دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده ریاضی و علوم‌کامپیوتر

ساخت بازی شطرنج

نگارش

محمدرضا کتابی‌دلشاد

استاد راهنما

مهدی قطعی

استادمشاور

امین رحمانی

تابستان 1401

به نام خدا

تاریخ:

تعهدنامه اصالت اثر

اینجانب محمدرضا کتابی‌دلشاد متعهد می‌شوم که مطالب مندرج در این پایان نامه حاصل کار پژوهشی اینجانب تحت نظارت و راهنمایی اساتید دانشگاه صنعتی امیرکبیر بوده و به دستاوردهای دیگران که در این پژوهش از آنها استفاده شده است مطابق مقررات و روال متعارف ارجاع و در فهرست منابع و مآخذ ذکر گردیده است. این پایان نامه قبلاً برای احراز هیچ مدرک هم‌سطح یا بالاتر ارائه نگردیده است.

در صورت اثبات تخلف در هر زمان، مدرک تحصیلی صادر شده توسط دانشگاه از درجه اعتبار ساقط بوده و دانشگاه حق پیگیری قانونی خواهد داشت.

کلیه نتایج و حقوق حاصل از این پایان نامه متعلق به دانشگاه صنعتی امیرکبیر می‌باشد. هرگونه استفاده از نتایج علمی و عملی، واگذاری اطلاعات به دیگران یا چاپ و تکثیر، نسخه‌برداری، ترجمه و اقتباس از این پایان نامه بدون موافقت کتبی دانشگاه صنعتی امیرکبیر ممنوع است. نقل مطالب با ذکر مآخذ بلامانع است.

محمدرضا کتابی‌دلشاد

چکیده

در این مقاله به ساختار بازی شطرنج در جاوا پرداخته شده است؛ در هر بخش یـک کـلاس از برنامـه بـه همراه توابع ضمیمه آن بررسی و بیان میشود .

واژه های کلیدی:

Class, Source , Destination, Action listener

فهرست مطالب

صفحه

فصل اول

Classes

Chessman -2-1

یک Abstract Class است که شامل خواص کلی هر مهره مانند موقعیت[1] ، اسم [2]، تصویر[3] و رنگ[4] است .این کلاس شامل تابعی است که به ازای هر subclass از این کلاس پیاده‌سازی میشود. به ازای هر نوع مهره شطرنج کلاسی داریم که از کلاس Chessman ارث‌بری[5] می‌کند. از آنجا که خود Chessman از Jcomponent ارثبری میکند پس هر مهره ای که از Chessman ارثبری میکند از Jcomponent نیز ارثبری میکند.

چون خود مهره ها component هستند میتوانیم به آنها تعدای component از قبیل image اضافه کنیم .

Valid move -2-1-1

تابعی در Chessman است که در آن به ازای هر مهره validationحرکت بررسی میشود (حرکت valid هست یا خیر).

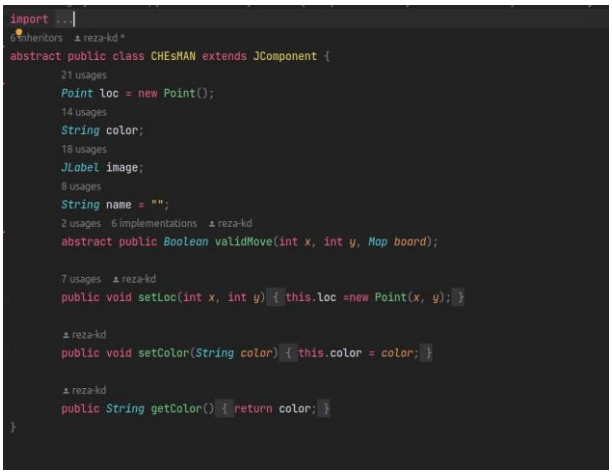
SubClasses of Chessman -2-1-2

به ازای هر نوع مهره یک کلاسی تعریف کرده ایم که از chessman ارثبری میکند و تابع valid move را اختصاصی برای آن مهره پیاده سازی میکنیم که هرکدام دارای یک constructor هستند و موقعیت، رنگ، اسم و تصویر متناسب در آن تنظیم گشته و به component مهره‌ها اضافه می‌شود.

