MARMARA UNIVERSITY

THE FACULTY OF ENGINEERING

CSE1142
COMPUTER PROGRAMMING II
SPRING 2021

TERM PROJECT

A JavaFX Game Application : Boooom!

Submitted to :

Assistant Professor SANEM ARSLAN YILMAZ

Due Date : 28.05.2021

CSE - EVREN KOYMAT - 150120518

CSE - SİNAN GÖÇMEN - 150120519

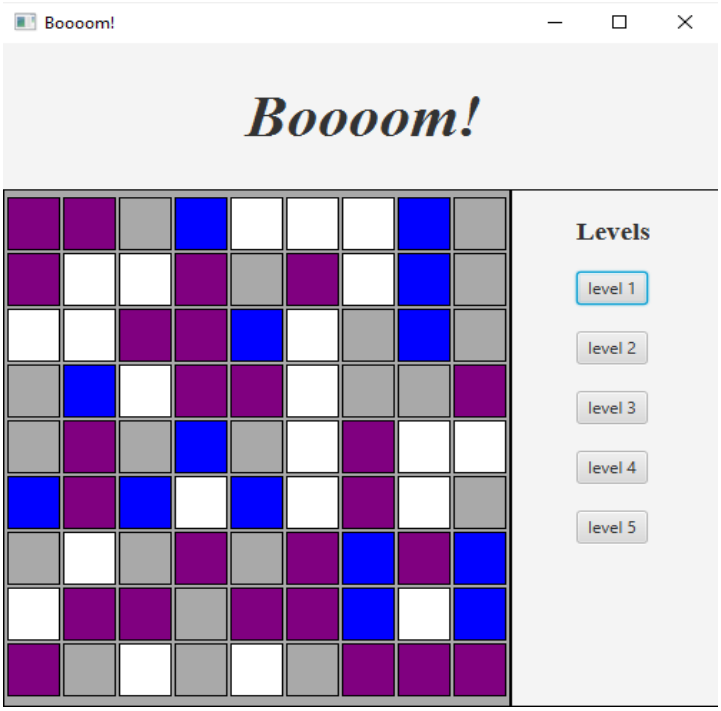
1. Problem Definition

This application is a kind of game named “Boooom!” . We are asked to do this by using JavaFX framework, which is a set of graphics that programmers use to create complex applications.

In this game, there are 5 levels and each of them includes several types of rectangles. The aim is to destroy rectangles by considering their durability and clicking as few as possible. When all the rectangles are destroyed, next level becomes available. The usage of the game will be explained in more detail in the implementation part.

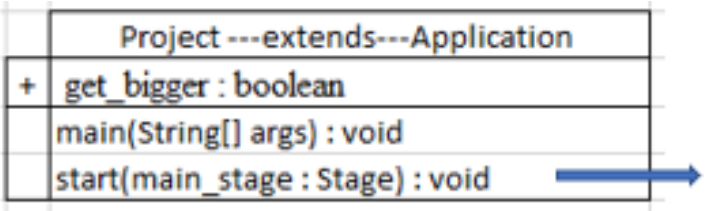
2. Implementation Details

In this program, there are 6 main classes. One of them is for the main menu, the others are for the levels. When we open project.java class in eclipse and start the program, The main menu is shown like this. At the top of the menu, there is the title which has a constant motion. At left, there is a simple example of a game board. It has a constant motion as well. At right, there are separate buttons to reach the levels one by one. When the previous level is done, the button for the next level becomes active. If we try to reach next level without completing the previous one, A warning text occurs just under of the buttons. When we click a button, a different window arises. For each level, there is a specific stage and window.



