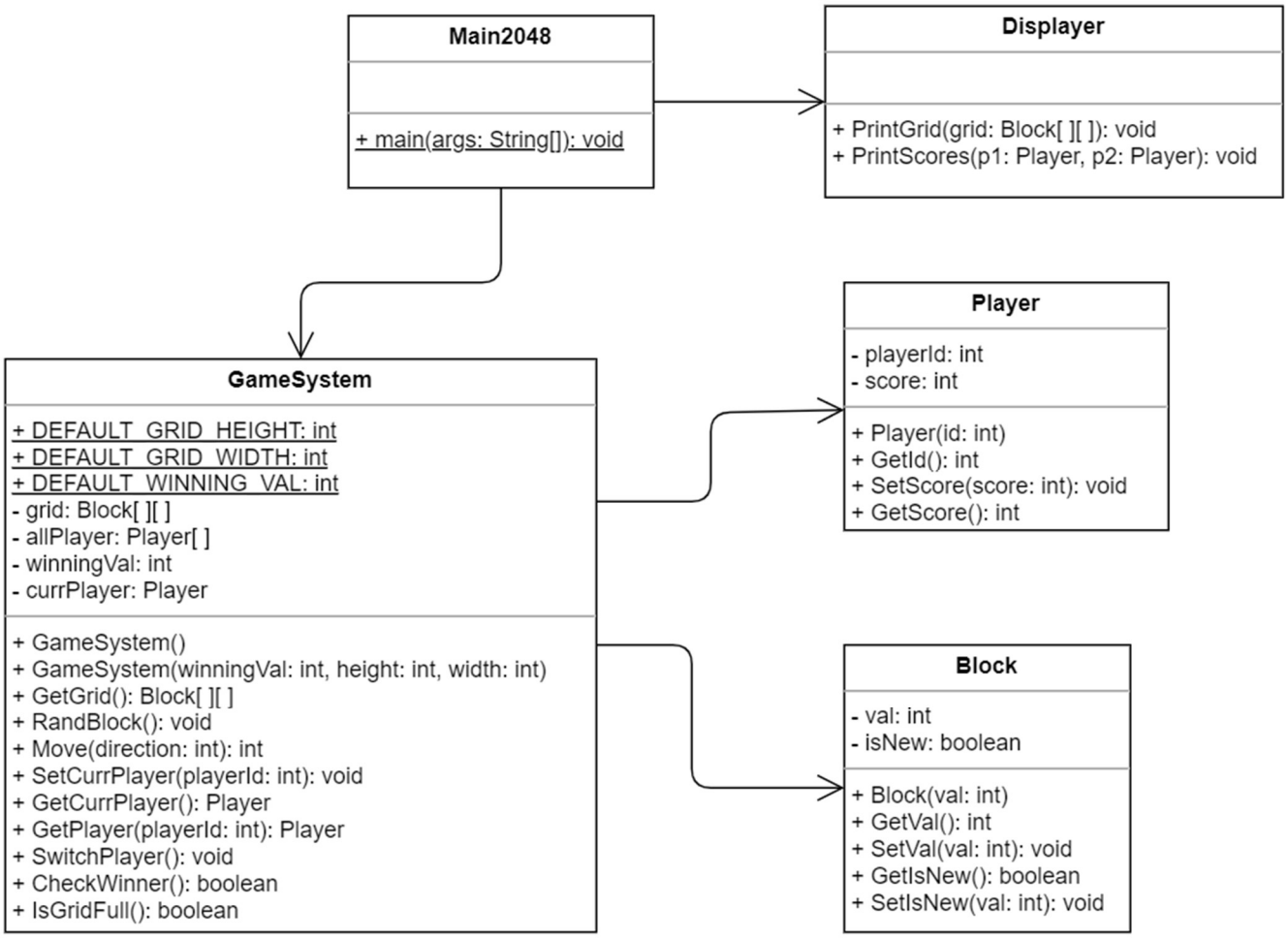
This time you will be programming the game 2048, with some modification. Here are the modifications:

- The game will be played by 2 players. Each player taking turns.
- When 2 blocks are combined, the player who initiated it will receive points.
- Allow user to player in original grid (4 x 4) or in custom grid.

