Name: Aichurok Kanatbekova
Neptun ID: RIMR52
Date: November 14, 2021
Task number: 6
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 neighbours 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.

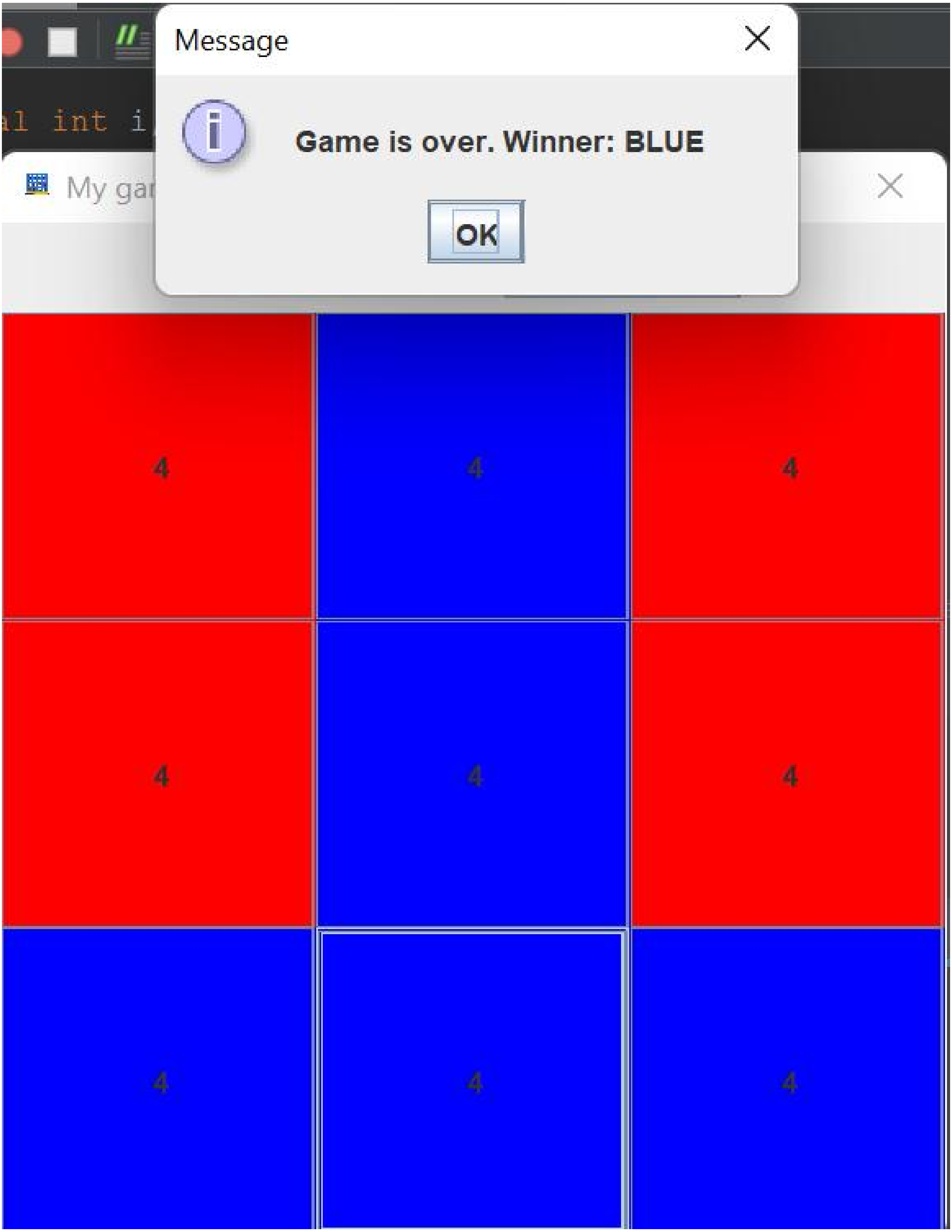
Decription of classes and methods:

EntryWindow - base window which has nothing but serves as base for other windows
MenuWindow - menu window where the level of game can be chosen
GameWindow - game window where the essence of game happens such as eventHandler registering, it has grid whose size is either 3x3,5x5, or 7x7
Player - enum which represents two players and also player Noone which will be also needed later on when the scores of
players equals
Cell - this class represents a cell in a grid. It has a counter and also a player attribute
Model - this class has table attribute which is size by size matrix and its constructor initializes all the matrix
cells' counter to 0 and player to NOONE.

