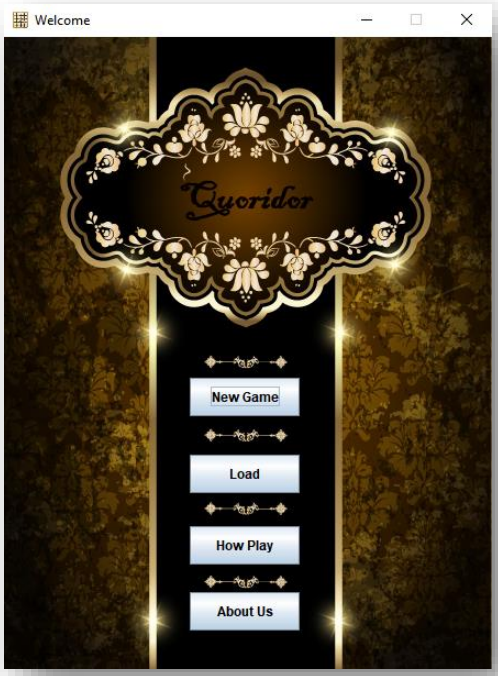
بسم الله الرحمن الرحیم



توضیحات نحوه ی کار کلاس ها و متد ها:

در متد main موجود در کلاس Main کانستراکتور کلاس Welcome را فرا می خوانیم.
با ورود به کلاس Welcom به وسیله ی swing نمای گرافیکی روبرو را نمایش می دهیم.
چند دکمه برای انتخاب نمایان می شود.
۱- NewGame: با کلیک بر روی این گزینه متد TwoPlayerButtonMouseClicked فرا خوانی میشود. در این متد از کلاس TowPlayer که وظیفه ی گرفتن اسامی بازیکنان را دارد شیئی ساخته می شود.
۲- Load: با کلیک بر روی این JButton متد loadButtonMouseClicked که وظیفه ی لود بازی را بر عهده دارد فراخوانی می شود.

```
20
21      private void loadButtonMouseClicked() {
22          Board board = new Board();
23          board.MakeAGraph();
24          board.StartLoading();
25          board.setVisible(true);
26          this.setVisible(false);
27      }
28
```

در این متد ابتدا شیئی از Board ساخته می شود. سپس متد MakeGraph در کلاس Board برای ساخت گراف از زمین بازی فراخوانی میشود.

۳- How Play : کانستراکتور کلاس PDF در پکیج Multimedia برای اجرای pdf مربوط به نحوه ی انجام بازی فراخوانی می شود.
۴- About Us : کانستراکتور کلاس Video در پکیج Multimedia برای اجرای کلیپ معرفی صدا زده می شود.

NewGame: با ورود به کلاس TowPlayer اسامی بازیکنان را می گیرد و سپس وارد کلاس Board می شود.

```
1   package Welcome;
2
3   import Board.Board;
4
5   import javax.swing.*;
6
7   public class TowPlayer {
8
9       TowPlayer(Welcome welcome) {
10          String NamePlayerOne = JOptionPane.showInputDialog( parentComponent: null, message: "Enter The Name Of Player 1", title: "Name Of Player 1", JOptionPane.INFORMATION_MESSAGE);
11          if (NamePlayerOne == null || (NamePlayerOne.equals(""))) { //If is null == Click cancel or close button
12              return;
13          }
14          String NamePlayerTwe = JOptionPane.showInputDialog( parentComponent: null, message: "Enter The Name Of Player 2", title: "Name Of Player 2", JOptionPane.INFORMATION_MESSAGE);
15          if (NamePlayerTwe == null || (NamePlayerTwe.equals(""))) { //If is null == Click cancel or close button
16              return;
17          }
18          showBoard(NamePlayerOne, NamePlayerTwe);
19          welcome.setVisible(false);
20      }
21
22      private void showBoard(String NamePlayerOne, String NamePlayerTwe) {
23          Board board = new Board();
24          board.setVisible(true);
25          board.setPlayerText(NamePlayerOne, NamePlayerTwe);
26          board.MakeAGraph();
27      }
28  }
```

بعد از ساخت شیء از کلاس بورد متد MakeAGraph فراخوانی میشود. این متد کانستراکتور کلاس MakeGraph را صدا می زند تا گرافی از بورد بسازد.

```java
public class Board extends JFrame {
    private final DefaultUndirectedGraph<JButton, DefaultEdge> graph = new DefaultUndirectedGraph<>(DefaultEdge.class);
    private final ArrayList<JButton> jButtonArrayList = new ArrayList<>();
    private AddElementToBoard addElementToBoard;
    private boolean isRoundPlayer1 = true;
    private int position, WallOfPlayer1 = 10, WallOfPlayer2 = 10, countVertical = 0, countHorizontal = 0;
    private FieldButton fieldButton;
```

