GAME FOUR

Subject: **Practical Software Enginiering**

Assaigment 2

Student: **Erblina Jakupi**

Teacher: **Szendrei Rudolf**

Your program should be user friendly, and easy to use. Apply object oriented concepts if it is possible. Multilayer architecture is not required. Use controls to make the GUI of the game. Some assignments require to use several board sizes. Make sure that the window size fits every time to the board of the game.

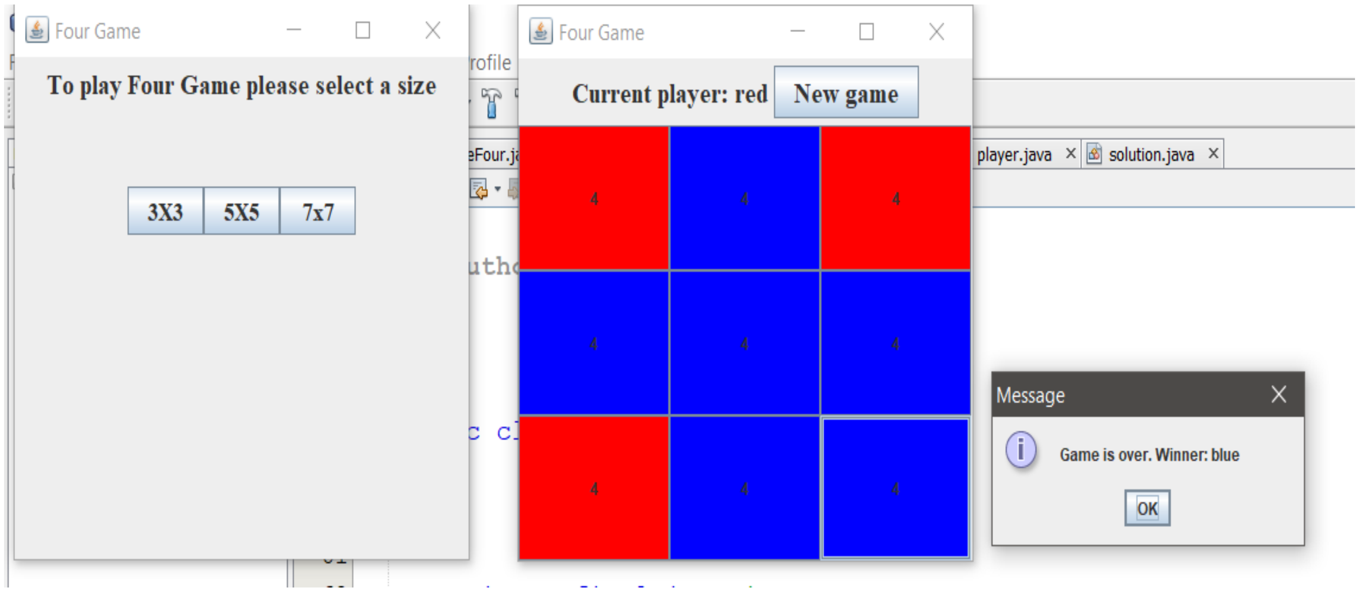
The documentation should contain:

- the description of the exercise,
- the class diagram,
- the short description of each methods,
- and the connections between the events and event handlers.

TASK DESCRIPTION

This a two-player game is played on a board consists of n x n fields, where each field contains a value between 0 and 4. Initially, all the fields contain the value of 0. If a player chooses a field, then the value of the field and its neighbor incremented by one (if the value is less than 4). The player's score represents how many fields did he make to have the value of 4. If a value of a field reaches 4, then the field is colorized with the color of the actual player (red or blue). The game ends, when all fields have the value of 4. The player having the higher score wins. Implement this game, and let the board size be selectable (3x3, 5x5, 7x7). The game should recognize if it is ended, and it has to show in a message box which player won. After this, a new game should be started automatically.

OUTPUT.



