MARMARA ÜNİVERSİTESİ

1883

2048 GAME

Java Project #1

Faculty of Electrical and Electronic Engineering

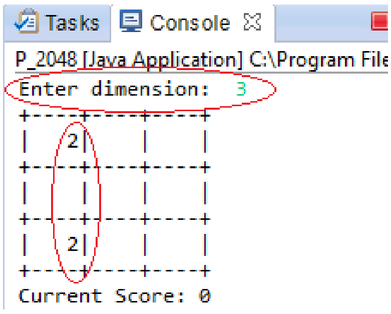
DATE: 28/12/2014

2048 is a game which will make you an addictive to it for sure. With this program, it gains a seamless interface to get its players to have a different and a joyful game experience.. Thanks to its developers.

P_2048 Class

First of all, we defined ‘grid’, ‘replay’, ‘arr’, ‘highscore’ variables and arrays as static in order to reach them from everywhere and to make our changes permient. Also we defined our scanner object ‘sc’ as static to make it easy to use it in every method without defining it again and again.

In main method, first we assign a board (a two dimensional array), which comes from ‘newGame’ method, to the array ‘arr’. We chose 3 as an example in this figure below:

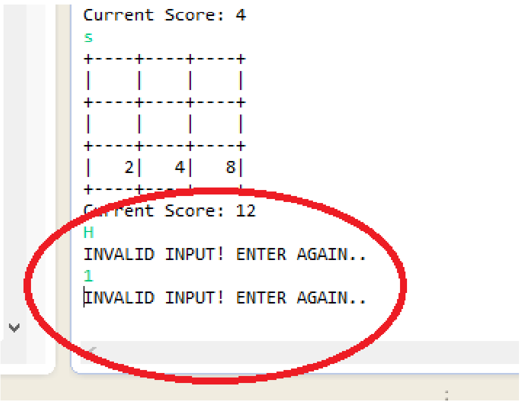


```
Enter dimension:  3
+----+----+----+
|   2|    |    |
+----+----+----+
|    |    |    |
+----+----+----+
|   2|    |    |
+----+----+----+
Current Score: 0
```

This newGame method takes an input from the user and assigns it to the grid, which specifies the dimensions of the board. After that it creates two random numbers (chooses between 2 or 4) for the beginning of the game and prints it to the console. (It generated ‘2’ twice for this time in the example figure above.)

‘While’ loop provides continuity as long as ‘replay’ variable stays ‘y’.

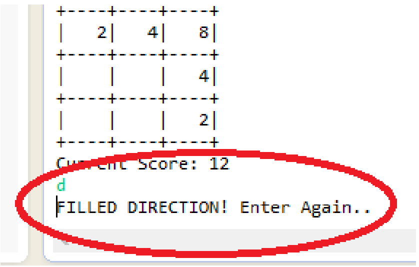
In that while loop, we create a copy of the actual board to the ‘copyArr’, in order to use it later in the background and make sure that everything is properly working. Later on, we ask an input from the user for using it as direction, therefore switch cases decides which section will be executed. User is only allowed to enter one of the characters w,a,s,d otherwise gets an error message and requires an input again. Here is the figure:



```
Current Score: 4
s
+----+----+----+
|    |    |    |
+----+----+----+
|    |    |    |
+----+----+----+
|   2|   4|   8|
+----+----+----+
Current Score: 12
H
INVALID INPUT! ENTER AGAIN..
1
INVALID INPUT! ENTER AGAIN..
```

* ‘SlideTable’ is a class that will be explained later on this report.

‘boardsEqual’ part provides us to control if the user’s movement makes change or not. To do that, we compare our copy array to the actual board and if it’s the same, any random number won’t be appeared on the board. (Copy array holds the one-step-back board). If ‘boardsEqual’ stands still false, random number will be generated onto the board. But if it changes to true, board will remain the same and user will get a warning like “Filled direction, try again”. Here is the example:



```
+----+----+----+
|   2|   4|   8|
+----+----+----+
|    |    |   4|
+----+----+----+
|    |    |   2|
+----+----+----+
Current Score: 12
d
FILLED DIRECTION! Enter Again..
```

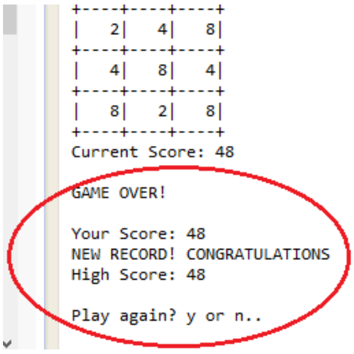
It couldn’t move right..

‘showboard’ is a method which prints the borders of the table. In the first part of this method, it will be determined if the block is going to printed empty. To do that, method searches for the every single element in the board (held in an array) and if the member is zero, it prints empty space into the block. If there exists a number, it simply prints it out into the block. Rest of the codes in this method are used to print borders as well as to lean numbers to the right-hand side, just for keeping the board in shape.

