گزارش کد

MatchingGame

پروژه Matching Game یک بازی پیاده‌سازی شده با جاوا است که کاربر در آن می‌تواند از بین تصاویر موجود دو تصویر را انتخاب نموده و در صورت مطابقت آن ها امتیاز مثبت کسب نماید. اگر تصاویر برابر نبودند آنگاه نمره منفی خواهد گرفت. این پروژه شامل ۳ فولدر با نام‌های game، images و sounds است که به ترتیب شامل فایل‌های جاوای برنامه، تصاویر و اصوات مورد استفاده در بازی می‌باشند. در ادامه روند پیاده سازی برنامه با توجه به موارد گفته شده در فایل پروژه کارگاه، توضیح داده خواهد شد.

۱) یک گزارش از کد فعلی تهیه کنید

این گزارش قبلا آماده و تحویل داده شد.

۲) برنامه باید دارای دو امکان یا مد مختلف اجرا باشد، الف) مد بازي نرمال ب) مد بررسی و آزمایش بازی

برای اینکه بتوان دو مد مختلف را انتخاب کرد، یک فرم جدید طراحی شده‌است که شامل سه کلید است. کلید اول برای ورود به مد بازی نرمال، کلید دوم برای ورود به مد بررسی و آزمایش و کلید سوم برای خروج از بازی قرار داده شده‌است. این فرم توسط کلاسی با عنوان ModesFrame.java طراحی شده است که از JFrame ارث‌بری می‌کند و در پکیجی با عنوان game همراه با سایر کلاس‌های دیگر قرار دارد. همانطور که گفته شد، این فرم شامل سه تا کلید است لذا متغیرهای مربوط به پنل آن و سه تا کلید به صورت زیر تعریف می‌شوند:

```
package game;

import java.awt.Color;

public class ModesFrame extends JFrame {

    JPanel mainPanel;
    JButton btnNormalPlay;
    JButton btnTestPlay;
    JButton btnExit;
```

پنل mainPanel، پنل اصلی فرم است و سه تا کلید به آن اضافه میشود. کلید btnNormalPlay برای اجرای مد نرمال بازی، کلید btnTestPlay برای اجرای مد آزمایش بازی و کلید btnExit برای خروج از برنامه استفاده می‌شود. برای اینکه با زدن هرکدام از این کلید ها، وظایف آن‌ها اجرا شود، باید یک ActionListener به آن ها اضافه کنیم و دستور مربوطه اجرا شود. برای کلید btnExit به صورت تصویر زیر است.

با کلیک کردن روی کلید متد actionPerformed اجرا می‌شود و در داخل آن نیز، متد exit فراخوانی می‌گردد. در درون متد exit دستور this.dispose() استفاده شده است و باعث می‌شود که فریم جاری یعنی فریم مدهای مختلف بازی dispose یا به عبارتی بسته شود و برنامه خاتمه یابد.

```
        btnExit = new JButton("Exit");
        btnExit.setBackground(new Color(255, 155, 55));
        btnExit.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                exit();
            }
        });
        mainPanel.add(btnExit);
    }
    private void exit() {
        this.dispose();
    }
```

در مورد کلید btnNormalPlay هم که قرار است بازی را در مود نرمال اجرا کند، باید فرم مربوط به سطح های مختلف بازی را به صورت نرمال نمایش دهد. برای اینکار از یک متغییر Boolean استفاده می‌کنیم که نشان دهد که آیا بازی مد آزمایش است یا در مد نرمال باید اجرا شود. اگر مقدار این متغیر را برابر false قرار دهیم به معنای این است که بازی در مد نرمال باید اجرا بشود. دستور آن به صورت زیر است:

```
        btnNormalPlay = new JButton("Normal Mode");
        btnNormalPlay.setBackground(new Color(55, 155, 255));
        btnNormalPlay.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                showNormalPlay();
            }
        });
        mainPanel.add(btnNormalPlay);
    }
    private void showNormalPlay() {
        new LevelsFrame(false).setVisible(true);
        this.dispose();
    }
```

در داخل متد actionPerformed متد showNormalPlay اجرا می‌شود. در داخل متد showNormalPlay، فرم مربوط به levelهای مختلف بازی ساخته شده و با دستور setVisible(true) نمایش داده می‌شود و فرم مدهای بازی یعنی فرم جاری با دستور this.dispose() بسته می‌شود تا فقط فرم مربوط به levelها در حالت نمایش قرار گیرد. برای ساخت فرم مربوط به levelهای بازی از دستور new LevelsFrame(false) استفاده شده است که یک فرم جدید می‌سازد و متغیر Boolean مربوطه با مقدار false استفاده شده است که نشان دهنده این است که قرار است در مد آزمایش نباشیم و مد نرمال بازی باشد. در مورد این متغیر در کلاس LevelsFrame توضیح داده خواهد شد.

در مورد کلید btnTestPlay هم که قرار است بازی را در مود آزمایش اجرا کند، باید فرم مربوط به سطح های مختلف بازی را به صورت مد آزمایش نمایش دهد. برای اینکار از یک متغییر Boolean استفاده می‌کنیم که نشان دهد که آیا بازی مد آزمایش است یا در مد نرمال باید اجرا شود. اگر مقدار این متغیر را برابر true قرار دهیم به معنای این است که بازی در مد آزمایش باید اجرا بشود. دستور آن به صورت زیر است:

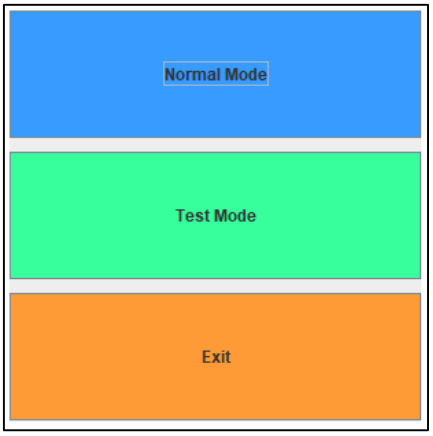
```
    btnTestPlay = new JButton("Test Mode");
    btnTestPlay.setBackground(new Color(55, 255, 155));
    btnTestPlay.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            showTestModePlay();
        }
    });
    mainPanel.add(btnTestPlay);
}
private void showTestModePlay() {
    new LevelsFrame(true).setVisible(true);
    this.dispose();
}
```

با زدن کلید مد آزمایش بازی، متد actionPerformed و در نتیجه متد showTestModePlay اجرا می‌شود. درون این متد با استفاده از دستور new LevelsFrame(true) یک فریم جدید از سطوح مختلف بازی ساخته شده و چون مقدار متغیر برابر true است در نتیجه این فریم در مد آزمایش ساخته و با دستور seVisible(true) نمایش داده می‌شود. دستور this.disposeهم باعث می‌شود که فرم جاری بسته شود و در نتیجه تنها پنجره ای که نمایش داده خواهد شد، پنجره نمایش levelهای مختلف بازی در مد آزمایش خواهد بود.

لازم به ذکر است که در این کلاس، یعنی کلاس ModesFrame، متد main وجود دارد. چراکه قرار است به محض اجرای برنامه، مدهای مختلف طبق صورت سوال گفته شده، به کاربر نمایش داده شود و کاربر آن را انتخاب کند. لذا متد main به صورت زیر پیاده سازی شده است:

```
public static void main(String[] args) {
    new ModesFrame().setVisible(true);
}
```

این متد به محض اجرای برنامه فراخوانی می شود و باعث میگردد که با دستور new ModesFrame یک فرم جدید از مدهای مختلف بازی ساخته شود و با دستور setVisible(true) این فرم ساخته شده، به نمایش در خواهد آمد که شکل آن به صورت زیر است و همانطور که گفته شد شامل سه تا کلید است که با رنگ مختلف مشخص شده است.



نکته دیگر در این کلاس، استفاده از دستوراتی است که باعث می‌شود پنجره فرم مربوطه، در وسط صفحه باز شود. برای اینکار کافیست ابعاد صفحه نمایش را در نظر گرفت، و پنجره را در نصف طول و عرض این ابعاد قرار داد و اندازه فریم را در نظر گرفت. برای اینکار از دستورات زیر استفاده شده است:

```
Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
this.setLocation(dim.width / 2 - this.getSize().width / 2, dim.height / 2 - this.getSize().height / 2);
```

لازم به ذکر است که ازین دستور در تمام فرم های دیگر برنامه استفاده شده است تا تمام فرم ها به صورت شکیل تری در وسط صفحه نمایش، به هنگام باز شدن قرار بگیرند.