IDEA OF THE GAME

-need to create a frame with size (3x3),(5x5),(7x7), after the user have selected one of them

- should have two players with two colors red and blue

LOGIC

-each neighbour boxes are increased by one from their selected neighbour

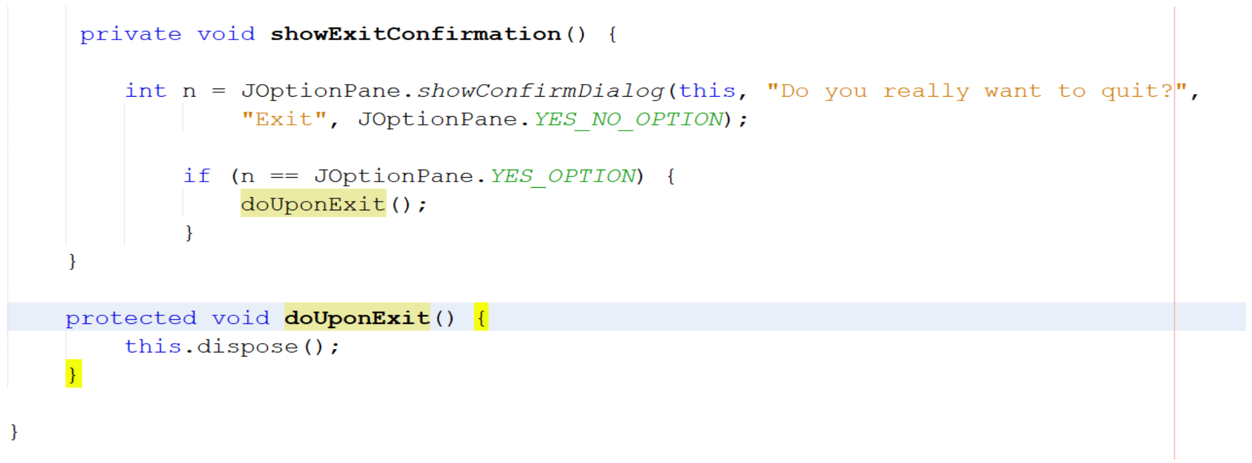
-count each steps for players

-winner is known from which color is dominant in the table

The main class for this project is GameFour class, which created a new object of the class home and set it as visible.

The **base** class is a frame which has a constructor to initialize the parameter of the frame. Besides the constructor it has two other methods. The method showExitConfirmation is extended from the Jframe class , it gets a decision to exit the game or no.

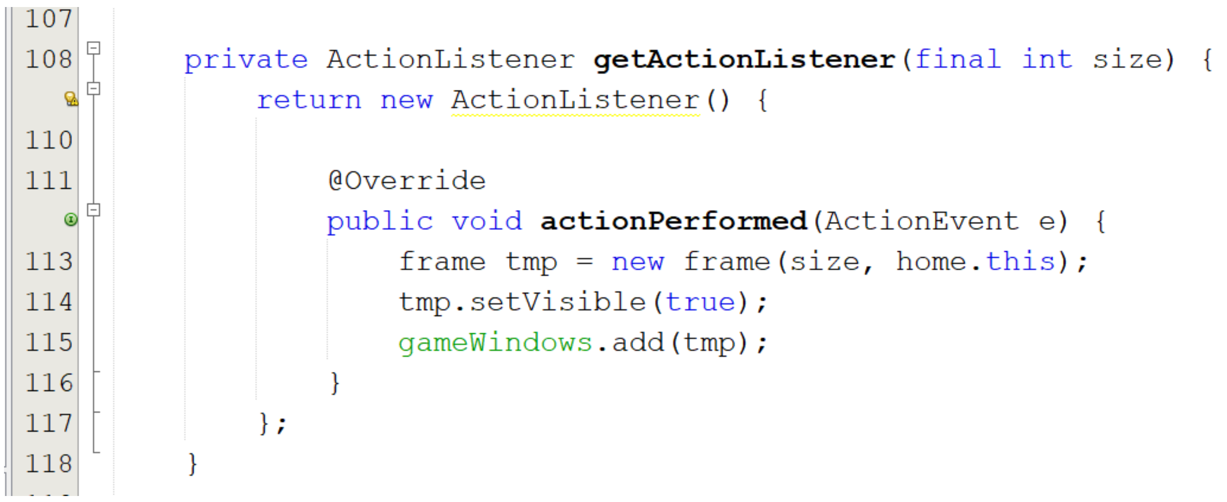
doUponExit is a method which exit when the user select the close button.



```
    private void showExitConfirmation() {

        int n = JOptionPane.showConfirmDialog(this, "Do you really want to quit?",
                "Exit", JOptionPane.YES_NO_OPTION);

            if (n == JOptionPane.YES_OPTION) {
                doUponExit();
            }
    }

    protected void doUponExit() {
        this.dispose();
    }

}
```

The **home** class extends the class base, this class it has three buttons to selected the size of the game and an array of the frame. In constructor class we initialize those attributes of the class. It inherits the method from the class base to exit.