‘RandomAt’ is another method used for generating random numbers into random locations of the board. It creates random numbers to be index to have the random numbers placed in it and by the while loop at the top, it finds another index number if the one is full. After finding an empty place, it generates 2 or 4 and it’s more likely to generate 2 than 4. (it approximately creates 66% two and 33% four.)

After all, board array sent to the method ‘GameOver’, in order to check if there is no move left and accordingly finish the game. At that point, our ‘copyArr’ plays an effective role as we try every possibility on it. Basically, we apply every move at the background and check out whether there is any change or not. If there is, ‘Slideable’ returns true and it means game is not over. But if there is no movement possibility ‘Slideable’ returns false and this simply means that game is over.

At the last part of the P_2048 class, program asks user if he/she want to replay after the appropriate announcements (results). User should enter y for yes and n for no.

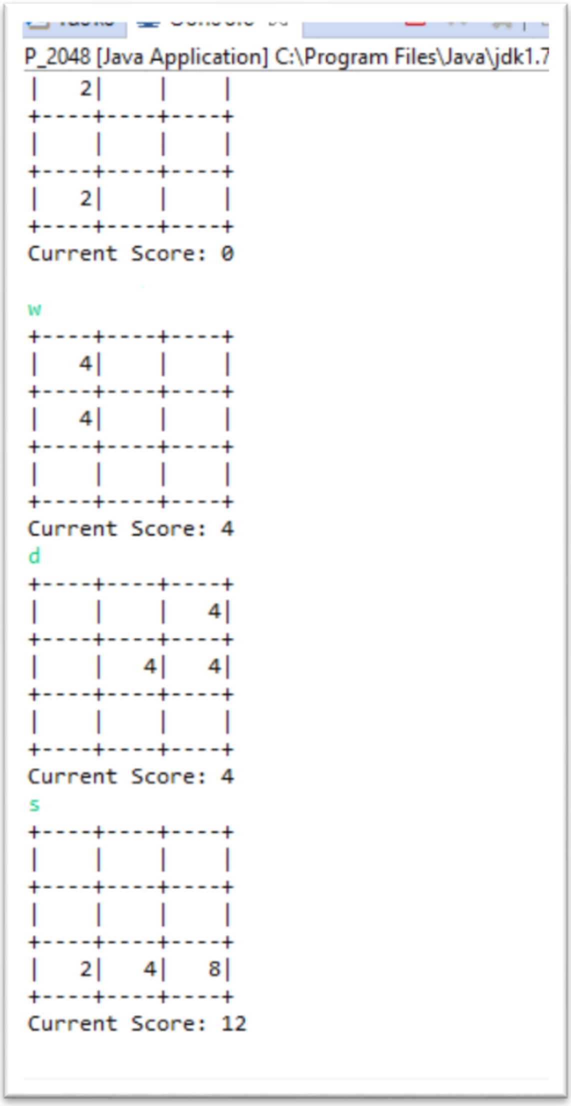


```
+----+----+----+
|   2|   4|   8|
+----+----+----+
|   4|   8|   4|
+----+----+----+
|   8|   2|   8|
+----+----+----+
Current Score: 48

GAME OVER!

Your Score: 48
NEW RECORD! CONGRATULATIONS
High Score: 48

Play again? y or n..
```

SlideTable Class

This class contains the codes that will make the changes on the board. It implies 'Align_Left', 'Sum_Left', 'Align_Right', 'Sum_Left', 'Align_Up', 'Sum_Up', 'Align_Down', 'Sum_Down' methods. The methods starting with "Align" shifts the board to the desired direction and the methods starting with "Sum" adds the same numbers which are side by side. For example, if the input is "w", which means player wants to prop numbers to the upward, 'Align_Up' starts to check from top to the bottom and places the numbers when there exists a zero in the block (That means it's empty). Then 'Sum_Up' takes the turn and adds the same numbers, by multiplying one of them with two. Others will be just moved upward.

Here is a snap shot of the movements:



```
P_2048 [Java Application] C:\Program Files\Java\jdk1.7
|   2|    |    |
+----+----+----+
|    |    |    |
+----+----+----+
|   2|    |    |
+----+----+----+
Current Score: 0

w
+----+----+----+
|   4|    |    |
+----+----+----+
|   4|    |    |
+----+----+----+
|    |    |    |
+----+----+----+
Current Score: 4
d
+----+----+----+
|    |    |   4|
+----+----+----+
|    |   4|   4|
+----+----+----+
|    |    |    |
+----+----+----+
Current Score: 4
s
+----+----+----+
|    |    |    |
+----+----+----+
|    |    |    |
+----+----+----+
|   2|   4|   8|
+----+----+----+
Current Score: 12
```