نکته بسیار مهم: برای راحتی کار و برای دسترسی آسان تر به مقادیر، یک کلاس با عنوان GameConstants تعریف شده است که در داخل این کلاس تمام مقادیری که ممکن است بارها در حین برنامه نویسی نیاز به تغییر و دسترسی باشد، تعریف شده است. این متغیرها مانند ابعاد فرمها، رنگ کلیدها، آیکن ها، فونت مربوطه و ... در آن به صورت متغییرهای final static تعریف شده اند تا بتوان به راحتی به آن ها دسترسی داشت و به آسانی تغییر داد.

```
public class GameConstants {
    public static final int MAXIMUM_NUMBER_OF_BONUS_IMAGES = 10;
    public static final int BONUS_SCORE = 5;
    public static final int PENALTY_SCORE = 3;
    public static final Dimension MODES_FRAME_DIMENSION = new Dimension(300, 300);
    public static final Dimension LEVELS_FRAME_DIMENSION = new Dimension(800, 300);
    public static final Dimension TITLE_PANEL_DIMENSION = new Dimension(300, 25);
    public static final Dimension CLOSE_DIMENSION = new Dimension(25, 25);
    public static final Dimension FOOTER_PANEL_DIMENSION = new Dimension(25, 25);
    public static final Color DISABLED_LEVEL_BUTTON_COLOR = Color.gray;
    public static final Color LEVEL1_COLOR = new Color(105, 205, 255);
    public static final Color TITLE_PANEL_BACKGROUND = new Color(120, 120, 120);
    public static final Color TITLE_BACKGROUND = new Color(85, 185, 255);
    public static final Color TIMER_BACKGROUND = new Color(255, 85, 185);
    public static final Color BONUS_TEXT_BACKGROUND = new Color(255, 185, 85);
    public static final Color BONUS_BUTTON_BACKGROUND = new Color(85, 255, 85);
    public static final Color TITLE_FOREGROUND = Color.white;
    public static final Color GAME_PANEL_BACKGROUND = new Color(155, 155, 155);
    public static final Color TILES_BACKGROUND = new Color(246, 246, 246);
    public static final Font TITLE_FONT = new Font("Tahoma", Font.BOLD, 14);
    public static final Cursor MOVE_CURSOR = new Cursor(Cursor.MOVE_CURSOR);
    public static final Cursor HAND_CURSOR = new Cursor(Cursor.HAND_CURSOR);
    public static final ImageIcon LOGO_IMAGE = new ImageIcon("src/images/logo.png");
    public static final ImageIcon MATCH_ALL_IMAGE = new ImageIcon("src/images/matchAll.png");
    public static final ImageIcon BONUS_IMAGE = new ImageIcon("src/images/bonus.png");
    public static final ImageIcon CLEAR_HISTORY_IMAGE = new ImageIcon("src/images/clear.png");
    public static final LineBorder HINT_BORDER = new LineBorder(Color.orange, 5);
}
```

۳) در مد بازي نرمال، بايد مراحل مختلف وجود داشته باشد و در ابتدا كه برنامه را اجرا می کنید، بايد يک پنجره نمايش داده شود كه مراحل مختلف بازي را نشان دهد، در حالی كه تنها مرحله اول بازي قابليت باز شدن داشته باشد. مراحل بعدي بازي بايد قفل باشند و پس از اينكه كاربر هر مرحله را با موفقيت به پايان رساند ، می تواند وارد مرحله بعدي بازي شود. تاريخچه مراحلی كه قبلا به صورت موفقيت آميز به پايان رسيده اند بايد پس از خروج و ورود مجدد به برنامه قابل بازيابی باشد و عملا اين تاريخچه با خروج از برنامه از بين نرود. وجود يک دكمه براي پاك كردن اين تاريخچه ضروري است.

برای اینکه یک پنجره نمایش داده شود که مراحل مختلف بازی را نشان دهد، فرمی در کلاسی با عنوان LevelsFrame ساخته می‌شود که شامل ۹ کلید برای ۹ مرحله مختلف بازی، یک کلید برای برگشت به صفحه انتخاب مختلف بازی و یک کلید برای پاک کردن تاریخچه مراحل است. همچنین این کلاس دارای دو متغیر level از جنس int برای ذخیره مرحله بازی و متغیر testMode برای نشان دادن اینکه بازی در مد نرمال است یا مد آزمایش که این دو متغیر در داخل سازنده کلاس مقداردهی می‌شوند. کد این موارد به صورت زیر است:

```
public class LevelsFrame extends JFrame {
    private int level;
    boolean testMode;

    JPanel mainPanel;
    JButton btnLevel1;
    JButton btnLevel2;
    JButton btnLevel3;
    JButton btnLevel4;
    JButton btnLevel5;
    JButton btnLevel6;
    JButton btnLevel7;
    JButton btnLevel8;
    JButton btnLevel9;
    JButton btnBack;
    JButton btnClearHistory;

    public LevelsFrame(boolean isTestMode) {
        this.level = getLevelHistory();
        this.testMode = isTestMode;
        initComponents();
        initLevels(isTestMode);
    }
```

ورودی کانستراکتور کلاس از نوع Boolean است که مقدار isTestMode را دریافت کرده و در متغیر testMode قرار می‌دهد. همانطور که گفته شد این مقدار تعیین کننده مد آزمایش و مد نرمال بازی است. اگر مقدار آن true باشد آنگاه باید مد آزمایش بازی اجرا شود و اگر مقدار آن false باشد باید مد نرمال بازی اجرا شود. با استفاده از متد getLeelHistory، اطلاعات تاریخچه مراحلی که قبلا گذرانده شده است، استفاده می‌شود. از متد initComponents برای ساخت اجزای پنجره و نمایش آن استفاده میگردد و از متد initLevels برای تعیین مرحله بازی باتوجه به مقدار isTestLevels استفاده میگردد که در ادامه توضیح داده می‌شود.

متد getLevelHistory به صورت زیر است. این متد به محض باز شدن پنجره مراحل، در سازنده آن فراخوانی می‌شود و وظیفه آن بازیابی اطلاعات تاریخچه بازی است. به این صورت که اگر برای اولین بار بازی انجام شود و در نتیجه تاریخچه ای وجود نداشته باشد، آنگاه یک فایل متنی ساخته و مقدار ۱ را به عنوان level در این فایل قرار می‌دهد. سپس در مراحل بعدی اجرای برنامه این فایل تاریخچه را خوانده و سپس مقداری که به عنوان level در این فایل قرار دارد، به عنوان خروجی متد برگشت می‌دهد.

```
private int getLevelHistory() {
    File levelHistory = new File("levelsHitory.txt");
    try {
        if (levelHistory.createNewFile()) {
            FileWriter write = new FileWriter(levelHistory);
            write.write(1);
            write.close();
            level = 1;
        } else {
            FileReader reader = new FileReader(levelHistory);
            level = reader.read();
            reader.close();
        }
    } catch (IOException e) {
        JOptionPane.showMessageDialog(this, "Something happen while loading levels history: \n" + e);
    }
    return level;
}
```

در این متد فایل levelsHistory.txtدر صورتی که با استفاده از متد createNewFile() قبلا وجود نداشته باشد، ساخته می‌شود و با استفاده از یک FileWriter مقدار ۱ به عنوان اولین مرحله در آن ذخیره می‌شود. در غیر این صورت اگر فایل قبلا ساخته شده باشد دستورات بخش else اجرا می‌شود و با کمک یک FileReader مقداری که در این فایل ذخیره شده است که قطعا فقط و فقط یک عدد یک رقمی است، از فایل خوانده می‌شود و در متغیر level قرار می‌گیرد. نهایتا متغیر level به عنوان خروجی با استفاده از دستور return level برگشته داده می‌شود.

با کمک متد initComponents تک تک کلیدها شامل ۹ تا کلید مراحل با شماره های ۱ تا ۹ مشخص شده و به پنل mainPanel اضافه می‌شوند. برای هر کلید از یک ActionListener استفاده شده است تا بتوان به محض زدن کلیک بر روی هر کدام از این کلیدها، عملیات مربوط به آن اجرا شود. برای نمونه از دستور زیر برای کلید مرحله ۱ استفاده شده است:

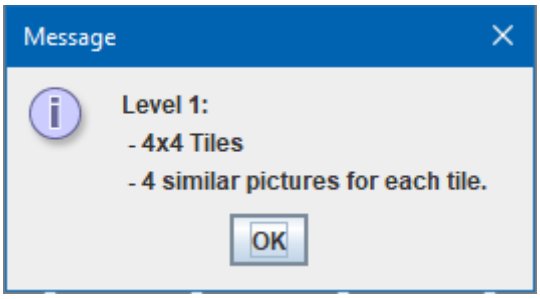
```
btnLevel1 = new JButton("1");
btnLevel1.addActionListener(new ActionListener() {

    @Override
    public void actionPerformed(ActionEvent e) {
        startLevel(1);
    }
});
mainPanel.add(btnLevel1);
```

همانطور که در اینجا دیده می‌شود، به محض کلیک کردن بر روی کلید مرحله ۱، متد actionPerformed فراخوانی شده که درون آن از متد startLevel(1) استفاده شده است. این متد باعث شروع level با مقدار ۱ می شود و یعنی که مرحله ۱ را آغاز کن. از این دستورات برای تمام کلیدهای مراحل استفاده شده است و تنها تفاوت آن این است که برای سایر مراحل، متد startLevel با مقدار مربوط به آن مرحله اجرا می‌شود، برای مثال برای کلید مرحله ۹ دستور startLevel(9) استفاده شده است.

بدنه متد startLevel(int) به صورت زیر است. وظیفه این متد این است که بر اساس مقدار ورودی که میگیرد، پنجره شروع بازی را برای آن مرحله نمایش دهد. لذا در این متد از یک Switch استفاده شده است که بر اساس هر case آن، یک متغیر message مقدار دهی می‌شود.

```
private void startLevel(int level) {
    String message = "";
    switch (level) {
    case 1:
        message = "Level 1:\n" + " - 4x4 Tiles\n" + " - 4 similar pictures for each tile.";
        break;
    case 2:
        message = "Level 2:\n" + " - 6x6 Tiles\n" + " - 2 similar pictures for each tile.";
        break;
        .
        .
        .
    }
    JOptionPane.showMessageDialog(this, message);
    new GameFrame(level, testMode).setVisible(true);
    this.dispose();
}
```

از متغیر messgage برای نمایش پیغام به کاربر استفاده می‌گردد. این پیغام با انتخاب هر مرحله به کاربر نمایش داده می‌شود تا اطلاعاتی در اختیار کاربر قرار دهد. اینکار با دستور showMessageDialog از کلاس JOptionPane انجام می‌گیرد. برای مثال با زدن کلید مرحله ۱ پیغام به صورت زیر است:



این پیغام به کاربر می‌گوید که مرحله ۱ شامل ۴ در ۴ عدد خانه برای بازی است که شامل ۴ تصویر مشابه است (این امکان توسط فایل پروژه کارگاه برای مرحله ۱ در نظر گرفته شده بود). لذا می‌توان گفت که این یک کار اضافه است که در این پروژه انجام شده است. با توجه به هر مرحله و توضیحات آن در فایل پروژه کارگاه، یک پیغام مناسب به کاربر نمایش داده می‌شود. سپس با دستور new Game فریم اصلی بازی ساخته شده و با دستور setVisible(tue) نمایش داده می‌شود. با دستور this.dispose پنجره مربوط به انتخاب مراحل بسته می‌شود تا فقط پنجره بازی نمایش داده شود.