2.1 Main Menu

In the main menu, there is a border pane which has three other panes named menu_top, menu_center and menu_right. Each of them is created with different types of pane according to their purposes.




```
BorderPane menu_pane = new BorderPane();
StackPane menu_top = new StackPane();//f
GridPane menu_center = new GridPane();//
VBox menu_right = new VBox(20);//for the
menu_pane.setTop(menu_top);
menu_pane.setCenter(menu_center);
menu_pane.setRight(menu_right);
```

In menu_top, there is just a label to show the title of the game. But it is more than that. There is an animation in this part. The get_bigger property depends on the size of the title “Boooom!”. When it becomes 30, get_bigger becomes true and activate an if statement which provides the title is getting bigger until it is 50. Then get_bigger becomes false and title is getting smaller until it is 30 and so on.

The creation of the title

```
Label game_title = new Label("Boooom!");
game_title.setFont(Font.font("Times New Roman",
        FontWeight.EXTRA_BOLD, FontPosture.ITALIC, 50));
game_title.setAlignment(Pos.CENTER);
```

The creation of the animation

```
EventHandler<ActionEvent> event_handler = e -> {

    //When it is small enough, it starts to get bigger and vice versa
    if(game_title.getFont().getSize()<=40)
    get_bigger = true;
    else if(game_title.getFont().getSize()>=50)
    get_bigger = false;

    if(get_bigger) {
        game_title.setFont(Font.font("Times New Roman",
        FontWeight.EXTRA_BOLD, FontPosture.ITALIC, game_title.getFont().getSize()+1));
    }
    else {
        game_title.setFont(Font.font("Times New Roman",
        FontWeight.EXTRA_BOLD, FontPosture.ITALIC, game_title.getFont().getSize()-1));
    }
};

Timeline animation = new Timeline(new KeyFrame(Duration.millis(20),event_handler));
animation.setCycleCount(Timeline.INDEFINITE);
animation.play();
```

In the menu_right , there is a title “Levels” and 5 buttons for each level. There is an also warning label which occurs when we try to reach levels without doing the previous one.

Each button has an action. When level_x button is pressed, level_x_stage.show() method becomes active and the window for level_x arises. Also we check if the previous level is done or not in setOnAction method as in the fallowing figure.

```
level5_button.setOnAction(e -> {
    if(BorderPane_for_L4.is_it_done)
    level5_stage.show();
    else {
        if(!menu_right.getChildren().contains(warning_label))
        menu_right.getChildren().add(warning_label);
    }
});
```

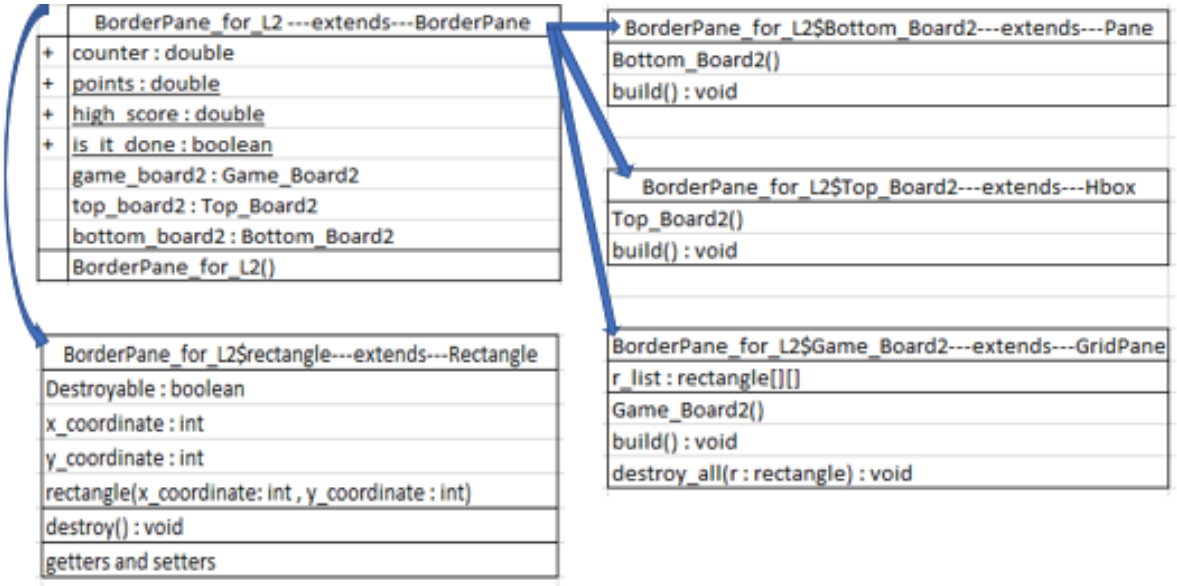
In the menu_left, There is a for loop. This loop creates 10x10 rectangles and put them to a specific position. Then it gives them random numbers from 0 to 4 to fill them in different random colors. The colors change every second thanks to a TimeLine animation as in the figure.

```
for(int i = 0; i<=8;i++) {
    for(int j = 0;j<=8;j++) {
        Rectangle r = new Rectangle();
        r.setWidth(36);
        r.setHeight(39);
        r.setLayoutX(j*36);
        r.setLayoutY(i*49);
        r.setStroke(Color.BLACK);

        //random coloring
        EventHandler<ActionEvent> event_handler_2 = e -> {

        int random_value = (int)(Math.random()*4);

        if(random_value == 0)
            r.setFill(Color.WHITE);
        else if(random_value == 1)
            r.setFill(Color.DARKGRAY);
        else if(random_value == 2)
            r.setFill(Color.BLUE);
        else
            r.setFill(Color.PURPLE);
        };

        //This is an animation which provides rectangles to change their color every second
        Timeline animation_2 = new Timeline(new KeyFrame(Duration.millis(1000),event_handler_2));
        animation_2.setCycleCount(Timeline.INDEFINITE);
        animation_2.play();

        menu_center.add(r,i,j);
```