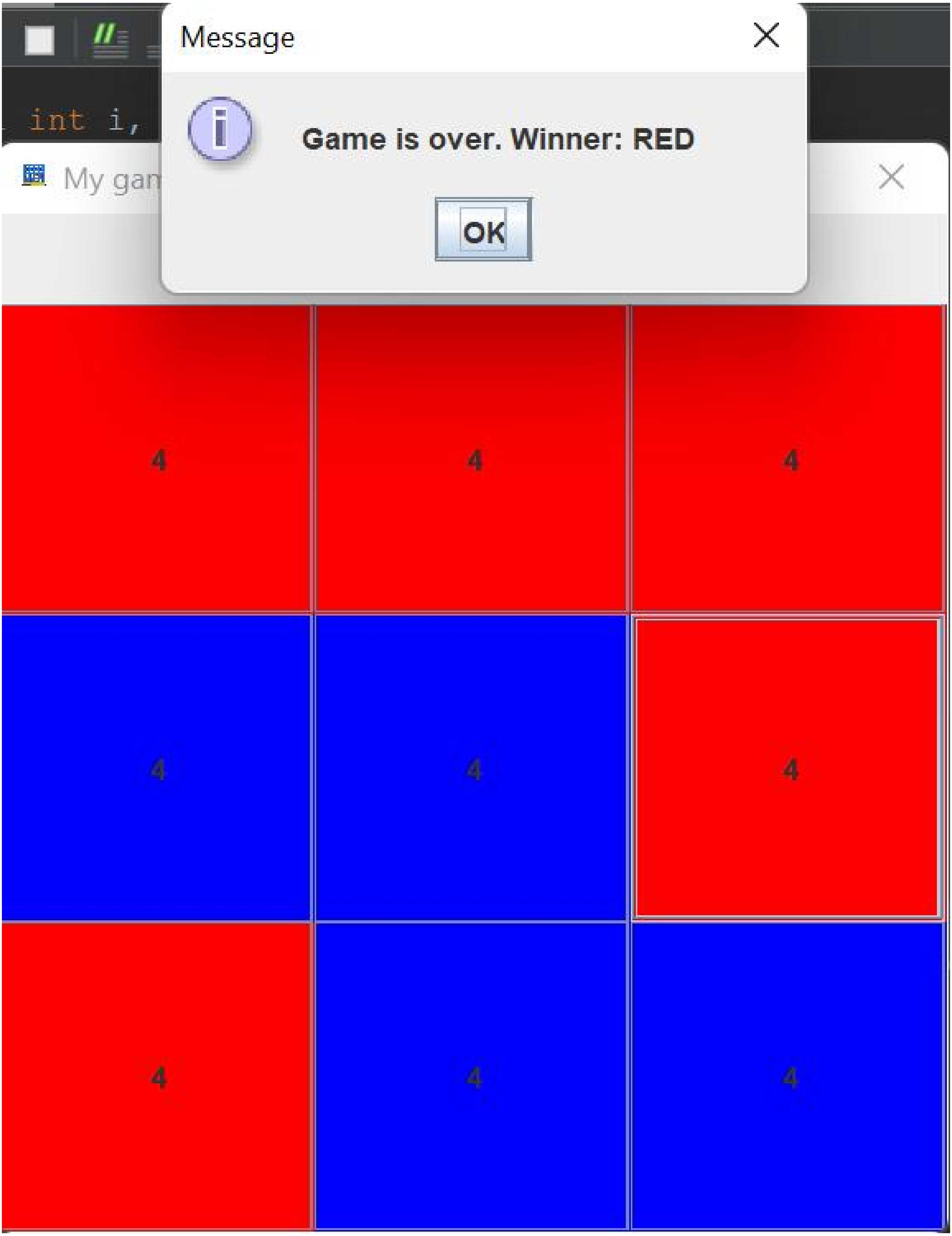
```
      1) step(i,j) - returns Cell, it checks whether table[i][j]'s counter < 4 and if yes, then it increases the counter of this cell
      otherwise, if cell's counter equals 4, then it's player attribute becomes the currentPlayer.
        this method also changes the player's turn
      2) findWinner() - this function checks if all the cells have player attribute other than NOONE and if yes,
      then it means the game is over. After, this method counts how many cells are Player.RED and Player.BLUE
      and returns the one with greater score.
      3) rangeChecker(i,j) - performs checking is counter is less or equal to 4
      4) getCell(i,j) - gets the cell by the given i and j indices
```