همانظور که گفته شد، علاوه بر ۹ کلید مراحل، دو کلید دیگر نیز به پنل اضافه شده است. با کمک دستورات زیر کلید clearHistory ساخته شده و با کمک متد setIcon یک تصویر با عنوان clear.png که در فولدر images قرار دارد به آن اضافه می‌شود تا شکیل تر به نظر برسد:

```
btnClearHistory = new JButton();
btnClearHistory.setBackground(GameConstants.CLEAR_HISTORY_BUTTON_BACKGROND);
btnClearHistory.setIcon(GameConstants.CLEAR_HISTORY_IMAGE);
btnClearHistory.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        clearHistory();
    }
});
mainPanel.add(btnClearHistory);
```

همانطور که دیده می‌شود، با کلیک کردن روی این کلید، متد clearHistory() اجرا می‌شود. وظیفه این متد این است که ابتدا از کاربر با کمک showConfirmDialog از کلاس JoptionPane بپرسد که آیا تمایل به حذف تاریخچه دارد، در صورت انتخاب گزینه Yes آنگاه فایل مربوط به تاریخچه حذف شده و پنجره مراحل بسته می‌شود و پنجره مربوط به مدهای مختلف بازی نمایش داده می‌شود:

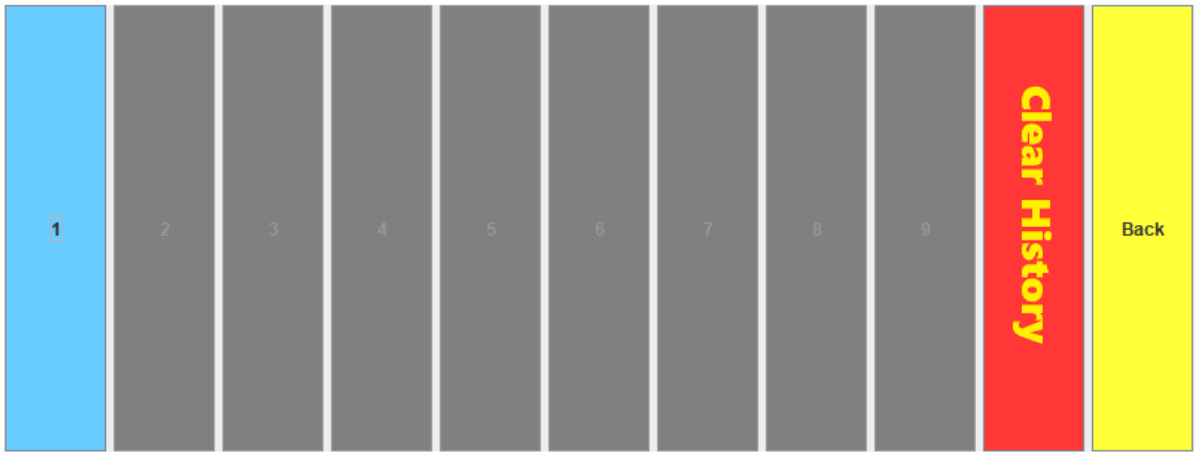
```
private void clearHistory() {
    int option = JOptionPane.showConfirmDialog(this, "Are you sure to delete Levels History?", "Delete History",
            JOptionPane.YES_NO_OPTION);
    if (option != JOptionPane.YES_OPTION)
        return;
    File levelHistory = new File("levelsHitory.txt");
    levelHistory.delete();
    new ModesFrame().setVisible(true);
    this.dispose();
}
```

نهایتا کلید btnBack به پنل با دستورات زیر اضافه می‌شود:

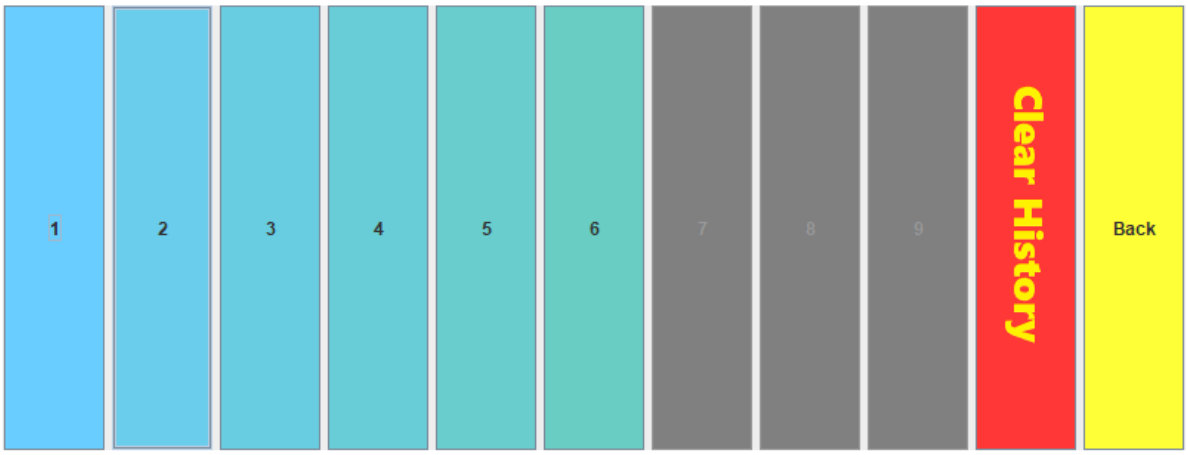
```
    btnBack = new JButton("Back");
    btnBack.setBackground(GameConstants.BACK_BUTTON_BACKGROND);
    btnBack.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            back();
        }
    });
    mainPanel.add(btnBack);
}

private void back() {
    new ModesFrame().setVisible(true);
    this.dispose();
}
```

با دستور setBackground رنگ زمینه این کلید از کلاس GameConstants انتخاب شده و به آن اعمال می‌گردد. در حال حاضر این رنگ زرد است و می‌توان تغییر داد. همچنین با زدن روی این کلید متد back() اجرا می‌شود که وظیفه آن نمایش پنجره انتخاب مدهای مختلف بازی با دستور new ModesFrame().setViible(true) و بستن پنجره مراحل بازی با دستور this.dispose است.

پس شکل ظاهری این پنجره در حالتی که متغیر isTestMode برابر با false باشد یعنی بازی در مد نرمال اجرا شود، به صورت زیر خواهد بود. در ابتدای کار چون فقط مرحله ۱ قابل اجرا است، لذا فقط کلید مرحله ۱ فعال است. سایر کلیدها غیر فعال بوده و به رنگ خاکستری هستند. وظیفه فعال و غیر فعال سازی این کلید ها با توجه به مراحل، برعهده متد initLevels است. در این متد با استفاده از مقدار متغییر level، مشخص می‌کند که چه کلیدهایی باید فعال و چه کلیدهایی غیر فعال باشد. برای مثال اگر مقدار level برابر با ۶ باشد، آنگاه باید کلید های ۱ تا ۶ فعال شده و رنگ آن ها از حالت خاکستری به آبی تغییر کند. لازم به ذکر است که این رنگ آبی قابل تغییر دادن در کلاس GameConstants می‌باشد. تمام کلیدها از رنگ بی با یک مقدار تیره تر استفاده می‌کنند، یعنی کلید ۲ از کلید ۱ تیره تر، کلید ۳ از کلید ۲ و ۱ تیره تر و الی آخر .

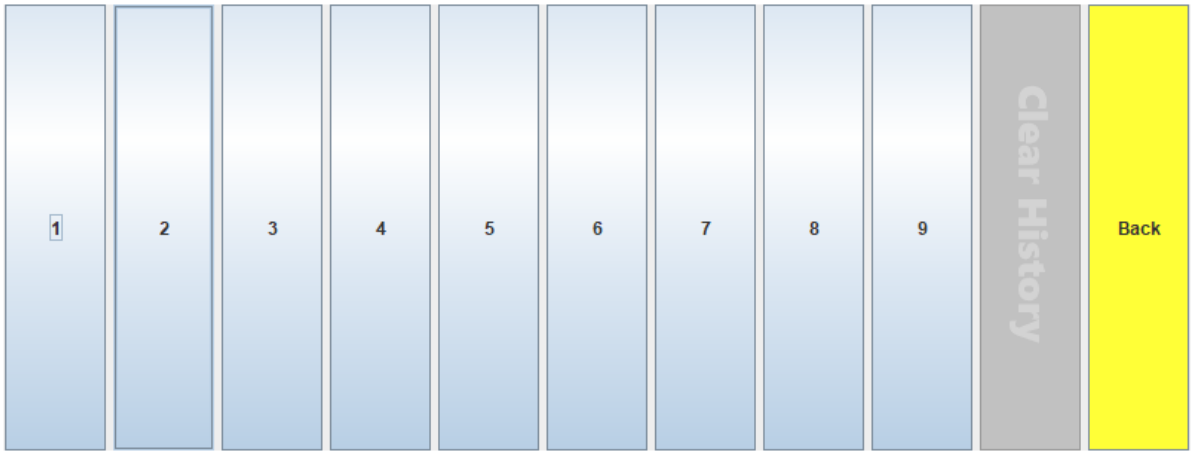


برای نمونه در حالتی که کاربر تا مرحله ۶ پیشرفته باشد، آنگاه پنجره برنامه به شکل زیر خواهد بود:



۴) مد بررسی و آزمایش بازي: براي بررسی پروژه امکان ورود به مراحل مختلف بازي بدون اجراي مراحل قبل وجود داشته باشد و بتوان در هر مرحله اي که بخواهیم بدون طی شدن مراحل قبل وارد شویم. در ضمن بتوان مقادیر مرحله بازي شده فعلی، امتیاز و امتیاز ویژه را تنظیم نمود.

اگر متغیر isTestMode برابر true باشد یعنی در مد آزمایش بازی قرار داریم و لذا باید تمام مراحل فعال باشند، پس در نتیجه متد initLevels کاری انجام نداده و سریع خارج می‌شود. در این حالت تمام کلیدها فعال هستند و رنگ آنها به حالت پیشفرض خواهد بود و تغییری نخواهند کرد که به صورت زیر خواهد بود:



لازم به ذکر است که کلید clear history غیر فعال می‌شود چراکه تاریخچه ای د مد آزمایش در نظر گرفته نمی‌شود تا بتوان آن را پاک نمود.

۵) مرحله ۱ بازي: صفحه بازي به صورت ۴*۴ شود و از هر تصوير چهار نمونه در صفحه بازي وجود داشته باشد

در ادامه این گزارش، مراحل مختلف بازی توضیح داده می‌شود. در هر مرحله یک یا چند متغیر جدید به برنامه اضافه شده است تا بتواند شرایط برنامه را مدیریت نماید. برای نمونه در مرحله یک قرار است که بازی شامل ۴*۴ اجرا شود و از هر تصویر ۴ نمونه در صفحه بازی وجود داشته باشد. برای نشان دادن تعداد خانه های جدول و تعداد تصاویر، از متغیرهای numberOfPictures و numberOfRows و numberOfColumns و برای امتیاز بازی از متغیر score و برای مشخص کردن مد بازی از متغیر isTestMode استفاده می‌شود که در تصویر زیر و در ابتدای کلاس GameFrame تعریف می‌گردند:

```
public class GameFrame extends javax.swing.JFrame implements ActionListener {
    private int gameLevel;
    private int numberOfPictures;
    private int numberOfRows;
    private int numberOfColumns;
    private int score;
    private boolean isTestMode;
```

این متغییرها با توجه به شرایط هر مرحله از بازی مقدار دهی می‌شوند. برای مرحله اول باید تعداد ردیف ها برابر ۴، تعداد ستون ها برابر ۴ و تعداد نمونه از هر تصویر برابر با ۴ باشد که این مقادیر از طریق سازنده کلاس تنظیم می‌شوند:

```
public GameFrame(int level, boolean testMode) {
    gameLevel = level;
    this.isTestMode = testMode;
    if (gameLevel == 1) {
        numberOfPictures = 4;
        numberOfRows = 4;
        numberOfColumns = 4;
    }
```

حال ازین مقادیر برای ساخت مرحله بازی استفاده می‌شود. از ضرب تعداد ردیف ها در تعداد ستون ها، تعداد کل خانه های مرحله به دست می‌آید. یعنی در مرحله یک که ۴*۴ است شامل ۱۶ خانه می‌شود و این خانه ها از طریق آرایه‌ای از کلاس Tile ساخته می‌شوند. همچنین تعداد کل تصاویری که در بازی باید قرار گیرد، شامل خود تصویر و نمونه های تصویر می‌شود. لذا تعداد کل تصاویر برابر است با تعداد کل خانه ها، تقسیم بر تعداد نمونه تصویر است. در این مرحله تعداد کل خانه ها برابر است با اندازه آرایه tiles یعنی ۱۶ تقسیم بر تعداد نمونه تصاویر یعنی ۴ که نتیجه برابر است با ۴. پس در این بازی باید حداقل ۴ تصویر مختلف قرار داده شود تا بتوان یک بازی ۱۶ تصویری با ۴ تصویر تکراری ساخت. لذا تصاویر را در آرایه ای از نوع ImageIcon قرار می‌دهیم که اندازه آن به صورت زیر است:

```
tiles = new Tile[numberOfRows * numberOfColumns];
icons = new ImageIcon[tiles.length / numberOfPictures];
```

حال ساختار پنجره مرحله و المان های آن از طریق متد initComponents انجام می‌شود که پس از اجرای برنامه در مرحله ۱ به صورت زیر خواهد بود.

نکته بسیار مهم: تمام مراحل بازی شامل یک پنل titlePanel ، یک پنل gamePanel و یک پنل footerPanel می‌شوند.

پنل titlePanel:

در پنل titlePanel، یک جعبه متنی Jtextfield با عنوان title قرار می‌گیرد که امتیاز بازی را نشان می‌دهد. همچنین در سمت راست این پنل یک کلید قرمز رنگ با علامت X وجود دارد که برای بستن مرحله استفاده می‌شود. همچنین در سمت چپ این پنل کلید علامت سوال سبز رنگ وجود دارد که برای نمایش help به کاربر استفاده می‌گردد. این سه مورد به صورت زیر پیاده سازی شده‌اند. مقدار رنگ title که در اینجا آبی است از طریق کلاس GameConstants و متد setBackground مقدار دهی می‌شود. همچنین فونت مربوط به این متن از طریق متد setFont تغییر داده شده‌است. تنها نکته این جعبه متنی این است که باید در حالت مد نرمال غیر قابل ویرایش باشد و وظیفه آن فقط نمایش score باشد. برای اینکار از دستور title.setEditable(false) استفاده مشده است که باعث می‌شود کادر متنی قابل ویرایش نباشد. اما در مد آزمایش و بررسی بازی باید بتوان مقدار امتیاز را تعیین کرد. لذا در مد آزمایش این کادر متنی باید قابل ویرایش باشد و سپس پس از زدن کلید Enter باید مقدار امتیاز جدید ثبت شود. لذا برای اینکار یک ActionListener در مد آزمایش بازی یعنی زمانی که isTestMode برابر true است، به جعبه متنی اضافه می‌شود. با زدن کلید Enter متد actionPerformed اجرا می‌شود و درون این متد نیز متد setScoreInTestMode() فراخوانی می‌شود که وظیفه آن تغییر مقدار score باتوجه به مقدار جدیدی است که وارد می‌شود. برای دو کلید close و help نیز که از جنس Label هستند، یک MouseListener اضافه شده است تا به محض کلیک کردن روی آن ها، عملیاتی انجام شود. با کلیک بر روی علامت X گوشه راست برنامه، متد closeMouseClicked فراخوانی می‌شود که وظیفه آن بستن پنجره مرحله و باز کردن پنجره مراحل است. با زدن کلیک بر روی علامت سمت چپ بالا، متد helpMouseClicked فراخوانی می‌شود که وظیفه آن، نمایش کل تصاویر بازی به عنوان help به کاربر است.

```
title = new JTextField();
title.setBackground(GameConstants.TITLE_BACKGROUND);
title.setFont(GameConstants.TITLE_FONT);
title.setForeground(GameConstants.TITLE_FOREGROUND);
title.setHorizontalAlignment(JTextField.CENTER);
title.setText("Score: " + score);
title.setToolTipText("After Clicking mouse here use arrow keys to move");
title.setBorder(null);
title.setCursor(GameConstants.MOVE_CURSOR);
if (isTestMode == false) {
    title.setEditable(false);
    title.setHighlighter(null);
}
if (isTestMode)
    title.addActionListener(new ActionListener() {

        @Override
        public void actionPerformed(ActionEvent e) {
            setScoreInTestMode();
        }
    });
```

```
close = new JLabel();
close.setFont(GameConstants.TITLE_FONT);
close.setForeground(Color.red);
close.setHorizontalAlignment(SwingConstants.CENTER);
close.setText("X");
close.setToolTipText("Close");
close.setCursor(GameConstants.HAND_CURSOR);
close.setPreferredSize(GameConstants.CLOSE_DIMENSION);
close.addMouseListener(new MouseAdapter() {
    public void mouseClicked(MouseEvent evt) {
        closeMouseClicked(evt);
    }
});
titlePanel.add(close, BorderLayout.LINE_END);
help = new JLabel();
help.setFont(GameConstants.TITLE_FONT);
help.setForeground(Color.green);
help.setHorizontalAlignment(SwingConstants.CENTER);
help.setText("?");
help.setToolTipText("Click to see Images");
help.setCursor(GameConstants.HAND_CURSOR);
help.setPreferredSize(GameConstants.CLOSE_DIMENSION);
help.addMouseListener(new MouseAdapter() {
    public void mouseClicked(MouseEvent evt) {
        helpMouseClicked(evt);
    }
});
titlePanel.add(help, BorderLayout.LINE_START);
```

```
private void helpMouseClicked(java.awt.event.MouseEvent evt) {
    if (evt.getButton() == MouseEvent.BUTTON1)
        if (helping == false) {
            showHelp();
            helpTimerCounter = 10;
            helping = true;
            title.setText("Timer: " + helpTimerCounter);
            helpTimer.start();
        }
}
private void showHelp() {
    for (int i = 0; i < tiles.length; i++) {
        if (tiles[i].isNoImage() == false) {
            tiles[i].showTile();
            tiles[i].setEnabled(false);
        }
    }
}
private void hideHelp() {
    for (int i = 0; i < tiles.length; i++) {
        if (tiles[i].isNoImage() == false) {
            tiles[i].setEnabled(true);
            tiles[i].hideTile();
        }
    }
}
```

با زدن کلید help متد helpMouseClicked فراخوانی می‌شود. وظیفه این متد این است که تصاویر تمام خانه ها را نمایان سازد. برای اینکار از متدی با عنوان showHelp استفاده می‌کند. در این متد و به کمک یک حلقه for، تمام خانه‌هایی را که تا به الان نمایان نشده‌اند و در بازی وجود دارند، با فراخوانی متد showTile نمایان می‌سازد. از متغیر Boolean با عنوان helping به این دلیل استفاده شده است که کاربر نتواند زمانی که متد showHelp در حال نمایش تصاویر است، مجددا کلید help را بزند. همچنین باید زمان مشخصی این عملیات راهنمایی انجام شود. برای اینکار از یک متغیر int با عنوان helpTimerCounter و یک زمان‌سنج جاوا از جنس Timer با عنوان helpTimer استفاده می‌شود و به محض زدن کلید help، این تایمر فعال شده و مقدار helpTimerCounter را یکی یکی از مقدا ۱۰ کاهش می‌هد تا به مقدار ۰ برسد. این عملیات هر یک ثانیه یک بار انجام می‌شود و این یعنی به مدت ۱۰ ثانیه، تمام تصاویر برای راهنمایی کاربر و به عنوان help به وی نمایش داده می‌شود. مقدار زمان هم در فیلد title نمایش داده می‌شود و هر یک ثانیه یک بار یک عدد کم می‌شود. این کار در متد initTimer انجام شده و توضیحات آن در ادامه گزارش آمده است.