We add an actionListener to each buttons, so when a buttons is clicked it has a size which represents the size of the frame. In this actionListener method I create a new objejct for the frame class and I add those objects to the arraylist of the frames.



```
    private ActionListener getActionListener(final int size) {
        return new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                frame tmp = new frame(size, home.this);
                tmp.setVisible(true);
                gameWindows.add(tmp);
            }
        };
    }
```

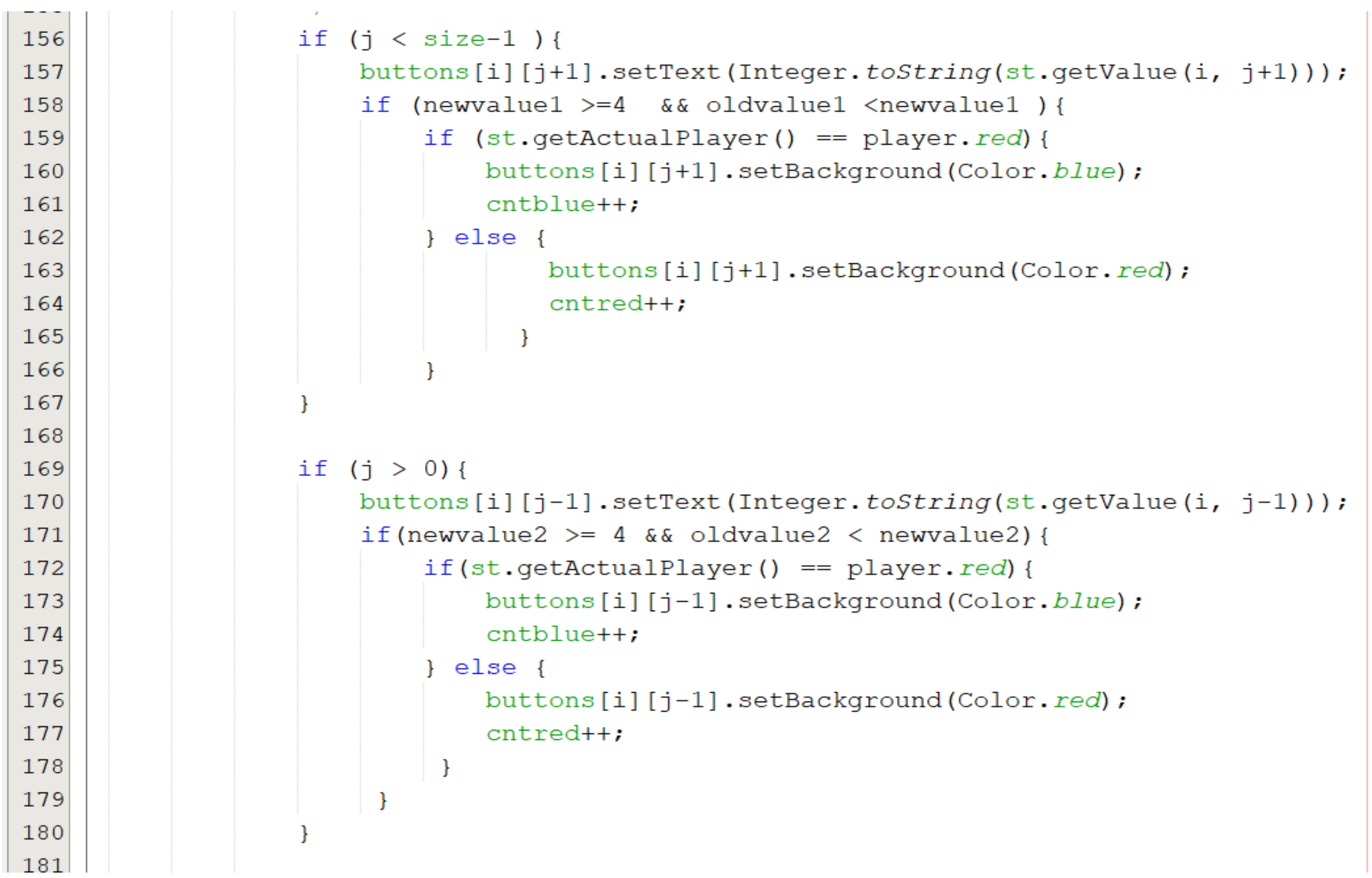
The class which creates the frames for the games is called frame class.