---

[1] location [2] name [5] move validation

[3] image [4] color [6] inheritance



```
import ..
6 inheritors   reza-kd *
abstract public class CHEsMAN extends JComponent {
        21 usages
        Point loc = new Point();
        14 usages
        String color;
        18 usages
        JLabel image;
        8 usages
        String name = "";
        2 usages   6 implementations   reza-kd
        abstract public Boolean validMove(int x, int y, Map board);

        7 usages   reza-kd
        public void setLoc(int x, int y) { this.loc =new Point(x, y); }

        reza-kd
        public void setColor(String color) { this.color = color; }

        reza-kd
        public String getColor() { return color; }
}
```

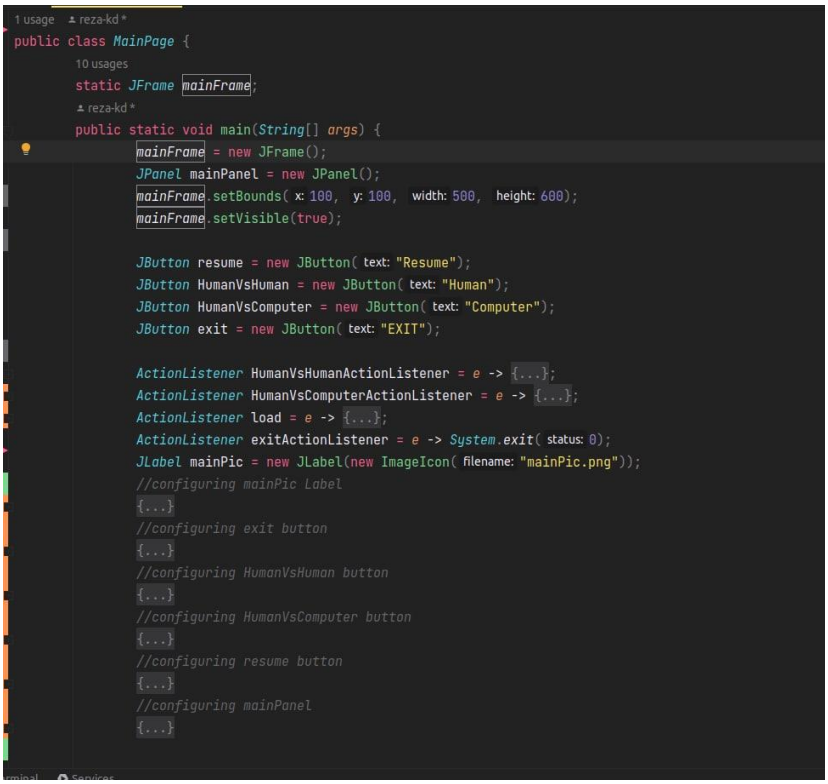
Figure 1

Main Page -2-2

هنگام اجرا شدن این کلاس اجرا می‌شود ،فریم اصلی باز شده و دکمه‌های زیر مشاهده می‌گردد؛

- Human vs Human : بازی دونفره بازیکنها
- Human vs Computer: بازی یک بازیکن با کـامپیوتر . وقتـی صـدا زده میشـود کـلاس setcolor فراخوانده میشود که حاوی یک فریم جدید است و از ما رنگ مهره های کامپیوتر را میپرسد و بازی مربوطه را میسازد.
- Resume: ادامه بازی قبل
- EXIT : خروج از بازی

داخل ActionListener هردکمه گفته شده است که در زمان انتخاب عمل مربوط به آن دکمه اجرا شود.



```
1 usage   reza-kd *
public class MainPage {
        10 usages
        static JFrame mainFrame;
        reza-kd *
        public static void main(String[] args) {
            mainFrame = new JFrame();
            JPanel mainPanel = new JPanel();
            mainFrame.setBounds( x: 100,  y: 100,  width: 500,  height: 600);
            mainFrame.setVisible(true);

            JButton resume = new JButton( text: "Resume");
            JButton HumanVsHuman = new JButton( text: "Human");
            JButton HumanVsComputer = new JButton( text: "Computer");
            JButton exit = new JButton( text: "EXIT");

            ActionListener HumanVsHumanActionListener = e -> {...};
            ActionListener HumanVsComputerActionListener = e -> {...};
            ActionListener load = e -> {...};
            ActionListener exitActionListener = e -> System.exit( status: 0);
            JLabel mainPic = new JLabel(new ImageIcon( filename: "mainPic.png"));
            //configuring mainPic Label
            {...}
            //configuring exit button
            {...}
            //configuring HumanVsHuman button
            {...}
            //configuring HumanVsComputer button
            {...}
            //configuring resume button
            {...}
            //configuring mainPanel
            {...}
```

Figure 2

Map -2-3

در کلاس Map محیط گرافیکی و داده های بازی شکل میگیرد و از فیلدهای زیر تشکیل میشود؛