باز دن کلید close نیز متد closeMouseClicked به صورت زیر فراخوانی می‌شود.

```
private void closeMouseClicked(MouseEvent evt) {
    if (evt.getButton() == MouseEvent.BUTTON1) {
        if (isTimerEnabled)
            timer.stop();
        new LevelsFrame(isTestMode).setVisible(true);
        this.dispose();
    }
}
```

با استفاده از دستور evt.getButton() می‌توان کلیک ماوس را تشخیص داد. همچنین در مرحله هفت بازی قرار است یک تایمر کلی به بازی اضافه شود، لذا در صورتی که این تایمر فعال باشد، باید آن را با دستور stop متوقف کرد. سپس با دستر new LevelsFramer پنجره کلی مراحل ساخته شده و با دستور setVisible(true) نمایش داده می‌شود. با دستور this.dispose نیز، پنجره جاری، یعنی پنجره مرحله مربوطه بسته می‌شود.

پنل gamePanel:

در پنل gamePanel، تمام تصاویر بازی باید چیده شود که شامل خود تصاویر و نمونه های آن می‌شود. در واقع در این پنل، تمام خانه ها یا tileهای بازی چیده می‌شود. لذا باید این پنل دارای چیدمان جدولی به طول numberOfrRows و عرض numberOfCulmns باشد. لذا به صورت زیر تعریف می‌شود:

```
gamePanel = new JPanel();
gamePanel.setBackground(GameConstants.GAME_PANEL_BACKGROUND);
if (gameLevel == 1)
    gamePanel.setPreferredSize(new Dimension(numberOfRows * 105, numberOfColumns * 105));
else
    gamePanel.setPreferredSize(new Dimension(numberOfRows * 85, numberOfColumns * 85));

gamePanel.setLayout(new GridLayout(numberOfRows, numberOfColumns, 5, 5));
getContentPane().add(gamePanel, BorderLayout.CENTER);
```

با دستور setBackground رنگ زمینه این پنل تغییر پیدا کرده است. با شرط if(gameLevel == 1) بررسی می‌کنیم که آیا مرحله ای که بازی می‌شود، مرحله یک است یا خیر. چون در ملحه ۱ فقط ۴*۴ خانه داریم در نتیجه اندازه پنل باید بزرگتر باشد، لذا با دستور setPreferredSize یک ابعاد جدید با ضریب ۱۰۵ که به صورت آزمایش و خطا به دست آمده است به پنل می‌دهیم. در غیر این صورت و برای مراحل بعد که شامل ۶*۶ یا ۸*۸ خانه است، باید اندازه پنل کمی کوچکتر باشد تا تمام خانه ها و صفحه بازی به صورت شکیل تری نمایش داده شود، لذا در این شرایط از ضریب ۸۵ که از طریق آزمایش و خطا به دست آمده‌است، استفاده می‌کنیم. نهایتا با دستور setLayout یک چیدمان جدولی از نوع GridLayout به طول و عرض گفته شده، برای این پنل تخصیص داده می‌شود.

حال برای قرار دادن تصاویر و مقداردهی خانه‌های این پنل، از دو متد initIcons و initTiles استفاده می‌کنیم که به صورت زیر است:

```
private void initIcons() {
    ImageIcon img;
    for (int i = 0; i < icons.length; i++) {
        img = new ImageIcon(getClass().getResource("/images/img" + i + ".png"));
        icons[i] = img;
    }
}
```

در این متد با استفاده از یک حلقه for، روی آرایه icons حرکت کرده و به ازای هر خانه i، یک شی ازکلاس ImageIcon ساخته شده که تصویر آن از طریق متد getClass().getResource با آدرس مشخص که در مسیر images و با نام‌های img0.png، img1.png، img2.png و ... وجود دارد، در داخل خانه icons[i] قرار می‌گیرد. با اینکار تمام تصاویر بازی که در فولدر images قرار دارند و نام آن ها با img شروع می‌شود در آرایه icons قرار می‌گیرد. دقت شود که در مرحله ۱ ما به اندازه ۴ تصویر نیاز داریم، چون اندازه آرایه icons برابر با ۴ است اما مثلن در مرحله ۲ به ۶*۶ تقسیم بر ۲ که تعداد نمونه های تصویر است، یعنی ۱۸ تصویر از شماره img0.png تا شماره img17.png ،نیاز خواهیم داشت. پس از اینکه این تصاویر را در آرایه icons قرار دادیم، آنگاه باید آن‌ها را در خانه های بازی یعنی tileها قرار دهیم که اینکار از طریق متد initTiles به صورت زیر انجام می‌شود:

```
private void initTiles() {
    int duplicateImages = 0;
    for (int i = 0; i < tiles.length; i++) {
        tiles[i] = new Tile(icons[duplicateImages], GameConstants.LOGO_IMAGE);
        tiles[i].addActionListener(this);
        tiles[i].setDisabledIcon(tiles[i].getImage());
        tiles[i].setBackground(GameConstants.TILES_BACKGROUND);
        if ((i + 1) % numberOfPictures == 0) {
            tiles[i].setMatchTile(tiles[i - 1]);
            tiles[i - 1].setMatchTile(tiles[i]);
            duplicateImages++;
        }
    }
    shuffleTiles();
}
```

با استفاده از یک حلقه for روی تمام tileها حرکت کرده و یک آبجکت جدید با استفاده از دستور new Tile از کلاس Tile می‌سازیم. این آبجکت شامل تصویر اول و تصویر دوم است. درواقع تصویر اول تصویر back و تصویر دوم تصویر front است که جنس آن ها از نوع ImageIcon است. تصویر پشت کارت که قرار است کاربر با کلیک روی خانه آن را مشاهده نماید، از طریق آرایه icons حاصل می‌شود. تصویر جلو یعنی تصویری که در ابتدای کار قبل از اینکه کاربر روی خانه کلیک کند، به کاربر نمایش داده می‌شود، تصویر یک علامت سوال است که با نام logo در فولدر images وجود دارد. به ازای هر تصویر، خانه بعدی که از طریق duplicateImage مشخص می‌شود نیز شامل تصویر نمونه می‌شود. برای مثال اگر از هر تصویر دو نمونه باید وجود داشته باشد، آن گاه در واقع خانه های 0 و 1 شامل یک تصویر، خانه 2 و 3 شامل یک تصویر و ... می‌باشد. همچنین با کمک متد setMatchTile برای هرکدام از tileها، خانه‌ای که با آن مطابقت دارد مشخص می‌کنیم. برای مثال خانه مطابق با tile[0] برابر با tile[1] و برعکس است. همانطور که می‌دانیم، کلاس Tile ز نوع JButton است لذا برای اینکه به محض کلیک بر روی خانه های بازی، عملیات مربوط به نمایش تصاویر انجام شود، یک ActionListener از جنس کلاس جاری با this به خانه‌ها اضافه می‌کنیم تا نهایتا با کلیک بر روی آن‌ها، متد actionPerformed کلاس اجرا گردد که در ادامه خواهیم دید.با کمک متد setDisabledIcon ، تصویری که در حالت غیرفعال بودن Button یا خانه جدول، نمایش داده شودف تعیین میگردد. همانطور که در متد showHelp دیدیم، با زدن کلید help تمام خانه ها یا tile ها به حالت غیرفعال با دستور setEnabled(false) در میایند و در نتیجه تصویری که برای حالت غیرفعال در نظر گرفته شده است، نمایش داده می‌شود که در واقع همان تصویر پشت خانه است که باید به کاربر جهت راهنمایی نمایش داده شود. نهایتا با کمک متد shuffleTiles() ترتیب خانه های بازی به صورت تصادفی بهم ریخته می‌شود تا بازی جذاب شده وشکل نهایی خود را بگیرد.