- برای ساخت گراف از کتابخانه ای به نام JGraphT استفاده کردیم و گرافی غیر جهت دار ساختیم.
- برای ذخیره ی تمام دکمه های صفحه بازی اعم از دیوار ها، محل قرارگیری مهره و دکمه سیو از arrayList<JButton> استفاده کردیم.
- برای اینکه مشخص کنیم نوبت بازی کدام مهره هست از یک بولین به نام isRoundPlayer1 استفاده کردیم که اگر true بود یعنی نوبت مهره ی اول است در غیر این صورت نوبت مهره ی دوم است.
- هر بازیکن ده دیوار دارد (WallOfPlayer1/2)
- position در واقع مختصات هر مهره را ذخیره می کند تا با برای حرکت مهره رنگ محلی که بوده است را بتوانیم به مرحله ی قبل برگردانیم.

```java
addElementToBoard = new AddElementToBoard(panel, jButtonArrayList, WallOfPlayer1, WallOfPlayer2);
for (int i = 0; i < jButtonArrayList.size() - 1; i++) {
    int finalI = i;
    jButtonArrayList.get(i).addMouseListener(new MouseAdapter() {
        @Override
        public void mouseClicked(MouseEvent e) {
            ButtonMouseClick(jButtonArrayList.get(finalI), finalI);
        }
    });
```

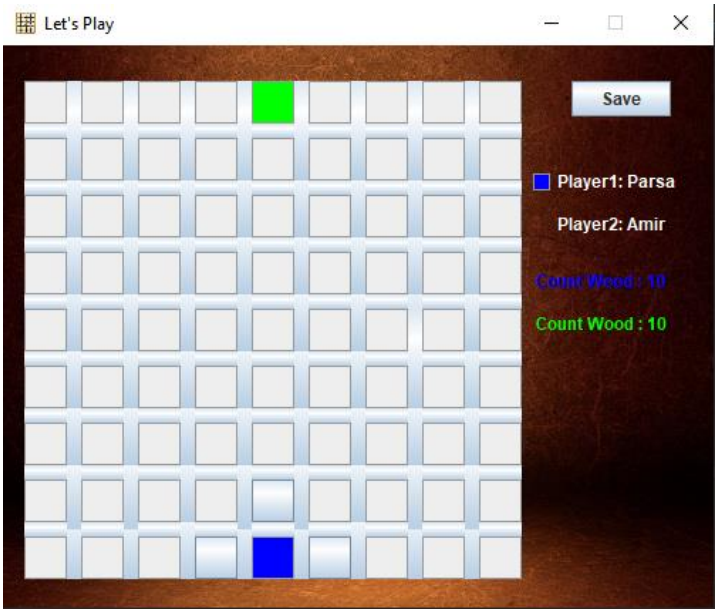
با استفاده از خط اول کد بالا شیئ از کلاس AddElementToBoard می سازیم. این کلاس وظیفه ی ساخت و قرار دادن دکمه های بورد بازی داخل JButtonArrayList را دارد. ابتدا ۶۴ دیوار عمودی به ترتیب از راست به چپ سپس ۶۴ دیوار افقی به ترتیب از راست به چپ وارد اری لیست می شوند. بعد از دیوار ها محل قرارگیری مهره ها به نام FieldButtons داخل اری لیست Add می شود و در آخر دکمه سیو ذخیره می شود.
در for برای هر کدام از دکمه ها یک اکشن لیسنر می سازد تا زمانی که بر روی دکمه کلیک شد عملیاتی انجام شود.
بعد از کلیک بر روی دکمه ای در برد غیر از سیو وارد متد ButtonMouseClick می شویم. دو شرط وجود دارد:

```java
if (sender.isEnabled()) {
    if (i < 128) { //Wall
        if (new WallButton().shortPath(jButtonArrayList, graph, i, countVertical, countHorizontal)) {
            if (isRoundPlayer1 && WallOfPlayer1 > 0) {
                new Audio( FilePath: "src/Files/Audio/WallSound.wav");
                WallOfPlayer1--;
                CountWood();
                check(sender, i);
                RoundPlayer1();
            } else if ((!isRoundPlayer1) && WallOfPlayer2 > 0) {
                new Audio( FilePath: "src/Files/Audio/WallSound.wav");
                WallOfPlayer2--;
                CountWood();
                check(sender, i);
                RoundPlayer2();
            }
            countHorizontal = 0;
            countVertical = 0;
        }
```