The class **frame** has a object for solution class, and object for home class, an array list with buttons, a size and a cout for red and a cout for blue.

The constructor for the class initialize all those attributes.As parameter the constructor takes the size and the home object. The two dimensional array list is initialized with the number from size. A simple JPanel just too show a text in the frame to Selet a size.

Another panel which will hold the rows and columns for the buttons. This panel should make a GridLayout with size number. In a for loop to add the rows and columns we call the method add buttons which this method creates the table. Inside the constructor we add the content of the frame with getContent() and posite them in places.

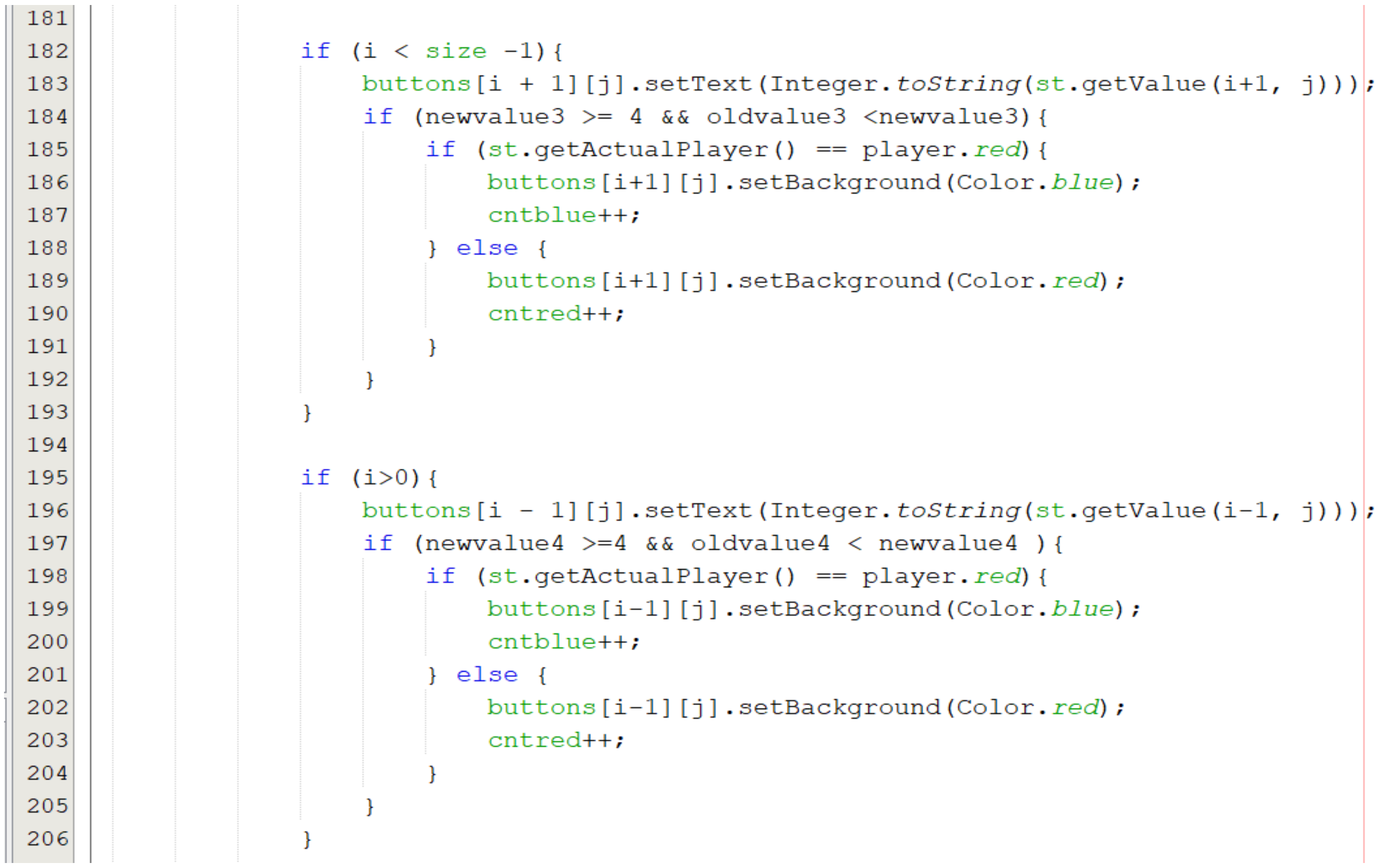
This is all about to hold the errors value for each colums



```
                if (j < size-1 ){
                    buttons[i][j+1].setText(Integer.toString(st.getValue(i, j+1)));
                    if (newvalue1 >=4  && oldvalue1 <newvalue1 ){
                        if (st.getActualPlayer() == player.red){
                            buttons[i][j+1].setBackground(Color.blue);
                            cntblue++;
                        } else {
                                buttons[i][j+1].setBackground(Color.red);
                                cntred++;
                            }
                        }
                }

                if (j > 0){
                    buttons[i][j-1].setText(Integer.toString(st.getValue(i, j-1)));
                    if(newvalue2 >= 4 && oldvalue2 < newvalue2){
                        if(st.getActualPlayer() == player.red){
                            buttons[i][j-1].setBackground(Color.blue);
                            cntblue++;
                        } else {
                            buttons[i][j-1].setBackground(Color.red);
                            cntred++;
                         }
                     }
                }
```