پنل footerPanel:

پنل footerPanel در پایین صفحه قرار دارد و شامل یک کلید برگشت به صفحه مراحل با عنوان Back است. همچنین این پنل بسته به مرحله بازی ممکن است شامل یک یا چند المان دیگر باشد. برای مثال در مرحله پنج کلید hint به این پنل اضافه می‌شود که با زدن آن به کاربر راهنمایی می‌دهد. ساختار این پنل به صورت زیر است:

```
footerPanel = new JPanel();
footerPanel.setPreferredSize(GameConstants.FOOTER_PANEL_DIMENSION);
footerPanel.setLayout(new GridLayout(1, 2, 5, 0));
back = new JButton();
back.setBackground(GameConstants.TITLE_BACKGROUND);
back.setFont(GameConstants.TITLE_FONT);
back.setForeground(GameConstants.TITLE_FOREGROUND);
back.setText("Back to Levels");
back.setCursor(GameConstants.HAND_CURSOR);
back.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        backActionPerformed(evt);
    }
});
footerPanel.add(back);
getContentPane().add(footerPanel, BorderLayout.SOUTH);
```

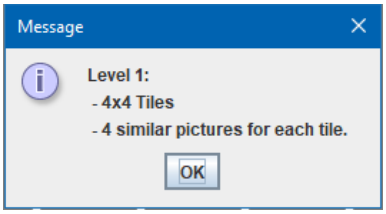
با زدن کلید back متد backActionPerformed اجرا می‌شود که به صورت زیر است:

```
private void backActionPerformed(ActionEvent evt) {
    new LevelsFrame(isTestMode).setVisible(true);
    this.dispose();
}
```

همانطور که دیده می‌شود، با اجرا شدن این متد، پنجره مراحل با دستور new LevelsFrame ساخته شده و با دستور setVisible(true) به نمایش در می‌آید. سپس با دستور this.dispose پنچره جاری یعنی پنجره بازی در مرحله مربوطه بسته می‌شود.

تصاویر بازی در مرحله ۱

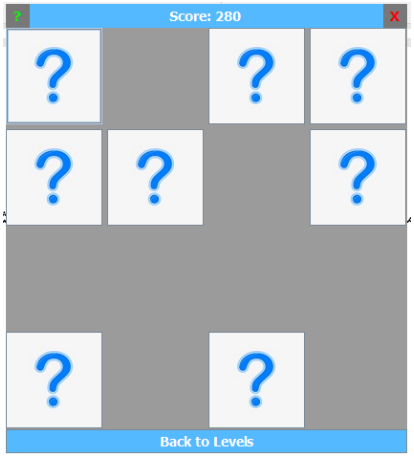
پس از انتخاب مرحله ۱، پیغام زیر به کاربر نمایش داده می‌شود:



سپس پنجره بازی به صورت زیر نمایش داده می‌شود:

با زدن کلید help درگوشه بالا سمت چپ، تمام تصاویر بازی (شامل هر تصویر ۴ نمونه)نمایش داده می‌شود:

با درست حدس زدن چند مورد و انتخاب خانه‌ها و نمونه‌ها به درستی، آنگاه آن خانه حذف شده و امتیاز داده می‌شود:

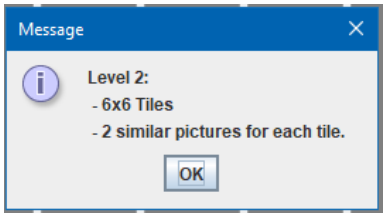


۶) مرحله ۲ بازي: صفحه ۶*۶ (کد موجود)

در این مرحله، همان کدهای قبلی اجرا می‌شود که توضیحات کامل آن در گزارش کد فعلی ارائه شد. برای تعیین مقادیر تعداد ستون‌ها و سطرهای بازی و تنظیمات آن، در سازنده کلاس GameFrame، به صورت if زیر بررسی می‌گردد:

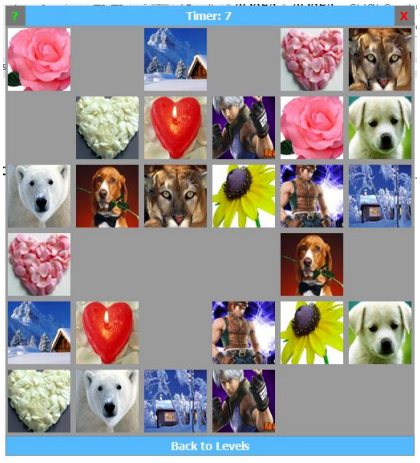
```
public GameFrame(int level, boolean testMode) {
    gameLevel = level;
    this.isTestMode = testMode;

    if (gameLevel == 2) {
        numberOfPictures = 2;
        numberOfRows = 6;
        numberOfColumns = 6;
    }
```

تعداد نمونه تصاویر ۲ است و اندازه جدول ۶ در ۶ خواهد بود یعنی باید ۱۸ تصویر وجود داشته باشد. سایر توضیحات مانند مرحله ۱ است. با انتخاب مرحله ۲، پیغام به صورت زیر نمایش داده می‌شود:



تصویر بازی در این مرحله به صورت زیر خواهد بود:

با انتخاب کلید help و نمایش تمام تصاویر ، پس از انتخاب درست یک یا چند خانه، به صورت زیر خواهد بود:

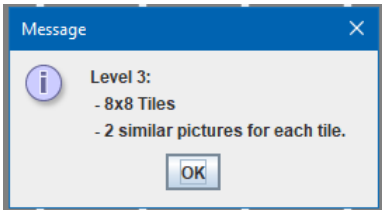


۷) مرحله ۳ بازي: صفحه ۸*۸ و از هر تصوير دو نمونه در صفحه بازي وجود داشته باشد.

این مرحله نیز مانند دو مرحله قبلی با دستور if زیر تنظیم می‌شود:

```
public GameFrame(int level, boolean testMode) {
    gameLevel = level;
    this.isTestMode = testMode;
     if (gameLevel == 3) {
        numberOfPictures = 2;
        numberOfRows = 8;
        numberOfColumns = 8;
    }
```

با انتخاب مرحله ۳، پیغامی به صورت زیر به کاربر نمایش داده می‌شود:



با اجرای بازی و زدن کلید help شکل آن به صورت زیر خواهد بود:

نکته‌ای که در این مرحله و مراحل بعدی وجود دارد، افزوده شدن چند تصویر دیگر به بازی است. تعداد تصاویر در فولدر image قبلا ۱۸ عدد بود ولی برای باز ۸ در ۸ که شامل ۶۴ خانه و از هر تصویر ۲ نمونه وجود دارد باید حداقل ۳۲ فایل تصویر در فولدر image وجود داشته باشد. لذا تصاویر جدیدی مانند آنچه که در فولدر images دیده می‌شود، به بازی اضافه شد و با متد initIcons این تصاویر وارد بازی می‌شود. توضیح سایر موارد و کدها مانند دو مرحله قبل است و فقط اندازه آرایه tiles و icons طبیعتا تغییر خواهد کرد.

۸) مرحله ۴: صفحه ۸*۸ و وجود هشت تصویر که قابلیت تطابق با هر تصویري را داشته باشند.

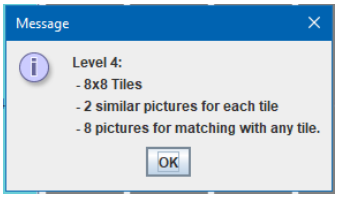
ساختار و کدهای این مرحله شبیه به مرحله ۳ است و تنها تفاوت این است که باید شامل ۸ تصویر باشد که در صورت انتخاب آن، با هر تصویر دیگری قابلیت تطابق داشته باشد. نحوه تنظیم مرحله چهار با if زیر است:

```
public GameFrame(int level, boolean testMode) {
    gameLevel = level;
    this.isTestMode = testMode;
    if (gameLevel == 4) {
        numberOfPictures = 2;
        numberOfRows = 8;
        numberOfColumns = 8;
        numberOfMatchAllImage = 8;
        isMatchAllImageEnabled = true;
    }
```

از یک متغیر boolean با عنوان isMatchAllImageEnabled استفاده می‌شود. در متد initTiles این متغیر بررسی شده تا در صورتی که برابر True باشد ، آنگاه هشت تصویر باید علاوه بر تصاویر قبلی، وارد بازی شود. برای اینکار از دستورات زیر در متد initTiles استفاده می‌شود:

```
if (isMatchAllImageEnabled)
    for (int i = 0; i < numberOfMatchAllImage; i++) {
        tiles[i] = new Tile(GameConstants.MATCH_ALL_IMAGE, GameConstants.LOGO_IMAGE);
        tiles[i].addActionListener(this);
        tiles[i].setDisabledIcon(tiles[i].getImage());
        tiles[i].setBackground(GameConstants.TILES_BACKGROUND);
        if ((i + 1) % 2 == 0) {
            tiles[i].setMatchTile(tiles[i - 1]);
            tiles[i - 1].setMatchTile(tiles[i]);
        }
    }
```

همانطور که دیده می‌شود، با استفاده از یک حلقه for به تعداد numberOfmatchallImage ، تصویری تحت عنوان MATCH_ALL_IMAGE که در فولدر images با نام matchAll.png وجود دارد و به صورت یک ستاره نارنجی رنگ است، به بازی اضافه می‌شود. با انتخاب این مرحله پیغام به صورت زیر به کاربر نمایش داده می‌شود:



در صورت زدن کلید help در این مرحله، تصویر بازی به صورت زیر خواهد بود:

۹) مرحله ۵: قرار دادن دکمه راهنمایی (hint) که با کلیک بر روي آن دو سلول را براي انتخاب به کاربر معرفی کند.