در این شرط کلیک بر روی دیوار را بررسی می کنیم.
یک شرط در داخل آن موجود است: متد shortPath در کلاس WallButton را صدا می زند تا در صورت کلیک برای ایجاد دیوار چک کند اگر دیوار قرار بگیرد راهی وجود دارد تا مهره ها به آن سمت صفحه ی بازی بروند. اگر پاسخ True بود دیوار ساخته میشود در غیر این صورت خیر.

اگر بر دیوار کلیک شد و نوبت نفر اول بود یا نفر دوم بود باید دیوار هم داشته باشد.
اگر این شرایط برقرار بود اول با استفاده از کلاس Audio صدای موجود در پکیج Multimedia اجرا میشود. سپس یک دیوار از تعداد دیوارش کم می شود. با CountWood، Label نمایش تعداد چوب های بازیکن ها آپدیت می شوند.
در متد Check رنگ دیوار نارنجی و غیر فعال می شود. در این متد، متد MakeFalse از کلاس WallButton صدا زده می شود. در این MakeFalse دیوار قبل و بعد از دیوار انتخابی غیر فعال خواهد شد.

```
92      } else if (i <= 210) { //Field
93          new Audio( FilePath: "src/Files/Audio/MenschSound.wav");
94          if (isRoundPlayer1) {
95              sender.setBackground(Color.BLUE);
96              if ((i - 127) <= 9) {
97                  isWin( str: addElementToBoard.Player1Name.getText() + " Win");
98              }
99              RoundPlayer1();
100         } else {
101             sender.setBackground(Color.GREEN);
102             if ((i - 127) >= 73) {
103                 isWin( str: addElementToBoard.Player2Name.getText() + " Win");
104             }
105             RoundPlayer2();
106         }
107         isRoundPlayer1 = !isRoundPlayer1;
108         jButtonArrayList.get(position).setBackground(null);
109     }
110     fieldButton = new FieldButton(jButtonArrayList, isRoundPlayer1);
111     setPositionButton(fieldButton.setPositionButton());
112 }
```



شرط دوم مخصوص دکمه هایی برای قرار گیری مهره هاست. مهره ها را با دو رنگ آبی و قرمز مشخص کرده ایم.
با کلیک بر روی محلی که قابلیت قرارگیری مهره برای حرکت را دارد اول با استفاده از کلاس Audio صدایی پخش می کنیم.
سپس جای مهره را تغییر می دهیم. سپس چک می کنیم اگر به آن سمت صفحه رفته بود با استفاده از متد isWin برنده را اعلام می کنیم.
در آخر نوبت را عوض می کنیم سپس دوباره fieldButton را آپدیت می کنیم. با نحوه ی کار این کلاس در ادامه آشنا خواهید شد.
در صورتی که بر روی save کلیک شد از کلاس Save یک شیء ساخته میشود.

وقتی در پنجره ی Welcome بر روی دکمه ی How Play کلیک کنید کلاس PDF فراخوانی می شود.

```
public PDF(String filePath) {
    try {
        File pdfFile = new File(filePath);
        if (pdfFile.exists()) {
            if (Desktop.isDesktopSupported()) {
                Desktop.getDesktop().open(pdfFile);
                JOptionPane.showMessageDialog( parentComponent: null,  message: "The PDF file will be open now.",  title: "Alert", JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog( parentComponent: null,  message: "Awt Desktop is not supported!",  title: "Error", JOptionPane.ERROR_MESSAGE);
            }
        } else {
            JOptionPane.showMessageDialog( parentComponent: null,  message: "File is not exists!\nPlease reaInstall This Game.",  title: "Error", JOptionPane.ERROR_MESSAGE);
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog( parentComponent: null, e.getMessage(),  title: "Error", JOptionPane.ERROR_MESSAGE);
    }
}
```

ابتدا چک می کند که آیا فیل موجود است اگر بود چک می کند در سیستم مورد نظر امکان باز کردن این فایل وجود دارد. اگر وجود داشت فایل را اجرا میکند.