Test cases: (jpg files will be included for test cases)

```
      3x3 - when red starts and blue's score is greater
      3x3 - when red starts and red's score is greater
      5x5 - when red starts and blue wins
      7x7 - when red starts and blue wins
```



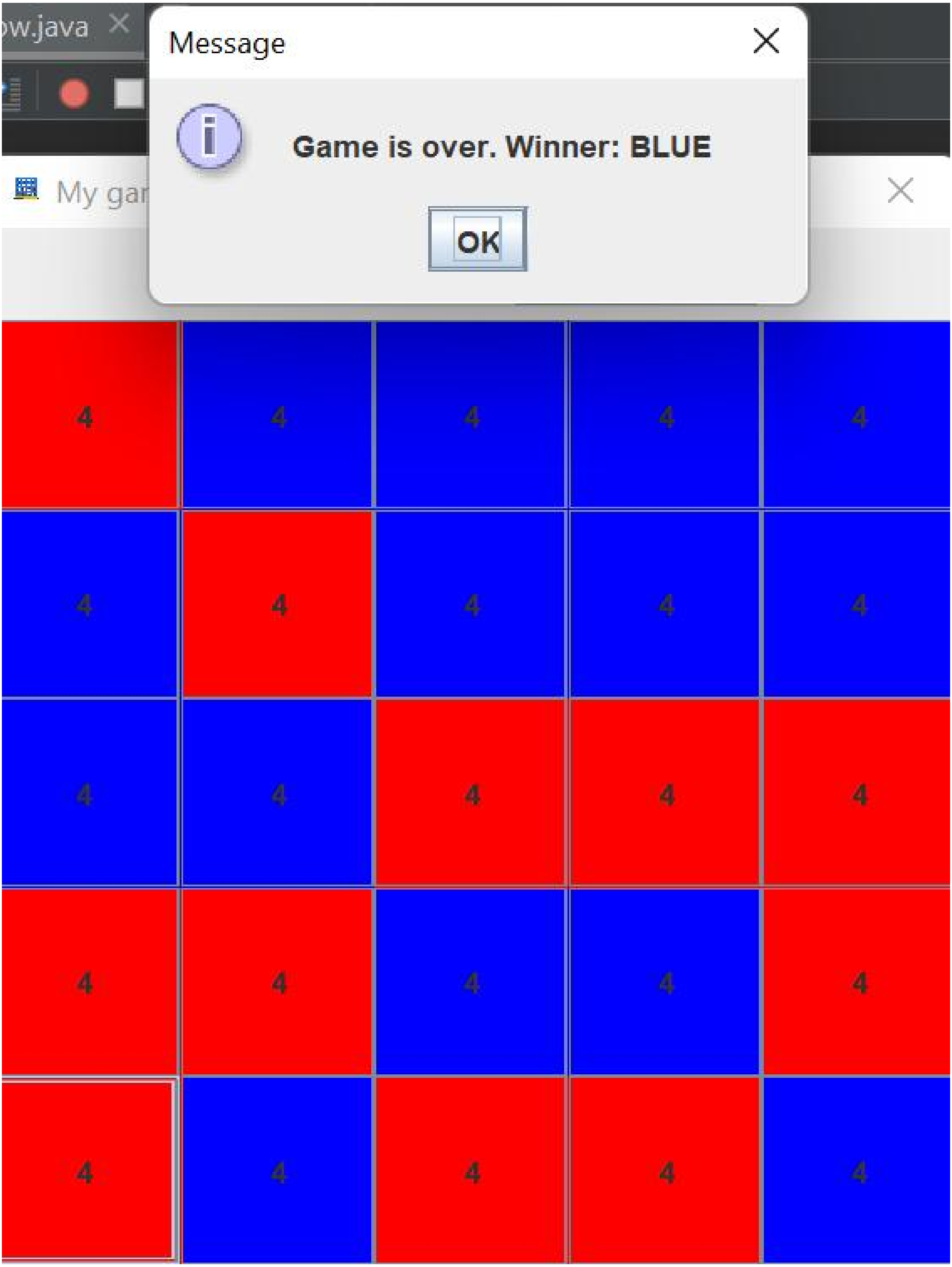
Message
Game is over. Winner: BLUE
OK
My ga
al int i
4
4
4
4
4
4
4
4
4