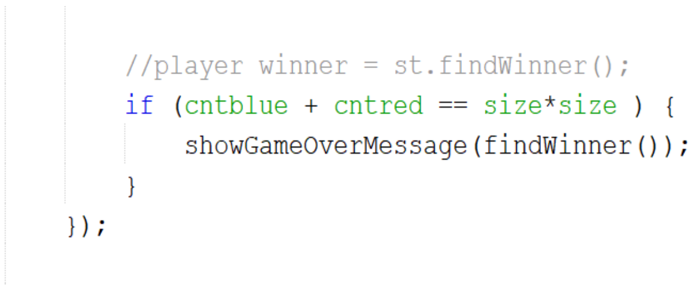
This is all about to hold the errors value for rows.

For example: We know that the j represents the column and size is the given size which get's from selected button so if the buttons which is selected hold two of the condition bellow then the condition will hold, when a button is selected it has i - which is row and j - which is column. A condition will j holds errors for columns and i - for rows. In first time if the button selected is buton[0][0] is the first button from the row and from the column. That buttons goes to each condition bellow and check which of them holds, in this case the first conitdion, hold because 0 < 2, at this button intexed we add the new value, we need to check if this new value is the same holds on for the rows as well, we increase the neighbours



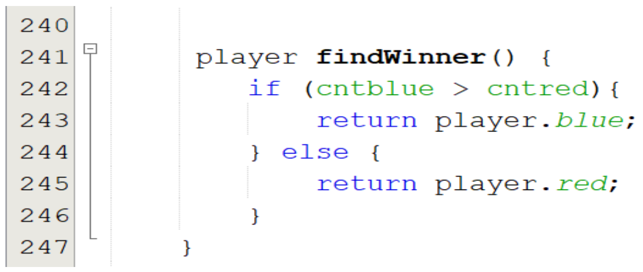
```

            if (i < size -1){
                buttons[i + 1][j].setText(Integer.toString(st.getValue(i+1, j)));
                if (newvalue3 >= 4 && oldvalue3 <newvalue3){
                    if (st.getActualPlayer() == player.red){
                        buttons[i+1][j].setBackground(Color.blue);
                        cntblue++;
                    } else {
                        buttons[i+1][j].setBackground(Color.red);
                        cntred++;
                    }
                }
            }

            if (i>0){
                buttons[i - 1][j].setText(Integer.toString(st.getValue(i-1, j)));
                if (newvalue4 >=4 && oldvalue4 < newvalue4 ){
                    if (st.getActualPlayer() == player.red){
                        buttons[i-1][j].setBackground(Color.blue);
                        cntblue++;
                    } else {
                        buttons[i-1][j].setBackground(Color.red);
                        cntred++;
                    }
                }
            }
```