You may play it online to see how the original looks like: https://2048game.com/

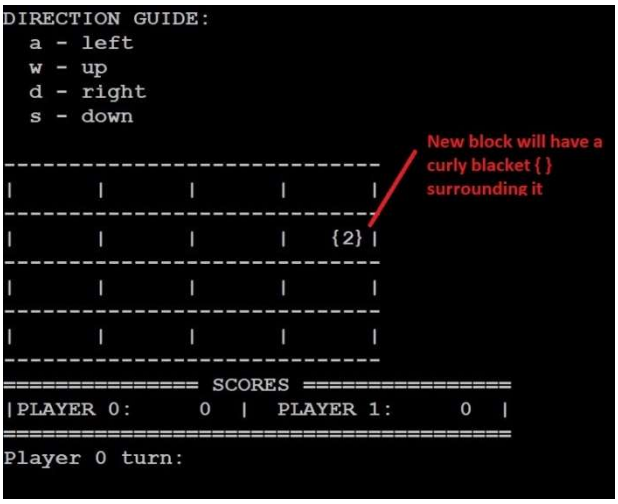
2. UML Diagram

Here's the UML diagram of the program. Your program must follow the same structure. It doesn't contain everything that you need for the program to work.

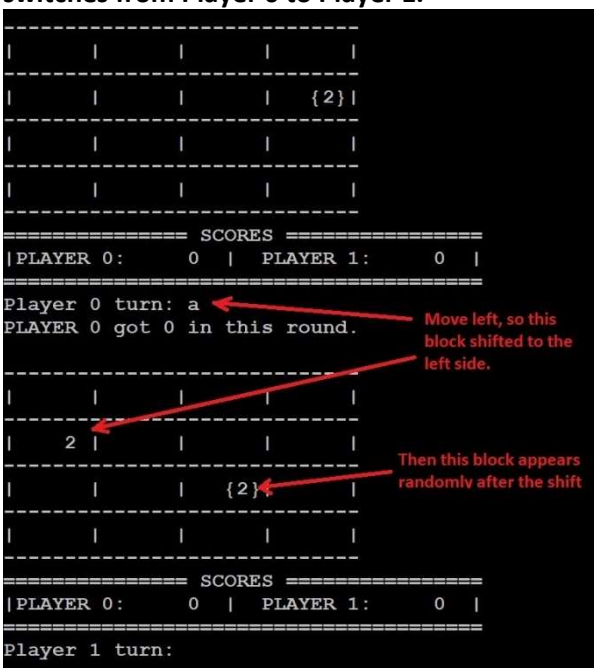


3. User Interface

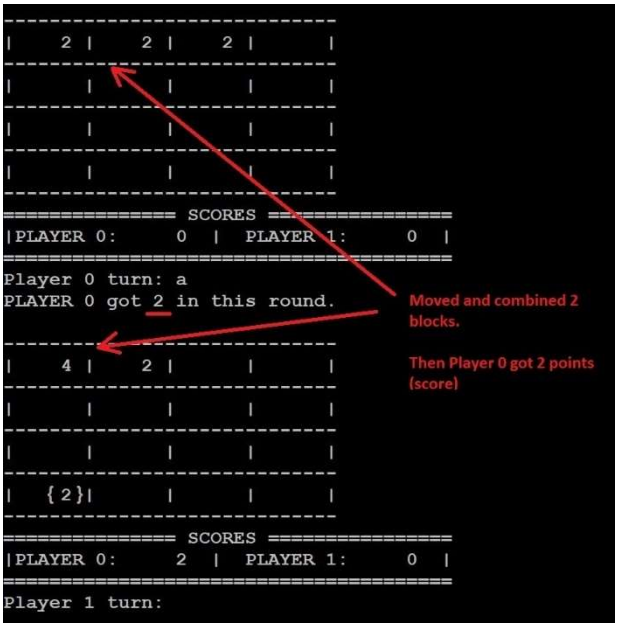
a) Main Screen. New randomized block will have a curly-bracket { } surrounding the number.