Message

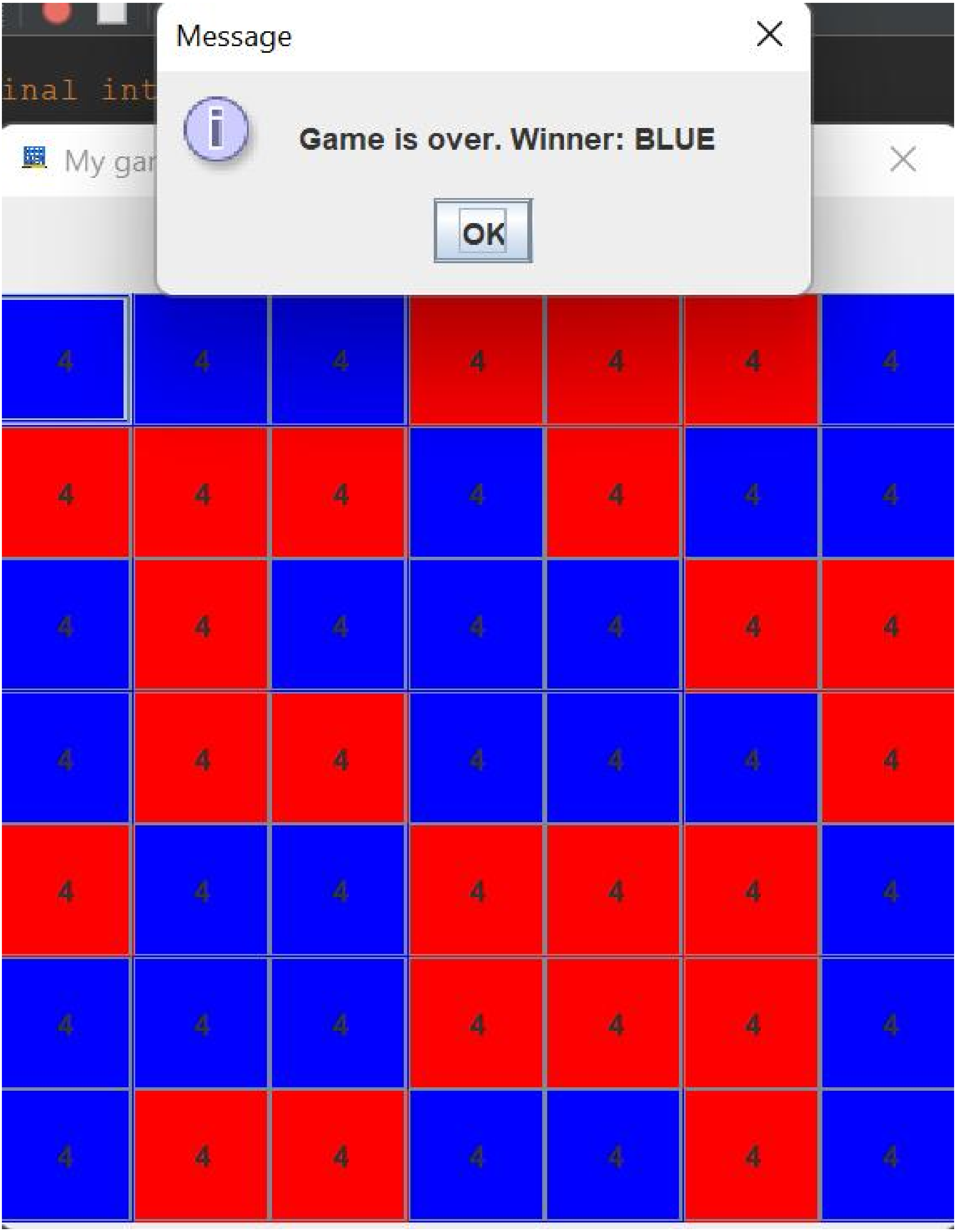
Game is over. Winner: RED

| | | |
|---|---|---|
| 🟥 4 | 🟥 4 | 🟥 4 |
| 🟦 4 | 🟦 4 | 🟥 4 |
| 🟥 4 | 🟦 4 | 🟦 4 |