2.2 Level Stages

As it is stated, there are separate stages for each level. Each level is a copy of each other. There are just small changes according to the unique properties of the levels like filled rectangles. In this part, the details about level-2 will be shared.

BorderPane_for_L2 Class Hierarchy



2.2.1 Border Pane for level-2

As we stated, all level classes are actually a border pane. Level-2 border pane contains three other panes in itself as in the fallowing figure, which are the top pane, the bottom pane and the game board. We created and add them to the border pane.

```
Game_Board2 game_board2 = new Game_Board2();
Top_Board2 top_board2 = new Top_Board2();
Bottom Board2 bottom board2 = new Bottom Board2();
   public BorderPane_for_L2() {
       setCenter(game_board2);
       setTop(top_board2);
       setBottom(bottom_board2);
   }
```

There are also four other properties in the border pane. Counter simply counts the number of the rectangles destroyed. Points take the point of the user by considering counter. High_score take the high score depends on the point of the user. Is_it_done property determines if the all rectangles are destroyed or not by considering their colors. More explanation will be given.

```
public double counter;
public static double points;
public static double high_score;
public static boolean is_it_done;
```

2.2.2 rectangle Class

We decided to create a rectangle class which extends the Rectangle class. The aim was to add new properties to make coding easier and also using the Rectangle class properties at the same time. It has destroyable, x and y properties. When a rectangle is brown or white, destroyable property is set as false and vice versa for blue and purple.

In the constructer of the rectangle, a default rectangle is created with a specific x and y property and also a default size and color. It is better to use new x and y properties than default x and y properties because they give us the pixel count. But new x and y properties give us a coordinate.

```
public rectangle (int x, int y)  {
    //Just some settings for default rectangle.
    this.setHeight(50);
    this.setWidth(50);
    this.x_coordinate = x;
    this.y_coordinate = y;
    this.setFill(Color.DARKGRAY);
    this.setStroke(Color.BLACK);
}
```