آرایه دو بعدی 8×8 از مهره ها

آرایه دو بعدی int از جایگاه مهره ها

Jpanel black out : مهره های مشکی که از بازی خارج شده اند

Jpanel white out: مهره های سفیدی که از بازی خارج شده اند

Bottom Panel : حاوی دکمه های save و exit

Whose Turn Panel: نشان میدهد نوبت چه کسی است

Jbotton دو بعدی 8×8 از دکمه ها که مهره ها به آنها اضافه میشوند

پنلی داریم که layout آن 8×8 Grade layout است که map botton به آن اضافه میشود و هر map botton یک action listener دارد که در عمل مربوط به آن نوشته شده است .

Preprocess of map تابعی است که در آن تمام اطلاعات گرافیکی و مختصات مورد نیـاز پنـل هـا داده شده است . layout تنظیم میشود ، رنگ هر component ست میشود و به component هـایی کـه قرار است به component مربوط به خود اضافه شوند را اضافه میکنیم .

Constructors: کانستراکتر اول محیط گرافیکی بازی را از ابتدا (بدون هیچ حرکـت اولیـه ای) میسـازد و هیچ پارامتری دریافت نمیکند . کانستراکتر دوم چند آرایه از ورودی میگیرد ؛ آرایه ای از جایگاه اولیه مهـره ها ، آرایه ای از مهره های سفید بیرون ، مهره های مشکی بیرون و پارامتری مربوط به نوبت حرکت بازیکنها. در این تابع محیط گرافیکی مربوط به این بازی را میسازیم .

Get chessman color: رنگ مهره ای که مختصاتش را میدهیم به ما برمیگرداند.

Main color: صفحه 8×8 که دکمه ها به آن اضافه شده اند (رنگ سیاه یا سفید دارند) را بـه رنـگ اولیـه خود برمیگرداند (رنگ ساده صفحه شطرنجی)

Get bord: آرایه دو بعدی برد را برمیگرداند که میگوید در هر خانه چه مهره ای قرار دارد

این کلاس نیز از Jcomponent ارثبری میکند .

```java
public class Map extends JComponent {
    94 usages
    CHEsMAN[][] map = new CHEsMAN[8][8];
    12 usages
    int[][] board = new int[8][8];
    5 usages
    JPanel blackOut = new JPanel(new GridLayout( rows: 8,  cols: 2));
    5 usages
    JPanel whiteOut = new JPanel(new GridLayout( rows: 8,  cols: 2));
    8 usages
    JPanel buttonPanel = new JPanel();
    8 usages
    JPanel WHOSTURNE = new JPanel(new BorderLayout());
    12 usages
    JLabel turn, wl, bl;
    42 usages
    JButton[][] mapButton;
    5 usages
    JPanel panel = new JPanel(new GridLayout( rows: 8,  cols: 8));
    3 usages
    Font font;
    3 usages
    static final Color WHITE = new Color( r: 243,  g: 198,  b: 89);
    3 usages
    static final Color BLACK = new Color( r: 114,  g: 67,  b: 56);
    2 usages
    static final int[] cntChessman = {0, 1, 1, 2, 2, 2, 8};
    2 usages  reza-kd
    private void preProcessOfMap() {...}
    reza-kd
    public Map(int[][] board, int[] whiteChessman, int[] blackChessman, String whosTurn) throws IOException {...}
    reza-kd
    public Map() throws IOException {...}
    8 usages  reza-kd
    public String getChessmanColor(int x, int y) { return map[x][y].getColor(); }
    2 usages  reza-kd
    public void mainColor() {...}
    reza-kd
    public int[][] getBoard() { return board; }
}
```

Figure 3

Game -2-4

در این کلاس کلیات منطق بازی قرار گرفته است و برای هر بازی جدید باید یک object از این کلاس با مشخصات بازی مربوطه تعریف کنیم .کلاس Game داری فیلد های زیر میباشد

Frame : فریم اصلی بازی

Board : ارایه نگه دارنده اینکه هر جایگاه چه مهره ای دارد.

Type of game : اگر مقدار آن "Both" باشد هر دو بازیکن Human هستند و اگـر رنگـی دریافـت شود آن رنگ مختص به Computer می باشد.

Src and Des: دو آبجکت از جنس پوینت که نشان می دهند مبدا و مقصد حرکت آتی در کجا قرار دارد.

