Assignment 4

Generated by Doxygen 1.9.1

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Namespace Documentation

3.1 Package src

Classes

- class TileT
- class BoardT
- class View

  This is where we communicate with the user! This file takes the role of a controller as well as a View module.

3.1.1 Detailed Description

Author: Jishan Sharif Revised: 12/04/2021

Description: Tile Object

Author: Jishan Sharif Revised: 12/04/2021

Description: Board Model

Author: Jishan Sharif Revised: 12/04/2021

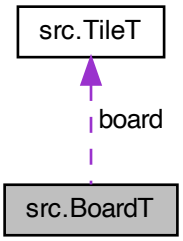
Description: Game simulation here

Chapter 4

Class Documentation

4.1 src.BoardT Class Reference

Collaboration diagram for src.BoardT:



Public Member Functions

- BoardT ()

  We initialize the board where each tile is of type TileT we initialize a board of size 4 ∗ 4. This is done by creating a 2-D array. For each row and column, we first initialize each tile to 0. We then insert two random values anywhere across the board.
- int getScore ()

  This method gets called when we wish to view the score of the user.
- TileT[ ][ ] getBoard ()

  This method gets called when we wish to view the board at the moment.
- void **setBoard** (TileT[ ][ ] board)
- void moveRight ()

  Simulating the event where we move the tiles right @detail We need to move each non empty tile to the right. Iterate through each row, if you see a non empty tile, shift it to the end of that row. For example if a value is in 0,0 move it to 0,3, if 0,3 is taken, move it to 0,2,if 0,2 is taken, move it to 0,1. if All of them are taken. Don't move tile, Go to the next row.
- boolean noMoreMoves (TileT[ ][ ] board)

This checks if there are any more moves possible in the game @detail The first thing we check here is if the board is completely filled. If the board has a value 0, we return false, if the board is completely filled, we check if any merges can be done.

- void moveLeft ()

  Simulating the event where we move the tiles left @detail We need to move each non empty tile to the left. Iterate through each row, if you see a non empty tile, shift it to the beginning of that row. For example if a value is in 0,3 move it to 0,0, if 0,0 is taken, move it to 0,1,if 0,1 is taken, move it to 0,2. if All of them are taken. Don't move tile, Go to the next row.
- void Transpose ()

  This method converts a board into a transposed board @detail A transposed board is a board where the rows and columns are swapped. Mathematically we know that transposing a board, moving left, and transposing again is the same as moving up. The same applies with moving down but we simulate moving right.
- void moveUp ()

  Simulating the event where we move the tiles up @detail We need to move each non empty tile upwards Iterate through each row, if you see a non empty tile, shift it to the top of the column. For example if you see a tile in 0,0. Don't do anything If you see a tile in 0,3, move it to 0,0. If 0,0 is taken, move it to 0,1. If 0,1 is taken move it to 0,2.
- void moveDown ()

  Simulating the event where we move the tiles down @detail We need to move each non empty tile downwards Iterate through each row, if you see a non empty tile, shift it to the bottom of the column. For example if you see a tile in 0,0. Move it to 3,0 If you see a tile in 0,3, Don't do anything. If 0,3 is taken, move it to 0,2. If 0,2 is taken move it to 0,1.

4.1.1 Member Function Documentation

4.1.1.1 getBoard()

```
TileT [][] src.BoardT.getBoard ( )
```

This method gets called when we wish to view the board at the moment.

Returns

a board which is a 2-D matrix of type TileT representing the board at the current moment.

4.1.1.2 getScore()

```
int src.BoardT.getScore ( )
```

This method gets called when we wish to view the score of the user.

Returns

a value of type int representing the score of the user.

4.1.1.3 noMoreMoves()

```
boolean src.BoardT.noMoreMoves (
            TileT board[][] )
```

This checks if there are any more moves possible in the game @detail The first thing we check here is if the board is completely filled. If the board has a value 0, we return false, if the board is completely filled, we check if any merges can be done.

Parameters

board	which is a 2-D list of type TileT representing the board

Returns

a boolean to determine whether there are any moves left, hence, if the game is over

The documentation for this class was generated from the following file:

- src/BoardT.java

4.2 src.TileT Class Reference

Public Member Functions

- TileT (int val)

  A tile can contain a positive even value or it can contain a value of 0 A board is made up of 16 tiles.
- void setValue (int val)

  This game will deal with a lot of updating the board. This means we will have to update the Tile values. This method does exactly that.
- int getValue ()

  We would want to know how our tile looks like and what values each tile contains.
- String toString ()

  Print the values in a readable form.

4.2.1 Constructor & Destructor Documentation

4.2.1.1 TileT()

```
src.TileT.TileT (
            int val )
```

A tile can contain a positive even value or it can contain a value of 0 A board is made up of 16 tiles.

Parameters

val	of type int is the initial value given to the Tile, we initially give it a 0

4.2.2 Member Function Documentation

4.2.2.1 getValue()

```
int src.TileT.getValue ( )
```

We would want to know how our tile looks like and what values each tile contains.

Returns

a value of type int representing the value of a specific tile.

4.2.2.2 setValue()

```
void src.TileT.setValue (
            int val )
```

This game will deal with a lot of updating the board. This means we will have to update the Tile values. This method does exactly that.

Parameters

val	of type int is the new value you wish to update the tile with

4.2.2.3 toString()

```
String src.TileT.toString ( )
```

Print the values in a readable form.

Returns

a string representing the value of a given tile

The documentation for this class was generated from the following file:

- src/TileT.java

4.3 src.View Class Reference

This is where we communicate with the user! This file takes the role of a controller as well as a View module.

Static Public Member Functions

- static void **main** (String[ ] args)

4.3.1 Detailed Description

This is where we communicate with the user! This file takes the role of a controller as well as a View module.

The documentation for this class was generated from the following file:

- src/View.java

Index