Also, the rectangle class has a method named destroy(). In destroy function, The color and the destroyable property changes as in the fallowing figure. The destroyable property is determined in the game_board2 according to the level-2 filling rules. If a rectangle is purple, destroy function makes it blue. If it is blue, then it will be white an also “not destroyable.”

```
public void destroy() {

    //If it is purple, it turns the color to blue

        if(getFill()==Color.PURPLE) {
            setFill(Color.BLUE);
        }

    //If it is blue, it turns the color to white and set destroyable

        else if(getFill()==Color.BLUE) {
            setFill(Color.WHITE);
            setDestroyable(false);
        }

    }
```

2.2.3 Top Board

In top board, there is just the name of the level, the point and also the high score . Additionaly, there is a small bug but it can be solved easily. More explanation will be given in the test cases. The build function of the class provides the class to have three labels as in the fallowing figure.

```
public void build() {

    setPadding(new Insets(10,10,10,10));
    setStyle("-fx-border-color: black");
    Label name = new Label("Level # 2                                    ");
    Label point = new Label("" + points);
    Label score = new Label("                              High Score: " + high_score);
    getChildren().add(name);
    getChildren().add(point);
    getChildren().add(score);
}
```

When we get a two-digit point, the point label does not show the point properly. But the point is stored correctly and there is a small solution. If we close the window and open it again in the main menu, we can see the point even if it is 2-digit and also can continue from where we left.

2.2.4 Bottom Board

In this board, there are several labels. As in project guide, when we click on a rectangle, the coordinates of the destroyed rectangles are shown. At the end of the level, a label arises which says “the next level is available!”. The implementation of these are in the game_board2 class, so more detail will be given in that part.

2.2.5 Game Board

In our game board ,we create a rectangle list to store the rectangles at first.

```
rectangle[][] r_list = new rectangle[10][10];
```

	Game_Board1 extends GridPane
+	build(); void
+	destroy_all(rectangle r); void

In the build() method, we create rectangles according to the i (row) and j (column) values in the first loop. Then , we activate the setOnMouseClicked event to destroy boxes .

```
for(int i = 0;i<10;i++) {
    for(int j = 0;j<10;j++) {
        rectangle r = new rectangle(i,j);
        r_list[i][j] = r;
        add(r, i, j);
        //When we click the mouse, destroy all function is activated.
        r.setOnMouseClicked(e -> {destroy_all(r);
        //points variable changes according to counter variable.
```

We increment the points value in this loop according to the counter value.And also created high_score value.

```
if (counter == 1)
    points -= 3;
else if (counter == 2)
    points -=1;
else if (counter == 3)
    points +=1;
else if (counter ==4)
    points +=2;
else if (counter == 5)
    points +=4;

if (points >= high_score)
    high_score = points;
```

In every mouseClicked event , we need to change the points and counter variables . So we used .clear() method and add them again.

```
top_board1.getChildren().clear();
top_board1.getChildren().add(name);
top_board1.getChildren().add(point);
top_board1.getChildren().add(score);
```

In the destroy_all(rectangle r) method, First, we check the IsDestroyable() value of the rectangle which is clicked. Then we try to find any affected rectangle by using if statements.

Also we need to write the clicked box and affected box on the bottom page , so we create hit# and box variables.

```
public void destroy_all(rectangle r) {
    //We get the coordinates of the rectangle to simplify.
    int i = r.getX_coordinate();
    int j = r.getY_coordinate();

    if(i==0 && j==0 && r.IsDestroyable() ) {
        if(r_list[i][j].IsDestroyable()) {
            r_list[i][j].destroy();
            //when we destroy a rectangle counter variable increase 1.
            counter +=1;
            //we need to ith and jth values for destroyed rectangle to prin
            Label box = new Label("Box:" +i + "-" +j );
            //.clear() method helps us to clear the previous bottom page. S
            bottom_board1.getChildren().clear();
            bottom_board1.getChildren().add(box);
        }
        if(r_list[i+1][j].IsDestroyable()) {
            r_list[i+1][j].destroy();
            counter +=1;
            //hit# label keeps the value of affected rectangle.
            Label hit1 = new Label("        Hit:" + (i+1) +"-" + j);
            bottom_board1.getChildren().add(hit1);
        }
        if(r_list[i][j+1].IsDestroyable()) {
            r_list[i][j+1].destroy();
            counter +=1;
            Label hit2 = new Label("            Hit:" + i +"-" + (j+1));
            bottom_board1.getChildren().add(hit2);
            }
```

We need to write gained or lost points according to each click event. So we add this code at the end of each if blocks.

