4x4 board object array
Objects have an associated integer and an associated image
4x4 boolean board placement array

One dimensional mapping
For the 12th item on the board
X coordinate = 12%4
Y coordinate = 12/4

Weighted Chance should give a randomly generated tile a 90% chance of being a 2 and a 10% chance of being a 4

Create a random object in the array for each time an arrow key is pressed

```
 public void CreateRand() {
                    if(isFull()) {//improving the efficiency adding this
line at the top rather than stack recursions
                        return;
                    }

                    int rand;
                    int x,y;
                    do{

                        rand = (int)Math.floor(Math.random() * 16);
                        x = rand%4;
                        y = rand/4;

                    }while(board[x][y].value != 0);

                    board[x][y].value = weightedChance();
                }
```

64x64 (pixels) tiles
256x256 board

Each Tile must be rendered :

```
 public void rendering(Tile t) {
                    switch(t.value) {
                        case 2: t.image = "Assets/2048 block 2.png";break;
                        case 4: t.image = "Assets/2048 4 block.png";break;
                        case 8: t.image = "Assets/2048 block 8.png";break;
                        case 16: t.image = "Assets/2048 Block
16.png";break;
                        case 32: t.image = "Assets/2048 32
block.png";break;
                        case 64: t.image = "Assets/2048 64
block.png";break;
                        case 128: t.image = "Assets/2048 128
block.png";break;
                        case 256: t.image = "Assets/2048 256
block.png";break;
                        case 512: t.image = "Assets/2048 512
block.png";break;
                        case 1024: t.image = "Assets/2048 1024
block.png";break;
                        case 2048: t.image = "Assets/2048 2048
block.png";break;
                        default: t.image = "Assets/2048 empty
tile.png";break;
                    }
                }
```

Up, down, left,right functionality

```
//works in a top-down manner
                //for each column
                /*
                 * [0][0][2][0]
                 * [0][0][0][0]
                 * [0][0][2][0]
                 * row 1 col 3 '2' value will not move, but the row 3 col
3 '2' value will and will also merge with the one above
                 *Multimerging? what if two '2's combine to make 4
underneath an existing 4? Should those 4s directly merge?
                 */
```

```
                    public void up() {
                        for(int x = 0; x < board[0].length; x++) {
                            for(int y = 1; y < board.length; y++) {

                                if(board[y][x].value != 0) {
                                    int k = y;
                                    while(k > 0 && board[k-1][x].value ==
0) {
                                        board[k-1][x].value =
board[k][x].value;
                                        board[k][x].value = 0;
                                        k--;
                                    }
                                    //if the while loop terminated because
the value above the current one equal to it, we want to merge them
                                    if(k > 0 && board[k-1][x].value ==
board[k][x].value) {
                                        board[k-1][x].value *= 2; //no
need for a merge function (faster)
                                        board[k][x].value = 0;
                                    }
                            }
                        }
                }

        }
                    public void down() {
                        for(int x = 0; x < board[0].length; x++) {
                            for(int y = 0; y < board.length-1; y++) {
                                if(board[y][x].value != 0) {
                                    int k = y;
                                    while(k < board.length-1 &&
board[k+1][x].value == 0) {
                                        board[k+1][x].value =
board[k][x].value;
                                        board[k][x].value = 0;
                                        k++;
                                    }
                                    //if the while loop terminated because
the value above the current one equal to it, we want to merge them
```

```
                                    if(k < board.length-1 &&
board[k+1][x].value == board[k][x].value) {
                                        board[k+1][x].value *= 2; //no
need for a merge function (faster)
                                        board[k][x].value = 0;
                                    }
                            }
                        }
                    }
                public void left() {
                        for(int y = 0; y < board.length; y++) {
                            for(int x = 1; x < board[0].length; x++) {
                            if(board[y][x].value != 0) {
                                int k = x;
                                while(k > 0 && board[y][k-1].value ==
0) {
                                    board[y][k-1].value =
board[y][k].value;
                                    board[y][k].value = 0;
                                    k--;
                                }
                                //if the while loop terminated because
the value above the current one equal to it, we want to merge them
                                if(k > 0 && board[y][k-1].value ==
board[y][k].value) {
                                    board[y][k-1].value *= 2; //no
need for a merge function (faster)
                                    board[y][k].value = 0;
                                }
                            }
                        }
                    }
                public void right() {
                        for(int y = 0; y < board.length; y++) {//rows
                            for(int x = 0; x < board[0].length-1; x++)
{
                            if(board[y][x].value != 0) {
                                int k = x;
```

```
                                    while(k < board[0].length-1 &&
board[y][k+1].value == 0) {
                                        board[y][k+1].value =
board[y][k].value;
                                        board[y][k].value = 0;
                                        k++;
                                    }
                                    //if the while loop terminated because
the value above the current one equal to it, we want to merge them
                                    if(k < board[0].length-1 &&
board[y][k+1].value == board[y][k].value) {
                                        board[y][k+1].value *= 2; //no
need for a merge function (faster)
                                        board[y][k].value = 0;
                                    }
                            }
                        }
                    }
                }
```

Images: