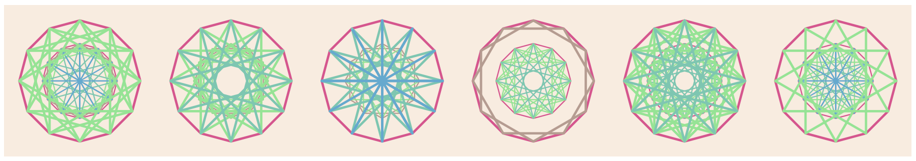
Exercise 6 for Students of Computer Science

Leon Tabak
l.tabak@ieee.org

24 July 2021



We are going to study and then modify a program that Sylvain Saurel wrote.

See here:

- ssaurel/GameOfFifteen
- SSaurel's Channel

Find the article about the 15 puzzle in the mathworld.com website We will include a link to this article in the documentation of your program.

Find a table of x11 colors or HTML colors on the Web. These tables associate names of colors with integers. Choose colors that appeal to you. Declare integer constants in the program to represent these colors. Replace the colors in the program (for example, the color assigned to FOREGROUND_COLOR) with these constants.

You can place Javadoc comments immediately before the definition of a class, instance variable, or class. Begin adding Javadoc comments to the program. Use the author@, version@, param@, and return@ tags. Include a comment that identifies the author of the original version of this program and explains that you are modifying this code.

Look at the constructor for the GameOfFifteen class. It contains calls to methods that a programmer could choose to preface with **this**. For example, you could write...

```
this.setPreferredSize(new Dimension(dimension, dimension + margin)
```

...instead of...

```
setPreferredSize(new Dimension(dimension, dimension + margin)
```

Search on the Web to learn what style other programmers favor: should we include the word **this** or not?

Look at the main() method of the program. Look for answers to these questions in the code, in the Java API documentation on the Web, and on Stack Overflow.

- What is a thread?
- What is the invokeLater() method?
- What is the argument to invokeLater()?
- What is JFrame.EXIT_ON_CLOSE?
- What happens if you replace "Game of Fifteen" with another string in the call to setTitle()?
- What is a layout manager? What is the default layout manager for the JFrame class?
- What is BorderLayout.CENTER?
- What happens if you change the values of the parameters of the GameOfFifteen constructor? (The current values are 4, 550, and 30.)
- Are the calls to pack() and setVisible() necessary?

Now look at the paintComponent() method.

- What is the significance/purpose of Override@? What happens if you leave out that line?
- What does that statement **super**.paintComponent(g) accomplish?
- What is the relationship of the Graphics2D class to the Graphics class?

1

Communicate with me each day. Do not worry if you cannot answer all of my questions or carry out all of the tasks that I suggest. Just keep up a dialogue with me.

I would like you to gain a deeper understanding of the GameOfFifteen program over the next couple of days.

Sylvain Saurel published the code for the Game of Fifteen on GitHub in a Gist. I did not know about Gists. I just a read a little about this feature of GitHub.

I copied Saurel's code into a NetBeans project that I created on my own computer. I created a local Git repository on my computer. Then I created a new remote repository on my own GitHub account. I push the contents of my local repository to the remote repository.

This repository is accessible to you and the rest of the world. It is here at https://github.com/leontabak/gameof15. I added a package statement at the top of the file and a JavaDoc comment that contains a link to Saurel's Gist.

The program shuffles integers. See here to learn more about the algorithm.

Your best friends in this exercise will be the Java API docs and Stack Overflow.

Look at the documentation for the Collections class in the Java API docs. That class includes a method called shuffle (). How could you use that in the GameOfFifteen program?

Look also at the Point class. It models a pair of integers x and y. The GameOfFifteen program refers to the coordinates of tiles with row and column numbers. Might the Point class be a better way of keeping track of where tiles are?

NetBeans tells me (by showing little yellow light bulbs on the left) that Saurel's code could be improved by removing some calls to methods from the constructor for the GameOfFifteen class. These methods include setPreferredSize (), setBackground(), setForeground(), setFont(), and addMouseListener(). These are methods whose definitions could be overridden in a sub-class. Including such calls in the constructor is dangerous. It is better to create a new private method (maybe call it configure ()) and put the calls to those methods there.

A single file contains the whole program. Look for places where you might want to split the big files into several smaller files. That would mean putting some of the code into other classes. For example, you might improve the program by putting the MouseListener code in a separate file.

Look also for names of variables or methods that you might want to replace with other names that more clearly suggest the purpose of a variable or method. Maybe you make the code easier to understand just by abbreviating less: "numberOfTiles" instead of "nbTiles" or "blankPosition" instead of "blankPos."

Look for Boolean expressions in if statements and loops. Maybe we would be wise to put some of that code into new functions? You might, for example, want to take a look at the code that determines the direction in which the program moves tiles.

Ask lots of questions!

2

Here are some more ideas for making small changes to the program and for getting a better handle on what's happening through experimentation.

Try replacing this code. The code is in the drawGrid() method.

```
            // for other tiles
            g.setColor(getForeground());
            g.fillRoundRect(x, y, tileSize, tileSize, 25, 25);
            g.setColor(Color.BLACK);
            g.drawRoundRect(x, y, tileSize, tileSize, 25, 25);
            g.setColor(Color.WHITE);
```

...with this code...

```
            // for other tiles
            int radius = 25;
            g.setColor(getForeground());
            g.fillRoundRect(x, y, tileSize, tileSize, radius, radius);
            g.setColor(Color.BLACK);
            g.drawRoundRect(x, y, tileSize, tileSize, radius, radius);
            g.setColor(Color.WHITE);
```

Run the program. Then change the value of radius to something other than 25. Run the program again. What happens?

What happens if you change Color.BLACK to some other color? What happens if you change Color.WHITE to some other color?

What happens if you add this line of code before the call to drawRoundRect()?

```
            g.setStroke( new BasicStroke(6));
```

What happens if you change that 6 to something else? If not, just search on the Web for \u2713.

3

I left out some text from my last note. I meant to include this:

Make the changes in newGame() that you see here. I have put the call to shuffle () in a comment and assigned **true** instead of **false** to gameOver.

```
    private void newGame() {
            do {
                reset(); // reset in intial state
                //shuffle(); // shuffle
            } while (!isSolvable()); // make it until grid be solvable

            gameOver = true; //false;
    }
```

Run the program. Now you see what a player will see after solving the puzzle. In the definition of drawGrid() you will see "2̆713."

Can you see what that code means? If not, search on the Web with \u2713.

4

The program uses 1-D data structure to model a 2-D game.

The 1-D data structure is an array of integers.

The 2-D data structure is a 4×4 grid.

The program translates between the 1-D model and the 2-D model.

Let's see how that works. We draw the 4 x 4 grid and label each of the 16 tiles with a row number and a column number. We start counting rows and columns at zero. The first row is row #0. The first column is column #0. The last of the four rows is row #3. The last of the four columns is column #3.

(00,00)	(00,01)	(00,02)	(00,03)
(01,00)	(01,01)	(01,02)	(01,03)
(02,00)	(02,01)	(02,02)	(02,03)
(03,00)	(03,01)	(03,02)	(03, 03)

As you see, the row number of the tile in the upper left corner is 0. Its column number is also 3.

The row number of the tile in the lower right corner is 3. Its column number is also 3.

Now let's draw the grid but label each tile with its index number. The index is a position in the 1-D array.

00	01	02	03
04	05	06	07
08	09	10	11
12	13	14	15

Note that the index number is not the number that appears on the screen! It is the position within the 1-D array where the program finds the number that it draws on the screen.

Note also that for each tile: $row \cdot 4 + column = index$.

Also, $row = index/4$ and $column = index\%4$. (This is integer division: $7/4 = 1$.)

5

A process is a program in execution. A process is the basic unit of work on a computer. Your program runs many processes at once. A thread is a "light weight process." Light weight means that the computer need not keep track of as much information about a thread as it must for a process. This is because threads do not have their own memory space, but share space with other threads. You can learn more by searching on Stack Overflow. Java allows us to create several threads in a single process and to make these threads communicate with one another. For example, a programmer might want separate processes to draw the elements of a graphical user interface and to query a database.

Graphics2D extends Graphics. This means that Graphics2D is a subclass of Graphics. A Graphics2D object is a kind of Graphics object—it can do everything that a Graphics object can, and then some more.

super.paintComponent() tells the computer to execute the parent's version of paintComponent() before executing our own version.

The default layout manager for other classes is indeed FlowLayout, but for JFrame the default layout manager is BorderLayout. BorderLayout puts one main big thing in the middle of a window and optionally puts something else above the thing the middle, something below, something to the right, and something to the left. In this case, we just put a JPanel in the middle. We do not have anything above, below, left, or right.

Override@ tells the compiler that we are redefining a method that we inherited from the parent class. The program will still work without this annotation, but its presence will enable the compiler to catch some kinds of errors that we might miss otherwise.

I did not try it on this program, but I think that if you remove the call to setVisible() you will no longer see the game.

Now you know what the parameters for the contructor mean. You can have a 3×3 grid or a 4×4 grid or a $5x5$ grid. You can make the grid fill the window or leave large margins around the grid. You can make a big window or a small window.

6

To finish this project, continue studying the program, experimenting with the program, documenting the program, and changing the program by following some of the suggestions that I have shared with you. We will find a time to meet in a video conference. We will both have the program open on our computers. Together, we will work on the program. I will lead you. I will teach you more about how the program works. You will share with me your understanding. I think that we can finish the capstone in this meeting.

Study how a player moves the blank tile on the board. Where must the player click to make the blank tile move? How far can one click move the tile?

If you do not already know how to write Javadoc comments, please learn. If you cannot find a good guide, ask me.

Take a look at this article about even permutations. Mathworld also has a longer article about permutations.

Here are some other little things to investigate and try:

- You could try using a different font or font weight or font style for the numbers on the tiles.
- You could see if the valueOf() method of the String class and the toString() method of the Integer class are equivalent. The program uses the valueOf() method. Could we use the toString() method? Is there any reason to prefer one over the other? (I do not know.)
- We are using the decimal integers 1 to 15 to label the tiles. We could instead use the hexadecimal digits 1 to F. I think that the Integer class provides a method that would make it easy to do that.
- Maybe we can come up with a way of keeping track of how many times or how far we move the blank tile?

Once again, please do not be shy about asking questions! I am not testing you to learn how much you already know. In this last class, I want to give you just a little more practice learning with and from a partner (that's me!).