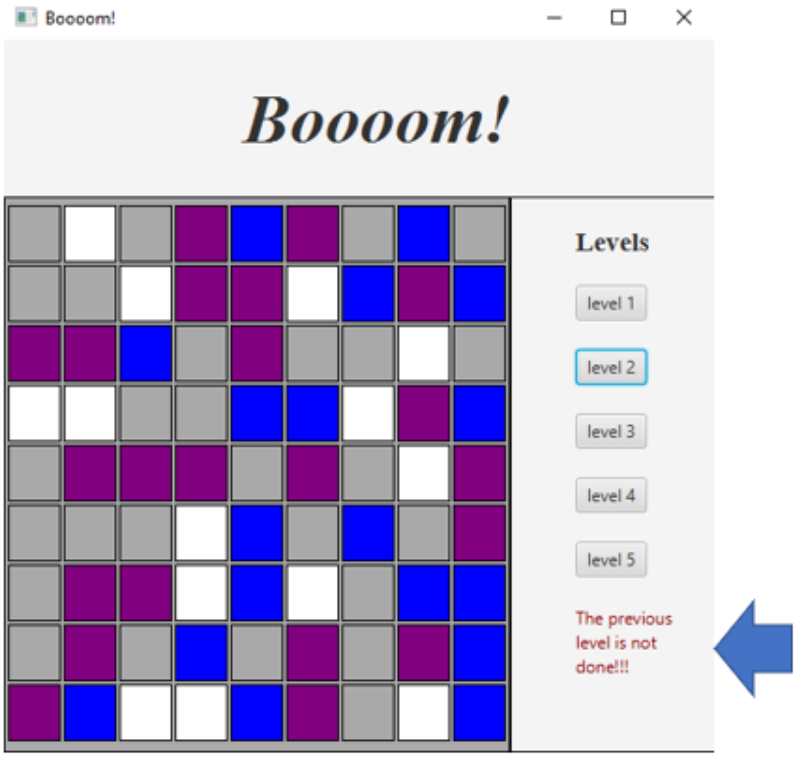
```
if (counter == 1)
    bottom_board1.getChildren().add(new Label("                                    (-3 points)"));
else if (counter == 2)
    bottom_board1.getChildren().add(new Label("                                    (-1 points)"));
else if (counter == 3)
    bottom_board1.getChildren().add(new Label("                                    (+1 points)"));
else if (counter ==4)
    bottom_board1.getChildren().add(new Label("                                    (+2 points)"));
else if (counter == 5)
    bottom_board1.getChildren().add(new Label("                                    (+4 points)"));
```

At the end of the GameBoard part, we need to write next level message if the current level is completed.By this code , we ensure that the level is completed or not:

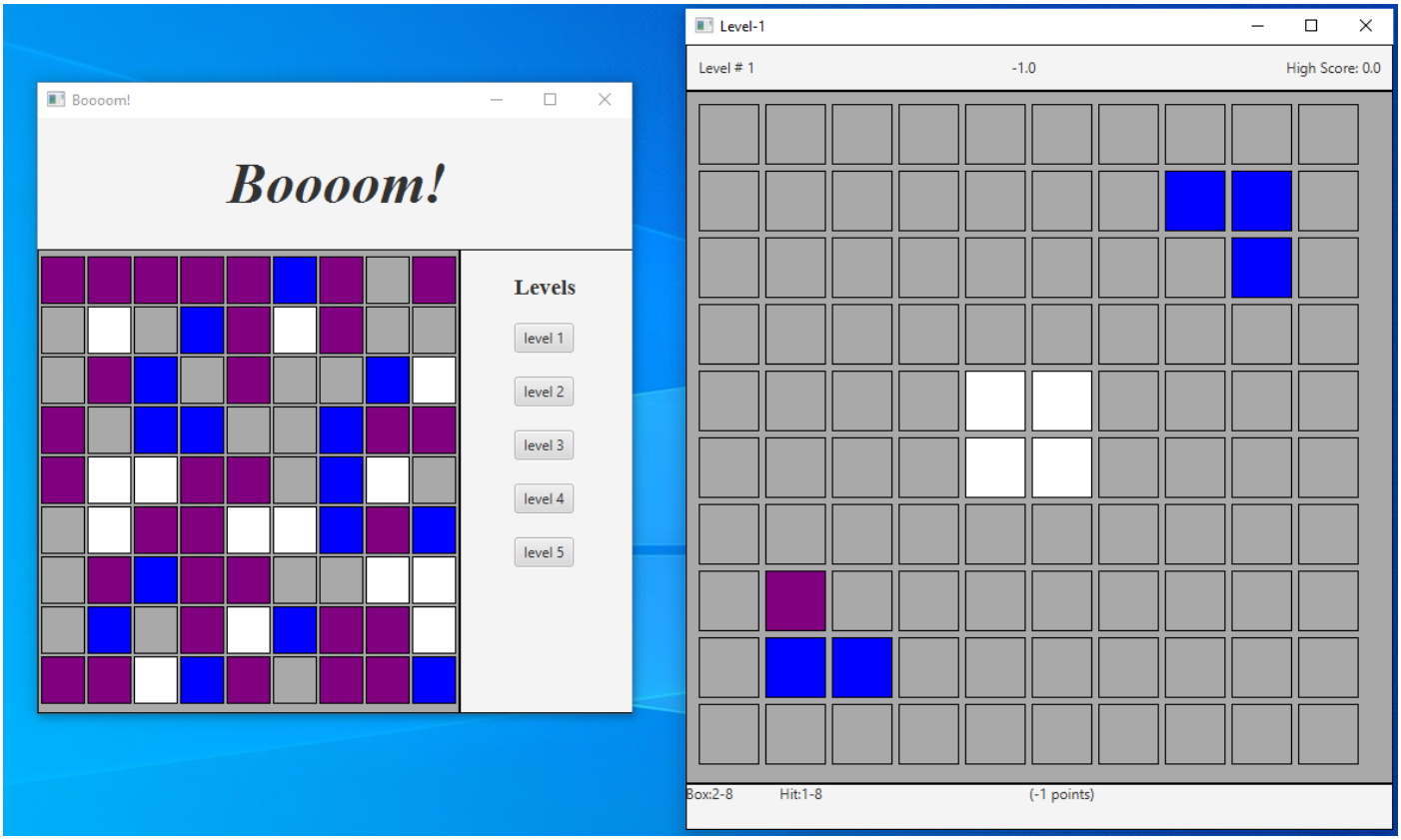
```
int r_count = 0;
for ( int m = 0 ; m < 10 ; m++) {
    for (int n = 0 ; n<10 ; n++) {
        if ((r_list[m][n].getFill() == Color.WHITE || r_list[m][n].getFill() == Color.DARKGRAY)) {
            r_count +=1;
    }
    if(r_count ==100)
    is_it_done=true;

    }
}
//When our level is completed , we print the next level is available now! message to the bottom page.
if(is_it_done == true && r_count == 100) {
    bottom_board1.getChildren().clear();
    Label nextLevel = new Label("next level is available now!");
    nextLevel.setAlignment(Pos.CENTER);
    bottom_board1.setPadding(new Insets(30,30,30,30));
    bottom_board1.getChildren().add(nextLevel);
    nextLevel.setLayoutX(220);
    nextLevel.setLayoutY(13);
```