Video Class: با کلیک بر روی About Us کلاس Video فراخوانی میشود. (در این بخش از کتابخانه ی jna و vlcj استفاده شده است.)

```
try {
    SwingUtilities.invokeLater(() -> {
        LibraryOfVLC();
        setTitle("About Us");
        setSize( width: 430,  height: 500);

        Canvas canvas = new Canvas(); //Create a blank rectangular area of the screen
        canvas.setBackground(Color.black);
        JPanel panel = new JPanel();
        panel.setLayout(new BorderLayout());
        panel.add(canvas, BorderLayout.CENTER);
        add(panel, BorderLayout.CENTER); //Add panel to JFrame

        //Initialize The MediaPlayer
        MediaPlayerFactory mediaPlayerFactory = new MediaPlayerFactory();
        //Create a media player instance
        EmbeddedMediaPlayer mediaPlayer = mediaPlayerFactory.newEmbeddedMediaPlayer();
        //Place that will be play video
        mediaPlayer.setVideoSurface(mediaPlayerFactory.newVideoSurface(canvas));

        //Hide the mouse cursor inside JFrame
        mediaPlayer.setEnableMouseInputHandling(true);
        //Disable the keyboard inside JFrame
        mediaPlayer.setEnableKeyInputHandling(false);

        mediaPlayer.prepareMedia(strPath);
        mediaPlayer.play();
    });
```

با instance گرفتن از canvace یک محیط مستطیلی می سازیم تا از آن برای پخش ویدیو استفاده کنیم. سپس یک پنل می سازیم و canvace را در آن می ریزیم. سپس خود پنل را در JFraim قرار می دهیم.
در متد LibraryOfVLC آدرس کتابخونه های VLC را مشخص می کنیم.
بقیه ی توضیحات مورد نیاز کامنت شده اند.

کلاس WallButton:

در کلاس MakeAGraph ابتدا تمامی دکمه های قرارگیری مهره ها به عنوان راس ذخیره می کنیم سپس یال ها را رسم می کنیم.
در متد shortPath با استفاده از کتابخانه ی JGraphT و کلاس DijkstraShortestPath در این کتابخانه می توانیم کوتاه ترین مسیر را پیدا کنیم. در صورتی که مسیری یافت نشود به Exeption بر می خوریم.
با استفاده از متد removeEdge یال بین دو راس مورد نظر پاک خواهد شد.
با استفاده از متد addEdge یالی بین دو راس مورد نظر زده خواهد شد.
با متد MakeFalse با زدن روی دیوار و بعد از عبور با صحت و سلامتی از شروط ذکر شده در توضیح کلاس Board دیوار قبلی و بعدی دیوار زده شده پنهان خواهند شد.

نحوه ی کار کلاس Save و Load:

:SAVE

```
tmpString += Player1Name + "\n" +
    Player2Name + "\n";
```

اطلاعات مربوط به اسامی رو ذخیره میکنیم.

```
tmpString += board.getIsRoundPlayer1() + "\n";
```

اینکه نوبت چه کسی بوده رو با متد ()getIsRoundPlayer که در کلاس بورد هست ذخیره میکنیم.

```
tmpString += PlayerColorRound1.getBackground().getRed() + " . " +
    PlayerColorRound1.getBackground().getGreen() + " . " +
    PlayerColorRound1.getBackground().getBlue() + " . " +
    PlayerColorRound1.isVisible() + " . " +
    PlayerColorRound1.isEnabled() + " . ";
```

در اینجا رنگ خانه و فعال بودن و قابل رویت بودن را برای بازیکن اول ذخیره می کنیم. کد زیر هم برای بازیکن دوم می باشد.

```
tmpString += PlayerColorRound2.getBackground().getRed() + " . " +
    PlayerColorRound2.getBackground().getGreen() + " . " +
    PlayerColorRound2.getBackground().getBlue() + " . " +
    PlayerColorRound2.isVisible() + " . " +
    PlayerColorRound2.isEnabled() + " . ";
```