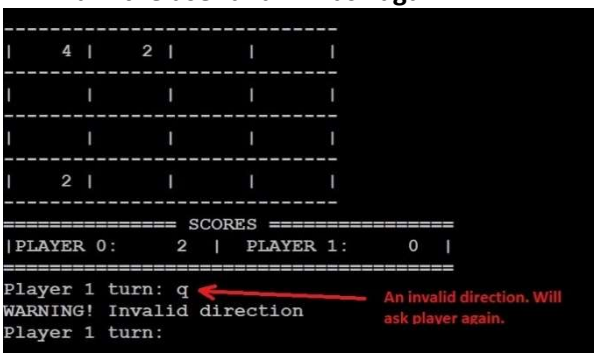
b) Moving left and shifts all the block. Then randomized a new block in the grid. Finally switches from Player 0 to Player 1.



c) Combining 2 blocks and score points



d) If player entered an invalid direction option. It will warn the user and will ask again.



e) When a player has no more moves, or chooses a wrong move.

```
-------------------------------
|    2 |    2 |    8 |    4 |
-------------------------------
|   16 |    4 |    2 |    2 |
-------------------------------
|    8 |    2 |    4 |    4 |
-------------------------------
|    4 |    4 |    2 |    2 |
-------------------------------
================ SCORES ================
|PLAYER 0:    16  |  PLAYER 1:    12  |
========================================
Player 0 turn: w
PLAYER 0 got 0 in this round.

-------------------------------
|    2 |    2 |    8 |    4 |
-------------------------------
|   16 |    4 |    2 |    2 |
-------------------------------
|    8 |    2 |    4 |    4 |
-------------------------------
|    4 |    4 |   {2}|    2 |
-------------------------------
The grid is full! Player 0 lose.
```

f) You may also select a customized game at the beginning

```
Which game to play?
1. Default Game (2048 winning value and 4 x 4 grid )
2. Custom Game
Your Option: 2
Winning Block: 128
Winning Val: 128
Grid Height: 6
Grid Width : 7
DIRECTION GUIDE:
  a - left
  w - up
  d - right
  s - down
```

Specify the Winning Block and Winning Value (points)

Allows a different grid

```
-----------------------------------------------
|      |      |      |      |      |      |      |
-----------------------------------------------
|      |      |      |      |      |      |      |
-----------------------------------------------
|      |      |      |      |      |      |      |
-----------------------------------------------
|      |      |      |      |      |      |      |
-----------------------------------------------
|      |      |  {2} |      |      |      |      |
-----------------------------------------------
|      |      |      |      |      |      |      |
-----------------------------------------------
================ SCORES ================
|PLAYER 0:     0  |  PLAYER 1:     0  |
========================================
Player 0 turn:
```

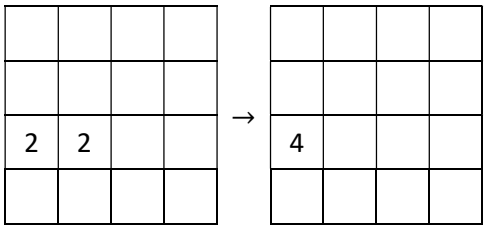
4. How to keep track of blocks and player's score

- Each element in the grid 2D array is a Block object. A null be will assigned to represent an empty position.
- Each Block object keeps track of the block's value.
- Score is stored in Player object.
- The game has a 1D array of size 2 to store the two Player object.