در این مرحله یک کلید hint به پنل footerPanel اضافه می‌شود. برای اینکار تنظیمات if به شکل زیر انجام می‌شود:

```
public GameFrame(int level, boolean testMode) {
    gameLevel = level;
    this.isTestMode = testMode;
    if (gameLevel == 5) {
        numberOfPictures = 2;
        numberOfRows = 8;
        numberOfColumns = 8;
        isHintEnabled = true;
    }
```

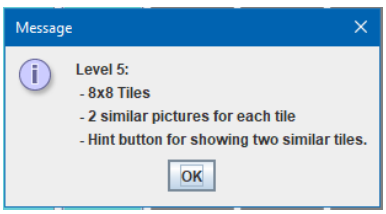
از یک متغیر boolean با عنوان isHintEnabled استفاده می‌شود که با true شدن آن، کلید hint اضافه می‌شود:

```
if (isHintEnabled) {
    hint = new JButton();
    hint.setBackground(GameConstants.BONUS_TEXT_BACKGROUND);
    hint.setFont(GameConstants.TITLE_FONT);
    hint.setForeground(GameConstants.TITLE_FOREGROUND);
    hint.setText("Hint");
    hint.setCursor(GameConstants.HAND_CURSOR);
    hint.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent evt) {
            hintActionPerformed();
        }
    });
    footerPanel.add(hint);
}
```

با زدن این کلید، متد hinActionPerformed اجرا می‌شود که به صورت زیر است:

```
private void hintActionPerformed() {
    if (isHintSelected)
        return;
    boolean showingHint = true;
    while (showingHint) {
        int x = (int) (Math.random() * tiles.length);
        if (tiles[x].isNoImage() == false) {
            tiles[x].setBorder(GameConstants.HINT_BORDER);
            tiles[x].getMatchTile().setBorder(GameConstants.HINT_BORDER);
            showingHint = false;
            isHintSelected = true;
        }
    }
}
```

در صورتی که کلید hint زده شود، متغیر boolean با عنوان isHintSelected برابر True می‌شود تا کاربر نتواند مجددا کلید hint را بزند. پس از زدن کلید hint، با استفاده از یک حلقه while، تا زمانی که دوتا خانه تصادفی انتخاب شود، این حلقه تکرار می‌شود. وقتی دو خانه تصادفی انتخاب شد، آنگاه کادری از با نام HINT_BORDER از نوع LinerBorder با رنگ نارنجی، با استفاده از متد setBorder دور خانه تصادفی و خانه‌ای که تصویر ان با خانه انتخابی مطابقت دارد، نمایش داده می‌شود و کاربر می‌تواند از راهنمایی مربوطه استفاده کند. با ورود به این مرحله پیغام زیر نمایش داده می‌شود.



با باز شدن صفحه این مرحله، یک کلید hint با رنگ نارنجی به پایین صفحه اضافه شده و با زدن کلید آن، دو خانه به صورت تصادفی جهت راهنمایی به کاربر نمایش داده می‌شود. نمونه آن در تصویر زیر آمده است.

لازم به ذکر است که با پایان یافتن بازی و حذف تمام خانه‌ها، آنگاه کلید hint غیر فعال می‌شود.

۱۰) مرحله ۶: اضافه کردن این امکان که با قرار گرفتن ماوس بر روي هر سلول، اعلام نماید که کدام سلول دیگر باید انتخاب شود.

تنظیمات این مرحله با استفاده از if زیر تعیین می‌شود:

```
public GameFrame(int level, boolean testMode) {
    gameLevel = level;
    this.isTestMode = testMode;
    if (gameLevel == 6) {
        numberOfPictures = 2;
        numberOfRows = 8;
        numberOfColumns = 8;
        isMouseOverHintEnabled = true;
    }
```

با استفاده از یک boolean با عنوان isMouseOverHintEnabled، که مقدار آن در این مرحله برابر true می‌شود، این امکان به بازی اضافه می‌شود که با رفتن اشاره‌گر ماوس روی یک خانه، خانه مطابق با آن را نمایش دهد. برای اینکار باید به کلیه خانه‌های جدول یک MouseListener اضافه شود که در کدهای زیر آمده است:

```
if (isMouseOverHintEnabled)
    tiles[i].addMouseListener(new MouseAdapter() {
        @Override
        public void mouseEntered(MouseEvent e) {
            showMouseOverHint(e, true);
        }

        @Override
        public void mouseExited(MouseEvent e) {
            showMouseOverHint(e, false);
        }
    });
```

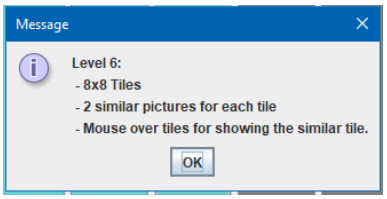
زمانی که این متغیر True باشد، آنگاه یک MouseAdapter به کلیه tileها اضافه می‌شود. با کمک متد mouseEntered در صورتی که اشاره‌گر ماوس وارد محدوده خانه شود، متدی با عنوان showMouseOverHint با مقدار ورودی true فراخوانی می‌شود. در صورتی که اشاره‌گر ماوس از محدوده خانه خارج شود، متد mouseExited فراخوانی شده که درون آن متد showMouseOverHint با ورودی false اجرا می‌شود. ساختار متد در زیر آمده است:

```
private void showMouseOverHint(MouseEvent e, boolean showHint) {
    if (showHint) {
        ((Tile) e.getSource()).setBorder(GameConstants.HINT_BORDER);
        ((Tile) e.getSource()).getMatchTile().setBorder(GameConstants.HINT_BORDER);
    } else {
        ((Tile) e.getSource()).setBorder(null);
        ((Tile) e.getSource()).getMatchTile().setBorder(null);
    }

}
```

با فراخوانی این متد، مقدار ورودی showHint بررسی می‌شود. اگر این مقدار true باشد یعنی اشاره‌گر ماوس وارد خانه شده است و باید عملیات نمایش hint یا راهنمایی انجام گیرد. برای اینکار، باید خانه‌ای که اشاره‌گر ماوس روی آن قرار دارد را دریافت کنیم و سپس برای آن یک border از نوع HINT_BORDER تنظیم کنیم. برای اینکار از متد e.getSource() استفاده می‌شود که خروجی آن از نوع Object است و سپس این Object را به کلاس Tile تبدیل یا cast می‌کنیم. سپس برای این tile متد setBorder را هم برای خود خانه انتخابی و هم برای خانه مطابق با آن پیاده‌سازی می‌کنیم. نتیجه این کار این است که کاربر به محض حرکت ماوس روی یک خانه، خانه مطابق با آن نمایش داده می‌شود.

در هنگام خروج اشاره‌گر ماوس از محدوده خانه، باید کادر نارنجی رنگ اضافه شده، به طریقی برداشته شود. برای اینکار از دستور setBorder(null) استفاده می‌شود تا کادر مربوطه با مقدار null تعیین شود و به معنای حذف کادر است.

با انتخاب کلید مرحله ۶ آنگاه پیغامی به صورت زیر به کاربر نمایش داده می‌شود:



با باز شدن پنجره بازی، با حرکت ماوس، خانه‌هایی که با هم مطابقت دارند، با کادر نارنجی رنگ مشخص می‌شوند.

۱۱) مرحله ۷: اضافه کردن تایمر و شمارشگر امتیاز ویژه در صفحه بازي (امتیاز ویژه: با انتخاب و انطباق بعضی از تصاویر ۵ امتیاز ویژه به کاربر اختصاص یابد. حداقل سه مورد از این گونه تصاویر باشد). به ازاء هر ۳۰۰ امتیاز عادي نیز ۵ امتیاز ویژه به کاربر تعلق بگیرد و به ازاء هر -۱۰۰ امتیاز عادي ۳ امتیاز ویژه از کاربر کسر شود). تبدیل امتیاز هاي عادي به ویژه در ضریب هاي ۳۰۰ و - ۱۰۰ انجام می شود.

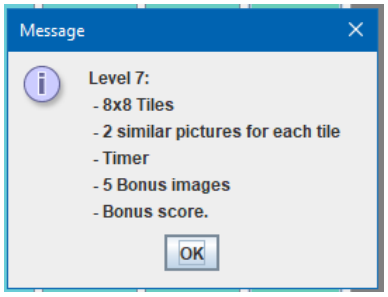
همانطور که در صورت مسئله گفته شده است، باید یک تایمر به بازی اضافه شود. علاوه بر آن باید حداقل ۳ تصویر با امتیاز ویژه نیز به بازی اضافه گردد. با انتخاب این تصاویر ۵ امتیاز ویژه با کاربر داده می‌شود. برای همین منظور، تصویر با عدد ۵ ساخته شده و با نام bonus.png در فولدر images قرار داده شده است تا بتوان آن را از طریق متد initTilers وارد بازی کرد. برای اینکار تنظیمات زیر در دستور if انجام می‌شود:

```
public GameFrame(int level, boolean testMode) {
    gameLevel = level;
    this.isTestMode = testMode;
    if (gameLevel == 7) {
        numberOfPictures = 2;
        numberOfRows = 8;
        numberOfColumns = 8;
        isTimerEnabled = true;
        isBonusImageEnaled = true;
    }
```

از یک متغیر Boolean با عنوان isTimerEnabled برای فعال سازی تایمر و از یک متغیر Boolean با عنوان isBonusImageEnabled برای اضافه کردن تصاویر ویژه استفاده می‌شود. برای فعال کردن تایمر، از یک آبجکت از کلاس Timer جاوا استفاده می‌شود، که هر یک ثانیه یک بار اجرا شده و مقدار زمان را در یک کادر متنی در پایین صفحه نمایش می‌دهد:

```
if (isTimerEnabled) {
    timer = new Timer(1000, new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            String s = "" + (++seconds);
            String m = "" + minutes;
            if (s.length() == 1)
                s = "0" + s;
            if (m.length() == 1)
                m = "0" + m;
            timerText.setText("Time: " + m + ":" + s);
            if (seconds >= 59) {
                seconds = -1;
                minutes++;
            }
        }
    });
    timer.start();
}
```