Main Action Listener : اکشن لیسنر دکمه های جدول 8×8

Constructors : کانستراکتر اول بازی را از ابتدا (بدون هیچ حرکت اولیه ای) میسازد ، تایپش را تنظیم میکند و action listener را اضافه میکند.

کانستراکتر دوم مربوط به ادامه بازی است ؛ در وردوی یک "map" داده میشود و از موقعیت اولیه ای کـه دلخواه ما است بازی را میسازد ، تایپش را تنظیم میکند و action listener را اضافه میکند.

addActionListenerToChessman -2-4-1

تابعی است که هر actionlistener را به دکمه های فیلد برد مربوط به آن اضافه میکند.

Move -2-4-2

در این تابع منطق کلی جابجایی شکل میگیرد و انتقال مهره انجام میشود بدین صورت که شرایطی بررسـی میشود و بعد از آن اگر شرایط اتمام بازی مهیا بود ، بازی را تمام میکند و بازنـده را اعـلام میکنـد و در غیـر اینصورت مهره گفته شده را از دو پوینت ذکر شده source را به destination انتقال میدهد. آخـر ایـن تابع change turnزده شده و نوبت عوض میشود.

ChangeTurn -2-4-3

این تابع نوبت بازیکن ها را بررسی میکند و آنرا عوض میکند یعنی اگر اکنون نوبت بـازیکن مشـکی حرکـت کرده باشد نوبت به بازیکن سفید داده میشود . نوبت کسی که حرکت میکند در محیط گرافیکی نوشته شده است و بدین وسیله ببرسی آن امکان پذیر میشود ؛ اگر کسی که باید حرکت کند ربات باشد next move را از رباتی که توسط تدریسیارها نوشته شده است دریافـت میکنـیم و حرکـت را انجـام میـدهیم و در غیـر اینصورت صبر میکنیم تا دکمه ها زده شود Human player مبدا و مقصد حرکت را انتخاب کنـد. – در actionlistener ذکر شده است که اگر مبدا و مقصد مشخص گردید حرکت انجام شود. –

Run -2-4-4

این تابع فریم اصلی بازی و سایز آنرا تنظیم میکند و محیط گرافیکی رو بارگذاری میکند . اگر نوبت ربات بود حرکت بعدی ربات را اجرا میکند.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionListener;
import java.io.FileNotFoundException;
import java.io.IOException;
        reza-kd *
8 usages   reza-kd *
public class Game {
        9 usages
        JFrame frame;
        56 usages
        Map Board;
        7 usages
        String typeOfGame;
        28 usages
        Point src, des;
        2 usages
        static ActionListener mainActionListener;
        2 usages   reza-kd
        public Game(String type) throws IOException {...}
        1 usage   reza-kd
        public Game(Map m, String type){...}
        2 usages   reza-kd *
        public void addActionListenerToChessman(){...}
        2 usages   reza-kd
        public void move() throws InterruptedException {...}
        1 usage   reza-kd *
        public void changeTurn() throws InterruptedException {...}
        3 usages   reza-kd
        public void run() throws InterruptedException {...}
```

Figure 4

Human Player -2-5

این کلاس دارای یک تابع next move است که یک بازی میگیرد و تابع move آن بازی را صدا میزند (وقتی که source و destination انتخاب شده باشند و نوبت بازیکن مربوطه باشد next move فراخوانده میشود)

```
1 usage   reza-kd
public class HMPlayer{
        1 usage   reza-kd
        public static void nextMove(Game game) throws InterruptedException {
                game.move();
        }
}
```

Figure 5

Aiplayer -2-6

این کلاس عوض شده و از پلیر اصلی extend نمیشود بصورتی که یک آبجکت از کلاس Game، یک آرایه دو بعدی از جایگاه مهره ها (داخل هر خانه آرایه مهره قرار گرفته شده ذکر شده است) و سفید بودن یا نبودن ربات را دریافت میکند سپس توابع از قبل پیاده سازی شده -از روی منطقی که از تدریسیاران گرفته شد- را اجرا میکند یعنی حرکت بعدی را میگیرد و source و destination را تشخیص میدهد آنرا بصورت دستی ست میکند . در آخر تابع move صدا زده میشود

```
import java.awt.*;

2 usages   reza-kd
public class AIPlayer{

        2 usages   reza-kd
        public static void nextMove(Game game, int[][] chess, boolean white) throws InterruptedException {...}

}
```

Figure 6

Load -2-7

این کلاس شامل دو تابع زیر میباشد؛

Save -2-7-1

در این تابع یک برد (جایگاه مهره‌ها) ، نوبت بازیکن‌ها و تایپ بازی را دریافت کرده و در فایل ذخیره میکنیم سپس ذکر میکنیم که از هر مهره و به هر تعدادی که داریم در کدام خانه قرار میگیرد .

Load last game-2-7-2

این تابع از فایل بازی آخر را میخواند و یک بازی جدید با اطلاعات فایل میسازد و آنرا اجرا میکند (تابع Run را صدا میزند) . هنگام اجرای بازی بررسی میشود که اگر نوبت ربات است next move را بگیرد و صدا زده میشود (خود بازی خودش را اجرا میکند) و اگر نوبت ربات نباشد منتظرم میمانیم تا Human Player انتخاب کند.