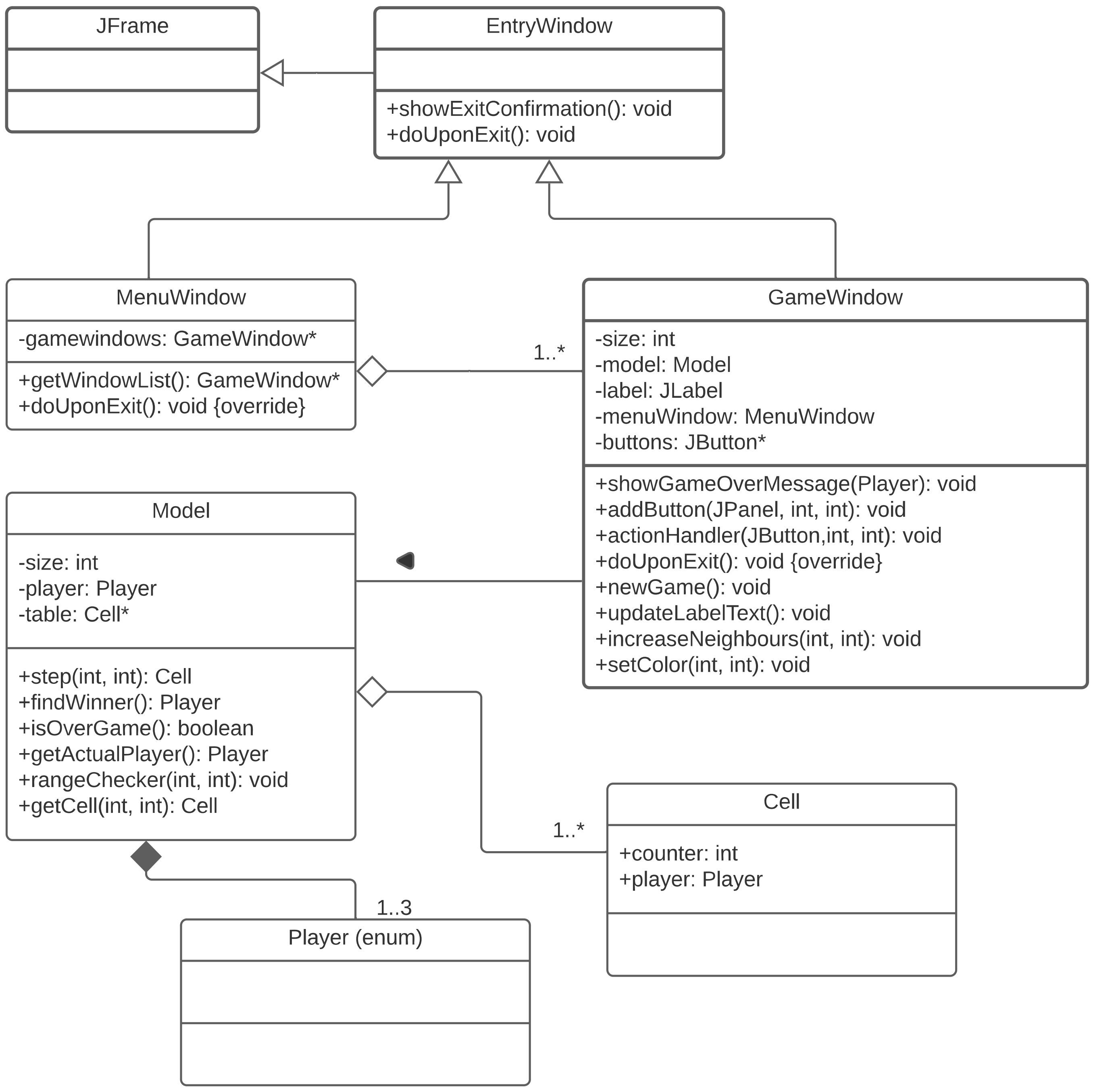
Game is over. Winner: BLUE

| | | | | |
|---|---|---|---|---|
| 🟥 4 | 🟦 4 | 🟦 4 | 🟦 4 | 🟦 4 |
| 🟦 4 | 🟥 4 | 🟦 4 | 🟦 4 | 🟦 4 |
| 🟦 4 | 🟦 4 | 🟥 4 | 🟥 4 | 🟥 4 |
| 🟥 4 | 🟥 4 | 🟦 4 | 🟦 4 | 🟥 4 |
| 🟥 4 | 🟦 4 | 🟥 4 | 🟥 4 | 🟦 4 |



Game is over. Winner: BLUE

| | | | | | | |
|---|---|---|---|---|---|---|
| 🟦 4 | 🟦 4 | 🟦 4 | 🟥 4 | 🟥 4 | 🟥 4 | 🟦 4 |
| 🟥 4 | 🟥 4 | 🟥 4 | 🟦 4 | 🟥 4 | 🟦 4 | 🟦 4 |
| 🟦 4 | 🟥 4 | 🟦 4 | 🟦 4 | 🟦 4 | 🟥 4 | 🟥 4 |
| 🟦 4 | 🟥 4 | 🟥 4 | 🟦 4 | 🟦 4 | 🟦 4 | 🟥 4 |
| 🟥 4 | 🟦 4 | 🟦 4 | 🟥 4 | 🟥 4 | 🟥 4 | 🟦 4 |
| 🟦 4 | 🟦 4 | 🟦 4 | 🟥 4 | 🟥 4 | 🟥 4 | 🟦 4 |
| 🟦 4 | 🟥 4 | 🟥 4 | 🟦 4 | 🟦 4 | 🟥 4 | 🟦 4 |


JFrame
EntryWindow
+showExitConfirmation(): void
+doUponExit(): void
MenuWindow
-gamewindows: GameWindow*
+getWindowList(): GameWindow*
+doUponExit(): void {override}
1..*
GameWindow
-size: int
-model: Model
-label: JLabel
-menuWindow: MenuWindow
-buttons: JButton*
+showGameOverMessage(Player): void
+addButton(JPanel, int, int): void
+actionHandler(JButton,int, int): void
+doUponExit(): void {override}
+newGame(): void
+updateLabelText(): void
+increaseNeighbours(int, int): void
+setColor(int, int): void
Model
-size: int
-player: Player
-table: Cell*
+step(int, int): Cell
+findWinner(): Player
+isOverGame(): boolean
+getActualPlayer(): Player
+rangeChecker(int, int): void
+getCell(int, int): Cell
1..*
Cell
+counter: int
+player: Player
1..3
Player (enum)