5. How blocks are combined

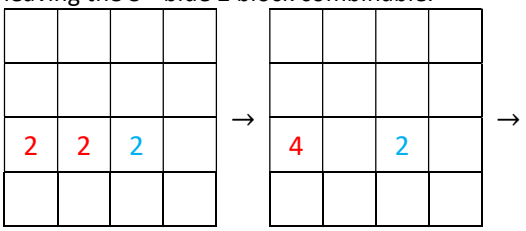
This version of the game is slightly different from the official game, where the combining works a little differently.

Example 1: Shifting Left with 2 of the same blocks



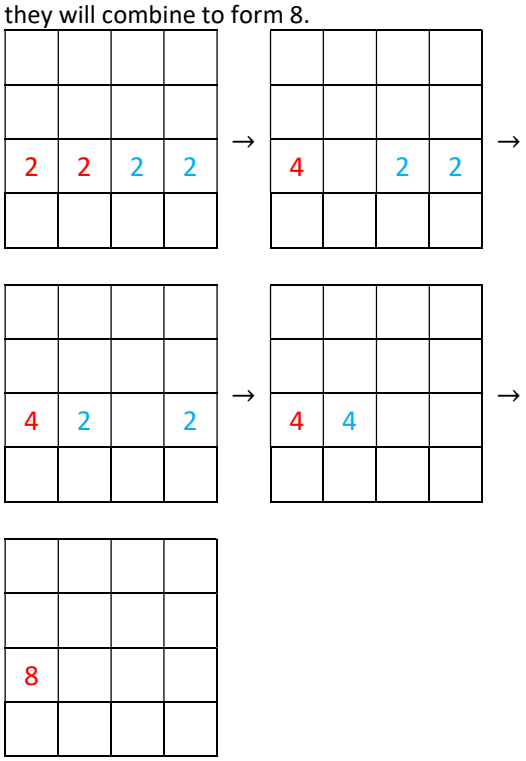
Points = 2

Example 2: Shifting Left with 3 of the same blocks. Left most blocks will combine with the 2nd red 2 as it moves to the left, leaving the 3rd blue 2 block combinable.



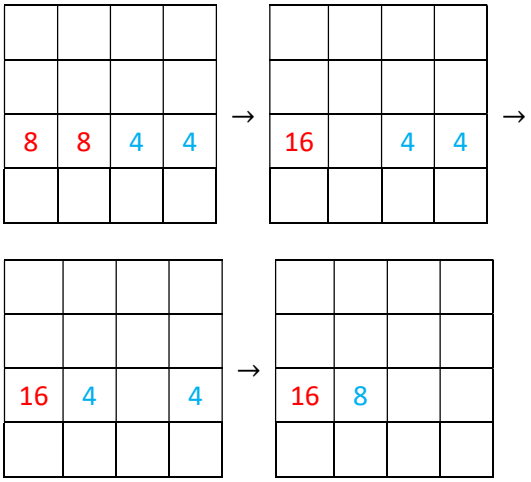
Points = 2

Example 3: Shifting with 4 of the same blocks. The reds will combine, and the blue 2 will shift. The last blue 2 will move along and combine with the first blue 2 to form blue 4. Then they will combine to form 8.



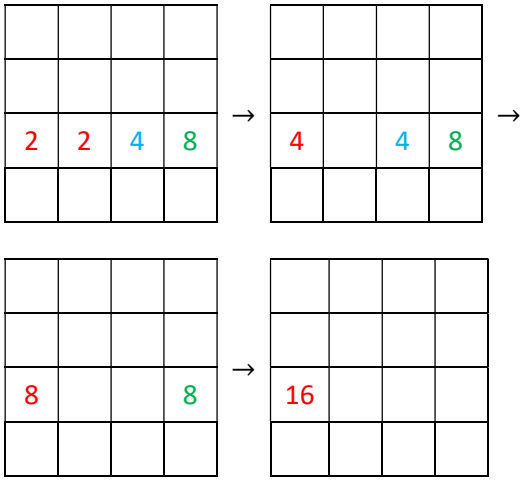
Points = 2 + 2 + 4 = 8

Example 3: Shifting with 2 sets of blocks. The red 8s will combine, and the blue 4 will move to the left. Finally the last blue 4 will move to the left, combining with the first blue 4 to form blue 8.