```
import ...
2 usages  reza-kd *
public class Load {
        1 usage  reza-kd
        public static void loadLastGame() throws IOException, InterruptedException {...}
        1 usage  reza-kd
        public static void save(int[][] board, String whosTurn, String type) throws FileNotFoundException {...}
}
```

Figure 7

Animation -2-8

کلاسی است که یک تابع play دارد یک map , destination , source و مهره ای که قرار است جابجا شود را دریافت میکند . منطق این کلاس درست است اما بدلیل مشکلات گرافیکی بدرستی اجرا نمیشود . ابتدا تک تک حرکات افقی را انجام میدهد و بعد از آن تمام حرکات عمودی را اعمال میکند.

```java
import java.awt.*;

import static java.lang.Math.abs;

1 usage  reza-kd *
public class Animation {
    1 usage  reza-kd *
    public static void play(Map Board, Point src, Point des, CHEsMAN srcChessMan) throws InterruptedException {
        Board.map[src.x][src.y] = null;
        int x1 = src.x, y1 = src.y;
        while (abs(des.x - x1) > 0){...}
        while (abs(des.y - y1) > 0){...}
    }
}
```

Figure 8

Select Color -2-9

وقتی بازی همراه با ربات است صدا زده میشود سپس، پنجره جدیدی باز میشود . این پنجره شامل دو دکمه و عبارت "select color" میباشد . رنگ منتخب مربوط به ربات است ؛ رنگ نوشته شده روی دکمه بعنوان رنگ ربات انتخاب میشود و بازی متناسب با آن ساخته و اجرا میگردد .

```
import ...

1 usage  reza-kd *
public class SelectColor {
        1 usage  reza-kd *
        public static void run() throws IOException {

                JFrame frame = new JFrame();
                //configuring frame
                {...}
                JPanel panel = new JPanel();
                //configuring panel
                {...}
                JLabel text = new JLabel( text: "SELECT COLOR");
                //configuring text Label
                {...}
                JButton white = new JButton( text: "White");
                JButton black = new JButton( text: "Black");

                ActionListener selectColorButtonActionlistener = e -> {...};

                //configuring White and Black Button
                {...}

                panel.add(white);
                panel.add(black);
                panel.add(text);
                frame.add(panel);
                frame.revalidate();
        }
}
```

Figure 9

فصل دوم

جمع‌بندی و نتیجه‌گیری و پیشنهادات

جمع‌بندی و نتیجه‌گیری

کلیات بازی ساخته شده بدین صورت است که ابتدا نوع بازی توسط بازیکن انتخاب میشود اگر بازی توسط بازیکنان در حال برگزاری باشد در هر مرحله مهره ها را حرکت میدهند و برحسب نوع حرکت شرایط سنجیده شده و نتیجه اعمال میشود . در غیر اینصورت اگر یک بازیکن خود کامپیوتر باشد رنگ مهره های کامپیوتر انتخاب شده و بعد از حرکت بازیکن ، کامپیوتر حرکت خود را انجام میدهد تا در نهایت برنده و بازنده بازی مشخص شوند.

پیشنهادات

پیشنهاد میشود برای ایجاد پروژه ای جدید به ساختار بازی Backgammon پرداخته شود

منابع و مراجع

Java Tutorial. (n.d.). Retrieved from W3school: https://www.w3schools.com/java/default.asp

Learn Java Programming. (n.d.). Retrieved from javatpoint: https://www.javatpoint.com/java-tutorial

Abstract

This article discusses the structure of the game of chess in Java; In each section, a class of the program with its attached functions is reviewed and expressed.

Key Words:

Class, Source , Destination, Action listener

Amirkabir University of Technology
(Tehran Polytechnic)

Department of Mathematics and Computer Science

The Structure of the Game of Chess

By

Mohamad Reza Ketbei Delshad

Supervisor

Dr.Ghatei

Advisor

Dr.Rahmanei

June 2022