3. Test Cases

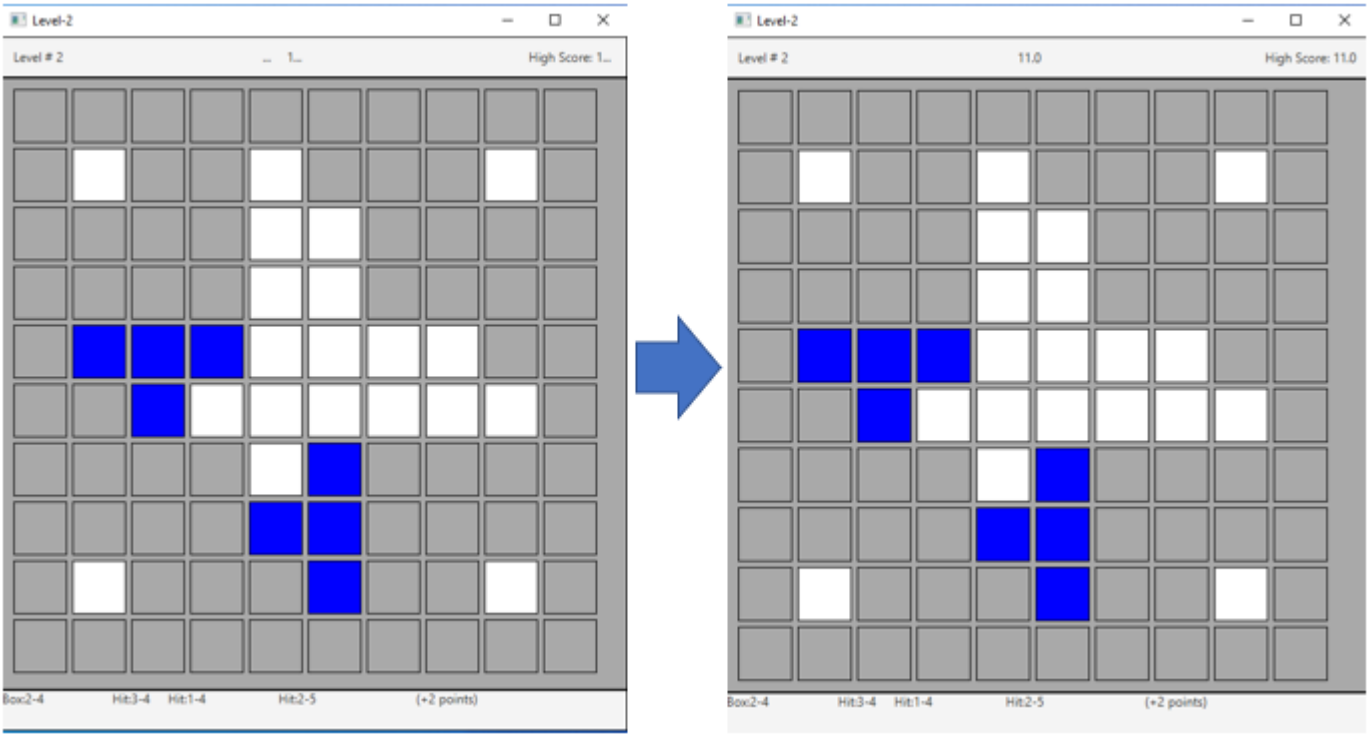
When we run the game, the main menu arises. If we want to open level-2 without complating level-1, it gives a warning as in the fallowing figure.