Points = 8 + 4 = 12

Example 5: Shifting with 4 of the same blocks. The 2nd Red 2 will move to the left and combine with the 1st Red 2. Then blue 4 move to the left and combine to form Red 8. Finally the last Green 8 will move to the left to form Red 16.



Points = 2 + 4 + 8 = 14

6. How Points Work

- A player will earn points by combining blocks.
- When a player combine 2 blocks to form a new block, the player will receive points equivalent to the pre-combine blocks. ***Example***: if the player combines two block of value 8, to form 16. Then the player will receive 8 points.

7. How to Win/Lose a Game

- The player who reaches 2048 points first (Or in a custom game, reach the specified points)
- The player who forces the opponent to have no more moves (Unable to move any block)

8. Requirements

- No **hardcoding**. Must be able to work even if the size of the grid changes. Example of **Hardcoding**:

```
grid[0][0] = grid[0][1];
grid[0][1] = grid[0][2];
grid[0][2] = grid[0][3];
```

- The custom grid could be any size, but must be no smaller than 4x4. ***Example***: 6x4, 5x7, etc.
- Must use primitive 2D array. (Can't use **`List`**, **`ArrayList`**, **`LinkedList`**, etc).
- All major displaying must be done in **`Displayer.java`**
- First player will have a Player ID of 0 and second player will have a Player ID of 1.
- Program must follow the UML diagram.
- Do not change the method header in the template code.
- Must complete all the methods in the template. Must complete each method template and use them throughout the program. May create helper methods only.
- Do not create extra classes.
- When a position has no block, must use **`null`**. Do not create a Block that has a value of 0.

9. Error Checking

- The user's direction input (If user entered a wrong direction, warn the user and let the same user choose a direction again)
- Shifting and combining works as intended.

10. Provided Templates and Files to Submit

Provided Template Classes	**Files to Submit**:	**Files NOT to Submit**:
● **`Block.java`** ● **`Displayer.java`** ● **`GameSystem.java`** ● **`Main2048.java`** ● **`Player.java`**	● **`Block.java`** ● **`Displayer.java`** ● **`GameSystem.java`** ● **`Main2048.java`** ● **`Player.java`**	● Any **`.class`** or **`.java~`**

Note:

- My sample program has around **515** lines of code.
- Working with friends is highly encouraged, and may share ideas. But no sharing of code. There are programs that can check the program's similarity, and I can see who are copying programs.

11. Marking Scheme: Checklist

Application		
Items	Your Mark	Out of
● **`GameSystem()`** constructor is working		2
● **`GetGrid()`** is working		1
● **`RandBlock()`** is working		2
● **`Move()`** is working		19.5
● **`SetCurrPlayer()`** is working		1
● **`GetCurrPlayer()`** is working		0.5
● **`GetPlayer()`** is working		0.5
● **`SwitchPlayer()`** is working		1
● **`CheckWinner()`** is working		4
● **`IsGridFull()`** is working		3
● **`PrintGrid()`** is working		4
● **`PrintScores()`** is working		4
Application Total:		**42.5**
Thinking		
Items	Your Mark	Out of
● Applied Encapsulation correctly/efficiently		4
● Created Useful (Non 1 line) Helper Methods and used them correctly/efficiently (minimum 4)		4
● Methods are modularized (No more than 30-ish lines)		4
● No hard-coding in moving and combining blocks		8
● Used naming conventions correctly		4
● Following Front-End & Back-End programming style		4
● Followed UML diagram design		4
Thinking Total:		**32**
Communicaiton		
Items	Your Mark	Out of
● Contain Comments in code		4
● Have **JAVA DOC** for Helper Methods that you created		4
● Code Indentation		4
● Clear User Interface		4
Communication Total:		**16**