We know that a game is over when all the buttons are field with colors.



```
        //player winner = st.findWinner();
        if (cntblue + cntred == size*size ) {
            showGameOverMessage(findWinner());
        }
    });
```

Find winner for the players

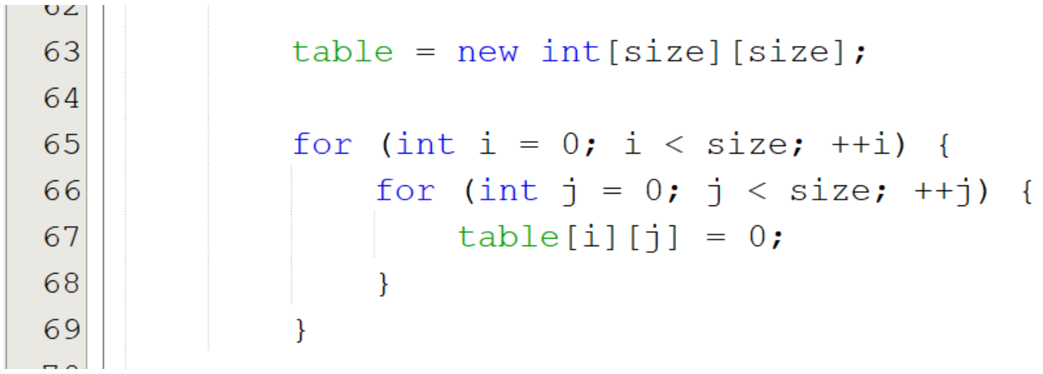


```

    player findWinner() {
        if (cntblue > cntred){
            return player.blue;
        } else {
            return player.red;
        }
    }
```

The class **solution** is the class which holds the game logic.

This class extends the base class, has an object for players enumerator, a table integer and two attributes to count the red and blue value. The constructor initialize those attributes.

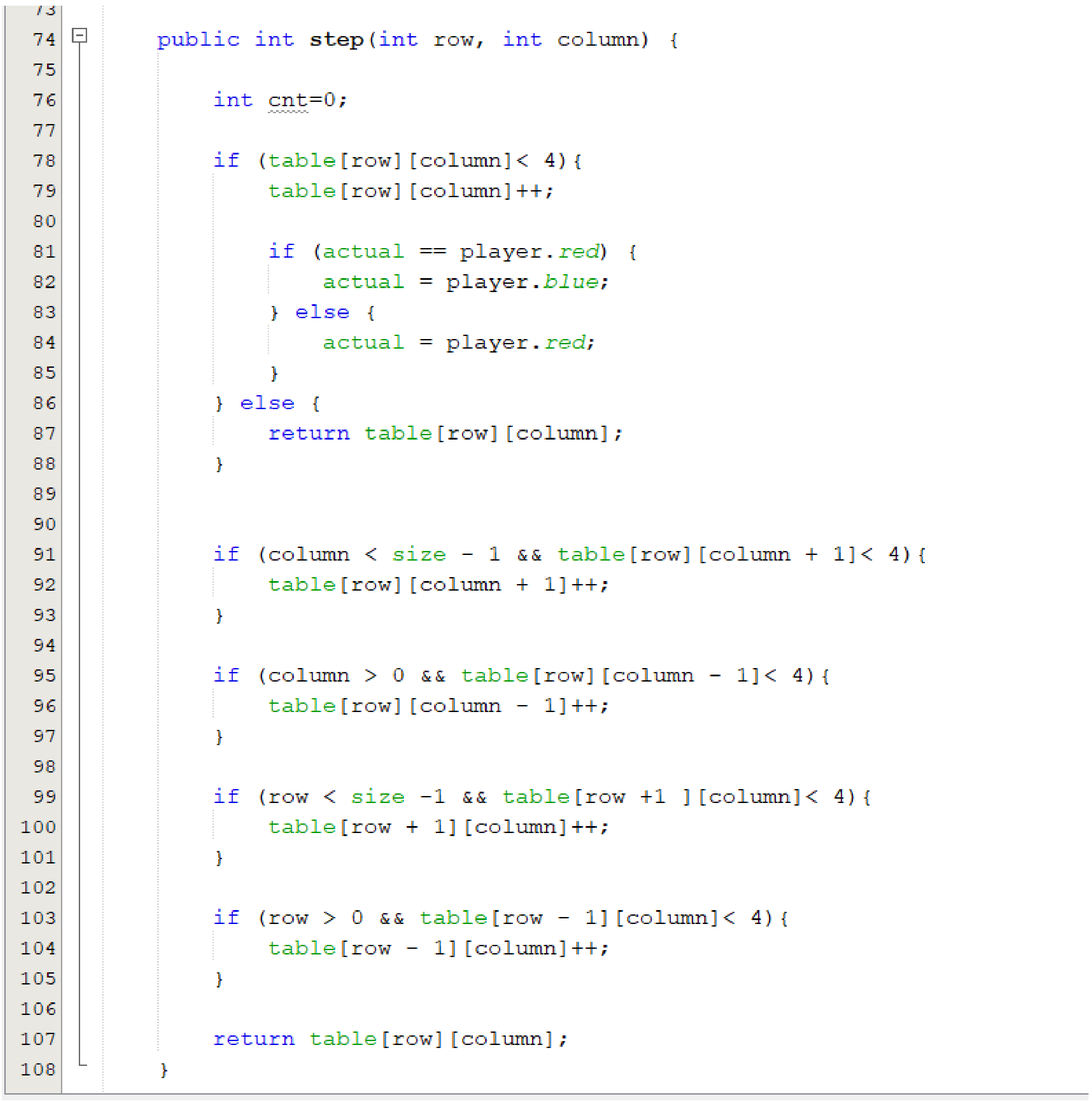
The table is two dimensional with size value. Firstly we initialize all the values to the table to 0.



```
        table = new int[size][size];

        for (int i = 0; i < size; ++i) {
            for (int j = 0; j < size; ++j) {
                table[i][j] = 0;
            }
        }
```