اما یک دید کلی رو دوباره داشته باشیم در بخش بعدی :

```
32
33          tmpString += board.getPosition() + " . " + WallOfPlayer1 + " . " + WallOfPlayer2 + "\n";
34
35          for (int i = 0; i < jButtonArrayList.size() - 1; i++) {
36              tmpString += jButtonArrayList.get(i).getBackground().getRed() + " " +
37                      jButtonArrayList.get(i).getBackground().getGreen() + " " +
38                      jButtonArrayList.get(i).getBackground().getBlue() + " " +
39                      jButtonArrayList.get(i).isVisible() + " " +
40                      jButtonArrayList.get(i).isEnabled() + "\n";
41          }
42          tmpString += jButtonArrayList.get(jButtonArrayList.size() - 1).getBackground().getRed() + " " +
43                  jButtonArrayList.get(jButtonArrayList.size() - 1).getBackground().getGreen() + " " +
44                  jButtonArrayList.get(jButtonArrayList.size() - 1).getBackground().getBlue() + " " +
45                  jButtonArrayList.get(jButtonArrayList.size() - 1).isVisible() + " " +
46                  jButtonArrayList.get(jButtonArrayList.size() - 1).isEnabled() + "\n";
47
48          for (int i = 128; i < jButtonArrayList.size() - 3; i++) {
49              if ((i - 127) % 9 != 0) {
50                  tmpString += graph.getEdge(jButtonArrayList.get(i), jButtonArrayList.get(i + 1)) + "\n";
51              } else {
52                  tmpString += "Enter\n";
53              }
54          }
55          tmpString += graph.getEdge(jButtonArrayList.get(jButtonArrayList.size() - 3), jButtonArrayList.get(jButtonArrayL
56
```

```
tmpString += board.getPosition() + " . " + WallOfPlayer1 + " . " + WallOfPlayer2 + "\n";
```

در این خط تعداد دیوار های باقیمانده را برای هر بازیکن و هم چنین پوزیشن رو ذخیره می کنیم.

```
for (int i = 0; i < jButtonArrayList.size() - 1; i++) {
   tmpString += jButtonArrayList.get(i).getBackground().getRed() + " " +
         jButtonArrayList.get(i).getBackground().getGreen() + " " +
         jButtonArrayList.get(i).getBackground().getBlue() + " " +
         jButtonArrayList.get(i).isVisible() + " " +
         jButtonArrayList.get(i).isEnabled() + "\n";
}
tmpString += jButtonArrayList.get(jButtonArrayList.size() - 1).getBackground().getRed() + " " +
      jButtonArrayList.get(jButtonArrayList.size() - 1).getBackground().getGreen() + " " +
      jButtonArrayList.get(jButtonArrayList.size() - 1).getBackground().getBlue() + " " +
      jButtonArrayList.get(jButtonArrayList.size() - 1).isVisible() + " " +
      jButtonArrayList.get(jButtonArrayList.size() - 1).isEnabled() + "\n";
```

در اینجا مشخصات مربوط به باتن ها مثل رنگ و ... ذخیره میشه.

```
for (int i = 128; i < jButtonArrayList.size() - 3; i++) {
   if ((i - 127) % 9 != 0) {
      tmpString += graph.getEdge(jButtonArrayList.get(i), jButtonArrayList.get(i + 1)) + "\n";
   } else {
      tmpString += "Enter\n";
   }
}
tmpString += graph.getEdge(jButtonArrayList.get(jButtonArrayList.size() - 3),
jButtonArrayList.get(jButtonArrayList.size() - 2));

for (int i = 128; i < jButtonArrayList.size() - 11; i++) {
   tmpString += graph.getEdge(jButtonArrayList.get(i), jButtonArrayList.get(i + 9)) + "\n";
}
tmpString += graph.getEdge(jButtonArrayList.get(jButtonArrayList.size() - 11),
jButtonArrayList.get(jButtonArrayList.size() - 2));
```

در اینجا یال های گراف را ذخیره میکنیم.

```
try {
         if (!new File("SaveFile.G3M").exists()) {
            new File("SaveFile.G3M");
         }
         JOptionPane.showMessageDialog(null, "Saved");
         FileWriter fileWriter = new FileWriter("SaveFile.G3M");
         fileWriter.write(tmpString);
         fileWriter.close();
      } catch (Exception e) {
         JOptionPane.showMessageDialog(null, "You have an error: " + e.getMessage(), "Alert",
JOptionPane.INFORMATION_MESSAGE);
      }
   }
}
```

کل اطلاعات را در استرینگ tmpString ذخیره کرده و با استفاده از FileWriter ذخیره می کنیم.