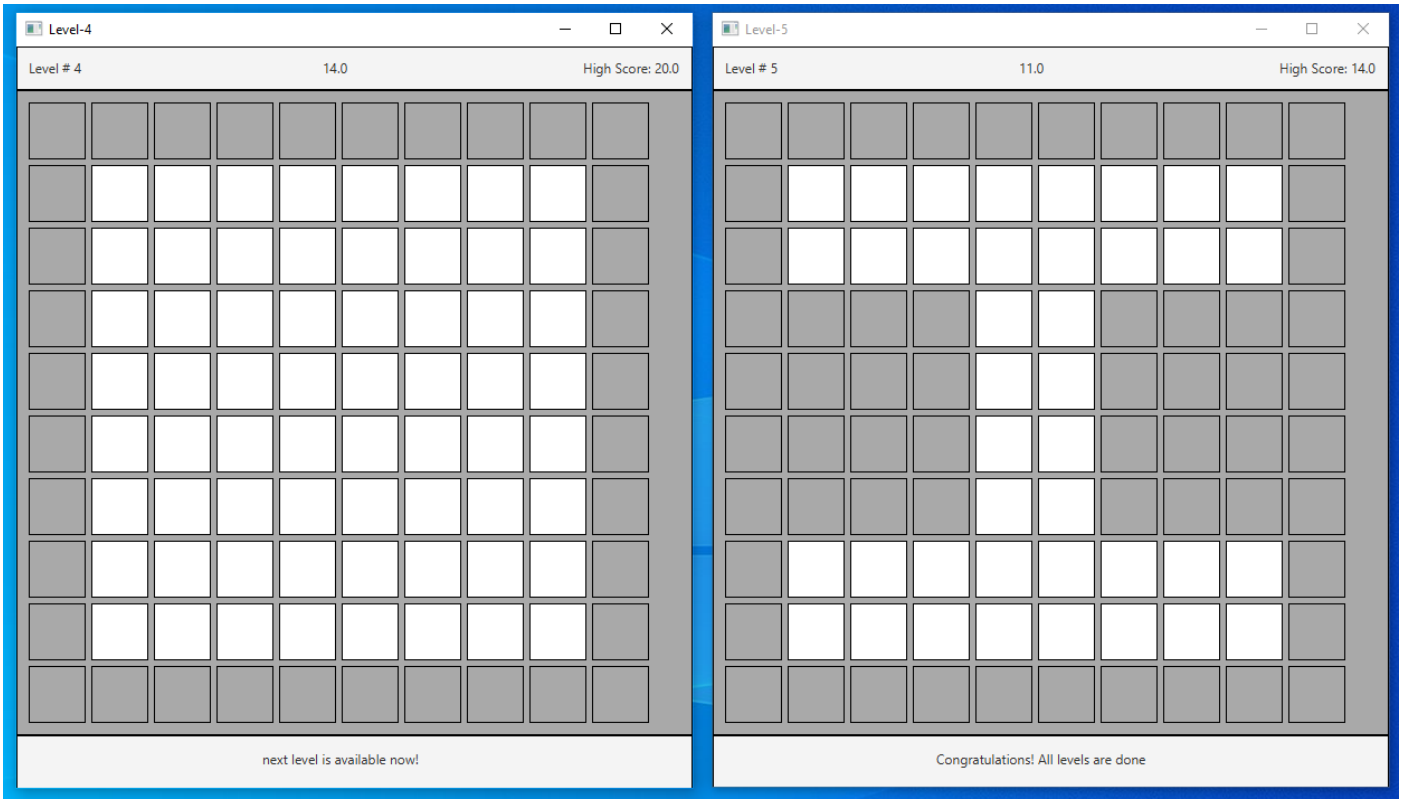
When we click the button level-1, the level-1 stage arises. At the end of the level, the next level becomes available.



When the point is two digit and not shown properly, we can close the window and open again.



When we complete the level, a label arises in the bottom pane which says “The next level is available!” However, when we complete the level-5, it says congratulations. It is shown in the fallowing figure.



4. Final Comments

When we started the project, we spend more than an hour to implement the main menu. Then we saw that the main menu was the easiest part. The hardest parts were to create the rectangles in the game board and also setting the point. We try to create rectangles one by one although we know it is bad coding and lots of work because we could not figure that out and try to do something. Then we create a qualified rectangle class and make the game board by using a loop.

In this project, we also saw that it is all about the process. When we did some little implementations and shared it with our mate almost every day at a specific hour, we got result and also get experienced. Additionaly, we saw that java is huge and we just know a little part of it.