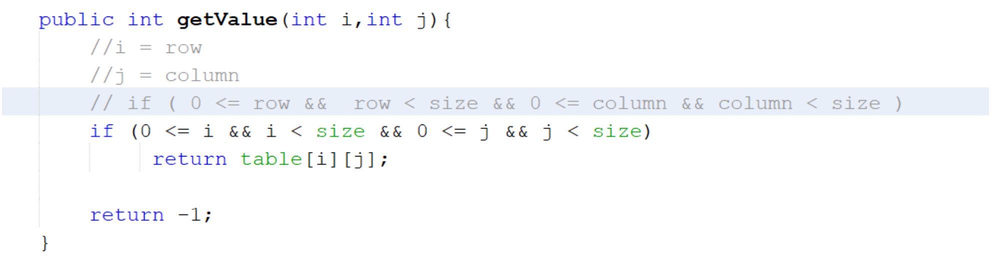
The step method calculated which is the player turn. We need to make condition to handle the errors for the table.

We know that the j represents the column and size is the given size which get's from selected button so if the buttons which is selected hold two of the condition bellow then the condition will hold, when a button is selected it has i - which is row and j - which is column. A condition will j holds errors for columns and i - for rows. In first time if the button selected is buton[0][0] is the first button from the row and from the column. That buttons goes to each condition bellow and check which of them holds, in this case the first conitdion, hold because 0 < 2, at this button intexed we add the new value, we need to check if this new value is the same holds on for the rows as well, we increase the neighbours



```
    public int step(int row, int column) {

        int cnt=0;

        if (table[row][column]< 4){
            table[row][column]++;

            if (actual == player.red) {
                actual = player.blue;
            } else {
                actual = player.red;
            }
        } else {
            return table[row][column];
        }


        if (column < size - 1 && table[row][column + 1]< 4){
            table[row][column + 1]++;
        }

        if (column > 0 && table[row][column - 1]< 4){
            table[row][column - 1]++;
        }

        if (row < size -1 && table[row +1 ][column]< 4){
            table[row + 1][column]++;
        }

        if (row > 0 && table[row - 1][column]< 4){
            table[row - 1][column]++;
        }

        return table[row][column];
    }
```

This is the current value for the selected button



```
public int getValue(int i,int j){
    //i = row
    //j = column
    // if ( 0 <= row &&  row < size && 0 <= column && column < size )
    if (0 <= i && i < size && 0 <= j && j < size)
        return table[i][j];

    return -1;
}
```