:LOAD

خب یک دید کلی تا اینجا :

```
int i = 0, colorR, colorG, colorB, Counter = 0;
try {
   if (!new File("SaveFile.G3M").exists()) {
      JOptionPane.showMessageDialog(null, "You don't saved game yet.", "Alert",
JOptionPane.ERROR_MESSAGE);
   } else {
      Scanner Reader = new Scanner(new File("SaveFile.G3M"));
```

خب در اینجا اول یک سری متغیر برای استفاده در بازیابی رنگ و جایگاه و حرکت ایجاد میکنیم و شروع به خواندن فایلی میکنیم که اطلاعات رو در اون به صورت رشته ذخیره کرده ایم.

```
if (Counter == 0) {
   Player1Name.setText(Reader.nextLine());
   Counter++;
```

در اینجا اطلاعات اسم بازیکن اول رو بازیابی میکنیم .

```
else if (Counter == 1) {
   Player2Name.setText(Reader.nextLine());
   Counter++;
```

بعد از اون در اینجا اطلاعات اسم بازیکن دوم رو بازیابی میکنیم .

```
else if (Counter == 2) {
   board.setIsRoundPlayer1(Reader.nextLine().equals("true"));
   Counter++;
```

در اینجا اطلاعات مربوط به نوبت رو بازیابی میکنیم.

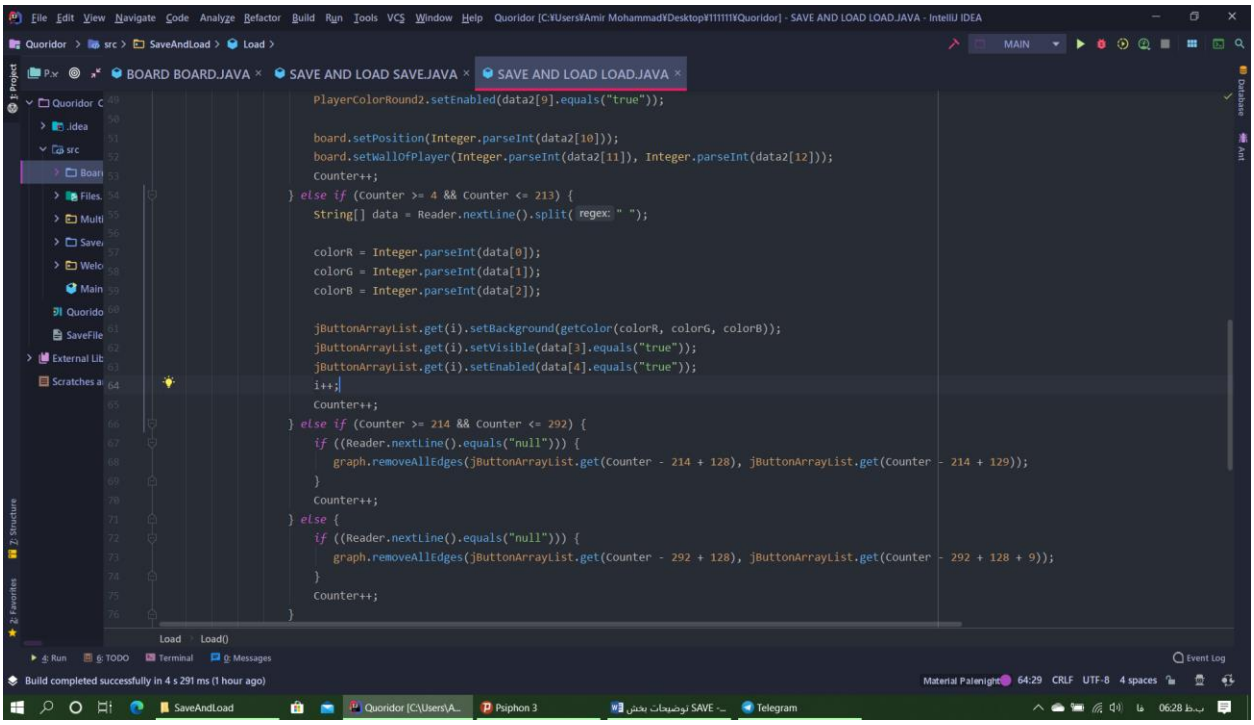
```
else if (Counter == 3) {
   String[] data2 = Reader.nextLine().split(" . ");
   colorR = Integer.parseInt(data2[0]);
   colorG = Integer.parseInt(data2[1]);
   colorB = Integer.parseInt(data2[2]);

   PlayerColorRound1.setBackground(getColor(colorR, colorG, colorB));
   PlayerColorRound1.setVisible(data2[3].equals("true"));
   PlayerColorRound1.setEnabled(data2[4].equals("true"));
```

در اینجا اطلاعات کلی بازیکن اول مثل رنگ و... در محیط بازی رو بازیابی میکنیم.

```
colorR = Integer.parseInt(data2[5]);
colorG = Integer.parseInt(data2[6]);
colorB = Integer.parseInt(data2[7]);
PlayerColorRound2.setBackground(getColor(colorR, colorG, colorB));
PlayerColorRound2.setVisible(data2[8].equals("true"));
PlayerColorRound2.setEnabled(data2[9].equals("true"));
```

اینجا هم همون اطلاعات رو راجع به بازیکن دوم بازیابی میکنیم .

