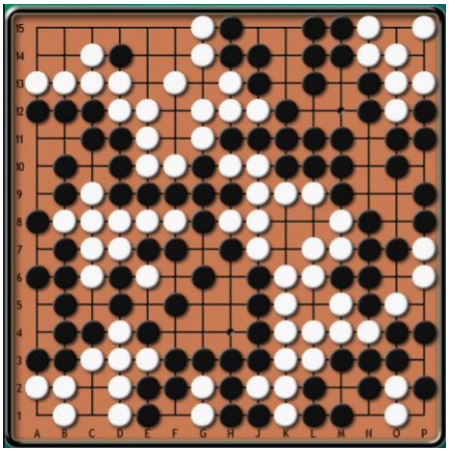
Lab on Array Manipulation

1 Introduction

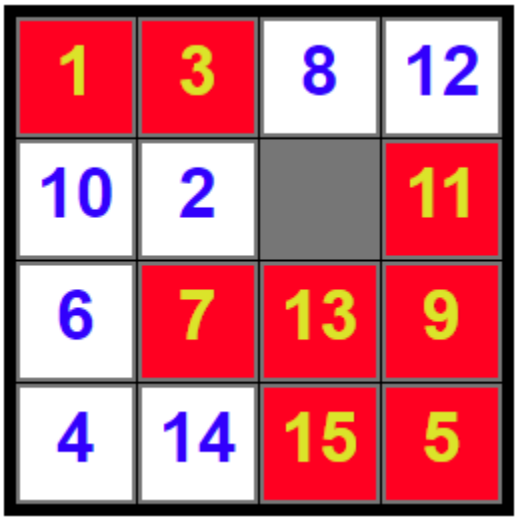
In this lab we will create the foundation for a Java game using 2D arrays. Even though we will NOT create the whole game in this lab, most of the functionality that it needs will be developed herein. After we learn about Java methods, we will migrate this code inside methods and by then we will have a fully functional game. This will happen in the next lab.

Many games found in the internet and at shopping mall stores are what we call board games. Board games can be represented by rows and columns, like Chess, Checkers, Monopoly, Rush hour, etc. Below are images of such kind of games.



In programming world, those games can be developed by using 2D arrays where the first dimension represents the row and the second represents the column of the board.

In this lab we will start creating the “engine” to play the tile game. The tile game is shown here:



It can be played on line at this URL: https://www.thonky.com/fifteen-puzzle/

2 Lab Activities

2.1 The TileGame Project

Create a NetBeans Java project under the name TileGame.

2.2 Create a 2D array

Create an integer 2D array and place the numbers shown in the Figure one in the previous page, replacing the empty stop with a 0.

2.3 Print the tile game

Create a code logic that loops over the 2D array and print it in the terminal window as shown below. Be careful with the alignment. It needs to be printed as is below.

```
 1  3  8 12
10  2  0 11
 6  7 13  9
 4 14 15  5
```

2.4 Find a Tile Location

Create a code logic that loops over the 2D array and finds a tile location. Example: Tile 11 is located at row 1, column 3 (Java starts rows and columns with 0). For this exercise, you can hard-code the tile number that you are looking for in the board. Note: if you set the tile number to Zero, this code would also find the location of the empty spot, right?

2.5 Check if tile can move

Create a code logic that loops over the 2D array and tells if the chosen tile, let's say 11 can move given its current location in the board. Think about when a tile could move in the board. It must have the empty spot as neighbor either in the top, left, right, or bottom. In the specific case of tile 11 the empty spot is its left neighbor, so your code logic would say TRUE, it can be moved. Conversely, your code logic would say FALSE for Tile 10.

2.6 Move Tile

If the code logic in Section 2.5 says TRUE, then create the commands that would move the tile from its current position (Section 2.4) to the empty spot location (also from Section 2.4).

After moving the tile, print the board again so we can see if the tile has really been moved (Section 2.3).