بازم یک دید کلی از بخش بعدی :




```
board.setPosition(Integer.parseInt(data2[10]));
board.setWallOfPlayer(Integer.parseInt(data2[11]), Integer.parseInt(data2[12]));
```

پوزیشن و تعداد دیوار ها رو بازیابی میشوند.

```
else if (Counter >= 4 && Counter <= 213) {
   String[] data = Reader.nextLine().split(" ");

   colorR = Integer.parseInt(data[0]);
   colorG = Integer.parseInt(data[1]);
   colorB = Integer.parseInt(data[2]);

   jButtonArrayList.get(i).setBackground(getColor(colorR, colorG, colorB));
   jButtonArrayList.get(i).setVisible(data[3].equals("true"));
   jButtonArrayList.get(i).setEnabled(data[4].equals("true"));
```

در اینجا هم اطلاعات مربوط به هر دکمه رو بازیابی میکنیم.

```
else if (Counter >= 214 && Counter <= 292) {
         if ((Reader.nextLine().equals("null"))) {
           graph.removeAllEdges(jButtonArrayList.get(Counter - 214 + 128), jButtonArrayList.get(Counter - 214 +
129));
         }
         Counter++;
      } else {
         if ((Reader.nextLine().equals("null"))) {
           graph.removeAllEdges(jButtonArrayList.get(Counter - 292 + 128), jButtonArrayList.get(Counter - 292 +
128 + 9));
         }
         Counter++;
      }
   }
   Reader.close();
}
```

یال های داخل گراف را بازیابی میکنیم و خواندن از روی فایل به اتمام میرسه.

```
} catch (NumberFormatException e) {
   JOptionPane.showMessageDialog(null, "The saved file is corrupt.", "Error", JOptionPane.ERROR_MESSAGE);
} catch (FileNotFoundException e) {
   JOptionPane.showMessageDialog(null, e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
} catch (Exception e) {
   JOptionPane.showMessageDialog(null, "The saved file is corrupt.\n" + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
}
```

در اینجا هم که خطا ها و استثنا ها رو مدیریت میکنیم و پیغام های مناسب رو به کاربر ارسال میکنیم.

```
private Color getColor(int Red, int Green, int Blue) {
   if (Red == 0 && Green == 0 && Blue == 255) {
      return Color.BLUE;
   } else if (Red == 0 && Green == 255 && Blue == 0) {
      return Color.GREEN;
   } else if (Red == 255 && Green == 200 && Blue == 0) {
      return Color.ORANGE;
   }
   return null;
}
```

در اینجا هم رنگ ها رو با توجه به اعدادی که در فایل هست تعیین و تنظیم میکنیم.

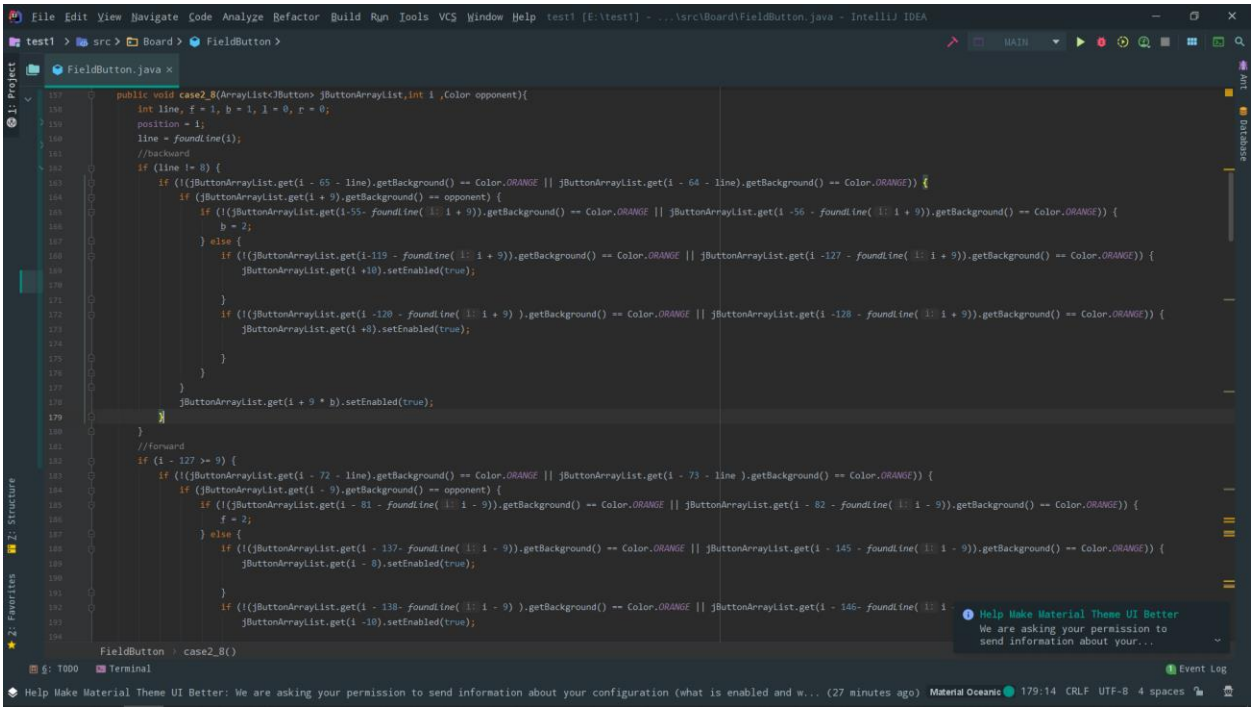
و در آخر: نحوه ی کار کلاس FieldButton

تابع fieldButtonEnableFalse برای اجرای قوانین دسترسی مهره ها به خانه های اطراف است که در این تابع بر اساس ستون ها خانه ها دسته بندی شده اند که هر کدام از ستون ها دارای توابع خاصی (Case0,case1,case2_8)هستند که هر کدام از این توابع برا اساس قوانین بازی به وسیله شرطهایی خانه هایی که هر مهره میتواند به آن دسترسی داشته باشد را مشخص میکند

این تصویر مشخص میکند که در تابع fieldButtonEnableFalse به وسیله switch ستون خانه ای که مهره در آن قرار دارد را مشخص میکنیم و پس از آن تابع مربوط به آن ستون را صدا میزنیم تا خانه هایی که این مهره میتواند ب آن دسترسی دارد را مشخص کند این بخش مربوط به مهره آبی است برای مهره سبز هم به همین صورت عمل میشود

```
37 @    private void doFalse(ArrayList<JButton> jButtonArrayList) {...}
42      protected int setPositionButton() { return position; }
45      public void case1(ArrayList<JButton> jButtonArrayList,int i,Color opponent){...}
157     public void case2_8(ArrayList<JButton> jButtonArrayList,int i ,Color opponent){...}
338 @   public void case0(ArrayList<JButton> jButtonArrayList,int i,Color opponent){...}
442     private void fieldButtonEnableFalse(ArrayList<JButton> jButtonArrayList, boolean isRoundPlayer1) {
            int line;
            if (isRoundPlayer1) {
                for (int i = 128; i < jButtonArrayList.size() - 1; i++) {
                    if (jButtonArrayList.get(i).getBackground() == Color.BLUE) {
                        Color opponent=Color.GREEN;
                        int f = 1, b = 1, l = 0, r = 0;
                        position = i;
                        line = foundLine(i);
                        switch (Math.abs(i - 127) % 9) {
                            case 1:   //forward
                           case1(jButtonArrayList,i,opponent);
                                break;
                            case 2:
                            case 3:
                            case 4:
                            case 5:
                            case 6:
```

این سه تابع مشخص شده در حقیقت توابعی هستند که حانه هایی که در جلو و عقب و راست و چپ مهره میتواند به آنها دسترسی داشته باشد را به وسیله شروط مشخص میکنند که تابع case1 مربوط به خانه های ستون اول از راست و تابعcase2_8 مربوط به خانه های ستونهای ۲تا۸ از سمت راست و تابعcase0 مربوط به خانه های ستون آخر از سمت راست است.



همانطور که در تصویر می بینید تابع case2_8 برای مشخص کردن خانه هایی که مهره به آن دسترسی دارد برای هر کدام از خانه ها که در جهت جلو و عقب هستند شروط خاصی دارد و در عکس پایین نیز شروط خاص مربوط به خانه هایی که در جهت چپ و راست مهره هستند مشخص است و تابع های case۰ و case۱ نیز همانند همین تابع هستند

برنامه نویسان: سید محمد صالح قطبانی، محمد جواد بنان، امیر محمد خسروی