آبجکت کلاس Timer یک ورودی از نوع میلی ثانیه (۱۰۰۰ میلی ثانیه برابر ۱ ثانیه است) دارد و یکی ورودی از جنس ActionListener دارد که د واقع عملیاتی را نشان می‌دهد که قرار است با هر بار اجرا شردن تایمر، این عملیات نیز اجرا می‌شود. در این جا از یک رشته s برای نمایش ثانیه، از یک رشته m برای نمایش دقیقه استفاده شده است. اگر ثانیه برابر با ۶۰ شد آنگاه یک واحد به دقیقه اضافه می‌شود. سپس این مقدار ثانیه و دقیقه با استفاده از متد setText در داخل کادر متنی با عنوان timerText در پایین صفحه نمایش داده می‌شود. در صورتی که متغیر isTimerEnabled برابر با true باشد آنگاه با دستور timer.start ، تایمری که ساخته شده است را اجرا می‌کنیم تا زمان را در کادر پایین صفحه مانند تصویر زیر نمایش دهد. قبل از آن با انتخاب کلید مرحله هفت، پیغام زیر به کاربر نمایش داده می‌شود:



متن زمان سنج به کادر پایین مانند تصویر زیر اضافه شده است. همچنین علاوه بر کادر زمان، کادر متنی برای نمایش امتیاز ویژه به قسمت پایین صفحه اضافه شده است:

برای اضافه کردن حداقل سه مورد و حداکثر ۱۰ مورد از تصاویر با امتیاز ویژه، در متد initTile به صورت زیر عمل می‌کنیم:

```
if (isBonusImageEnaled) {
    int numberOfBonusImages = 3 + (int) (Math.random() * (GameConstants.MAXIMUM_NUMBER_OF_BONUS_IMAGES - 2));
    if (numberOfBonusImages % 2 != 0)
        numberOfBonusImages++;
    for (int i = 0; i < numberOfBonusImages; i++) {
        tiles[i] = new Tile(GameConstants.BONUS_IMAGE, GameConstants.LOGO_IMAGE);
        tiles[i].addActionListener(this);
        tiles[i].setDisabledIcon(tiles[i].getImage());
        tiles[i].setBackground(GameConstants.TILES_BACKGROUND);
        if ((i + 1) % 2 == 0) {
            tiles[i].setMatchTile(tiles[i - 1]);
            tiles[i - 1].setMatchTile(tiles[i]);
        }
    }
}
```

حداقل تعداد تصاویر ویژه ر متغیری با عنون MAXIMUM_NUMBER_OF_BONUS_IMAGES ذخیره شده است که برابر ۱۰ است. از این مقدار دو واحد کم می شود تا برابر ۸ گردد. حال با کمک Math.random() یک عدد تصادفی تولید شده که در ۸ ضرب می‌کنیم. با اینکار عدد تصادفی در بازه ۰ تا ۷ خواهد بود. حال عدد ۳ که حداقل تعداد تصاویر است را به این بازه اضافه می‌کنیم و در متغیری تحت عنوان numberOfBonusImages قرار می‌دهیم. پس با اینکار این متغیر مقداری بین ۳ تا ۱۰ خواهد داشت و چون باید تعداد این تصاویر زوج باشد، در نتیجه اگر عدد

تولید شده فرد بود، یک واحد آن را افزایش می‌دهیم. حال با استفاده از یک حلقه for، تصویر ویژه با نام BONUS_IMAGE را ساخته و در مکان ۰ تا تعداد تصاویر ویژه، قرار می‌دهیم. سپس مانند سایر مراحل، متد shuffle را فراخوانی می‌کنیم تا ترتیب تصاویر تغییر کند.

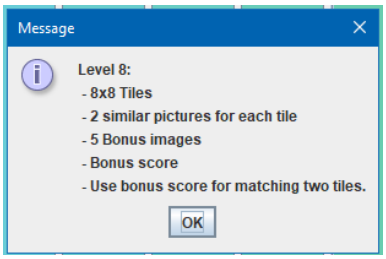
با زدن کلید help می‌توان تصاویر امتیاز ویژه را دید که حداقل سه مورد و حداکثر ۱۰ مورد از آن ها با هر بار اجرای این مرحله در بازی اضافه می‌شود:

برای تبدیل امتیاز ها در ضرایب ۳۰۰ و ۱۰۰- در متد check به صورت زیر عمل می‌کنیم:

```
bonusValue += 100;
if (isBonusImageEnaled) {
    bonusScore += (bonusValue / 300) * GameConstants.BONUS_SCORE;
    if (firstSelection.getImage() == GameConstants.BONUS_IMAGE)
        bonusScore += GameConstants.BONUS_SCORE;
    bonusText.setText("Bonus: " + bonusScore);
    bonusValue %= 300;
}
```

۱۲) مرحله ۸: امکان استفاده از امتیاز ویژه فراهم شود. بدین صورت که به ازاء هر ۱۰ امتیاز ویژه دو سلول که تصاویر مشابه هم هستند به صورت اتوماتیک انتخاب شوند.

با انتخاب کلید مرحله ۸، پیغام زیر به کاربر نشان داده می‌شود:



برای اینکه کابر بتواند از امتیاز ویژه خود استفاده کند، یه کلید به صورت زیر به پایین صفحه اضافه می‌شود:

```
if (useBonusScoreEnabled) {
    useBonus = new JButton();
    useBonus.setBackground(GameConstants.BONUS_BUTTON_BACKGROUND);
    useBonus.setFont(GameConstants.TITLE_FONT);
    useBonus.setForeground(GameConstants.TITLE_FOREGROUND);
    useBonus.setText("Use Bonus");
    useBonus.setCursor(GameConstants.HAND_CURSOR);
    useBonus.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent evt) {
            useBonusActionPerformed();
        }
    });
    footerPanel.add(useBonus);
}
```

این کلید با عنوان Use Bonus و با رنگ سبز به پایین صفحه به صورت زیر اضافه می‌شود:

با زدن کلیک بر روی این کلید، متد actionPerformed مربوطه اجرا می‌شود که در داخل ان متدی با عنوان useBonusActionPerformed فراخوانی می‌شود. کد این متد به صورت زیر است:

```
private void useBonusActionPerformed() {
    if (useBonusScoreEnabled) {
        if (bonusScore < 10)
            return;
        boolean matching = true;
        while (matching) {
            int x = (int) (Math.random() * tiles.length);
            if (tiles[x].isNoImage() == false) {
                firstSelection = tiles[x];
                secondSelection = tiles[x].getMatchTile();
                matching = false;
                check();
            }
        } // while
        bonusScore -= 10;
        bonusText.setText("Bonus: " + bonusScore);
    } // if
}
```

اگر مقدار bonusScore کمتر از ۱۰ باشد، آنگاه زدن این کلید نباید راهنمایی انجام دهد. در صورتی که امتیاز بیشتر از ده بود، آنگاه با کمک یک حلقه while، به صورت تصادفی، دوتا خانه انتخاب شده، و نمایان خواهند شد. سپس ۱۰ امتیاز از کاربر کسر می‌گردد.

تنظیمات این مرحله به صورت دستور if زیر در داخل سازنده کلاس GameFrame انجام می‌شود:

```
public GameFrame(int level, boolean testMode) {
    gameLevel = level;
    this.isTestMode = testMode;
    if (gameLevel == 8) {
        numberOfPictures = 2;
        numberOfRows = 8;
        numberOfColumns = 8;
        isBonusImageEnaled = true;
        useBonusScoreEnabled = true;
    }
```

همانطور که دیده می‌شود، در این مرحله تصاویر با امتیاز ویژه از طریق متغیر Boolean با عنوان isBonusImageEnabled مانند مرحله قبلی، به بازی اضافه می‌شوند. با استفاده از متغیر Boolean با عنوان useBonusScoreEnabled، کلید use bonus به پایین صفحه اضافه می‌شود تا کاربر بتواند از امتیاز های ویژه خود استفاده کند.

۱۳) مرحله ۹: با انتخاب یک سلول و کسر ۲ امتیاز ویژه سلول متناظر با سلول انتخاب شده به صورت اتوماتیک انتخاب شود. کاربر در استفاده از این امکان اختیار داشته باشد.

در این مرحله مانند مرحله قبل انجام می‌شود با این تفاوت که می‌توان با کسر ۲ امتیاز ویژه، سلول متناظر با یک سلول منتخب را نمایش داد. برای اینکار تنظیمات if به صورت زیر است:

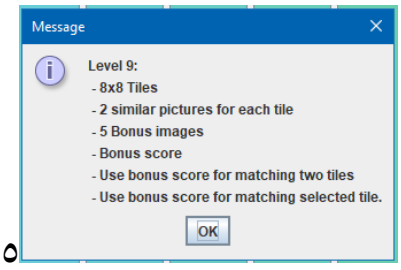
```
public GameFrame(int level, boolean testMode) {
    gameLevel = level;
    this.isTestMode = testMode;
    if (gameLevel == 9) {
        numberOfPictures = 2;
        numberOfRows = 8;
        numberOfColumns = 8;
        isBonusImageEnaled = true;
        useBonusScoreEnabled = true;
        useBonusScoreForTwoTiles = true;
    }
```

همانطور که دیده می‌شود، متغیرها مانند مرحله قبل است اما متغیری با عنوان useBonusScoreForTwoTiles از نوع Boolean اضافه شده است. این متغیر در متد useBonusActionPerformed به صورت زیر بررسی می‌شود:

```
private void useBonusActionPerformed() {
    if (useBonusScoreEnabled) {
        if (useBonusScoreForTwoTiles) {
            if (bonusScore < 2)
                return;
            if (firstSelection != null && firstSelection.isHidden() == false) {
                secondSelection = firstSelection.getMatchTile();
                bonusScore -= 2;
                check();
                return;
            }
        }
```

در صورتی که این متغیر true باشد آنگاه bonusScore بررسی میشود. اگر این مقار کوچکتر از ۲ باشد، آنگاه کلید use bonos هیچ کاری انجام نخواهد داد. در غیر این صورت، خانه ای که با عنوان firstSeletion انتخاب اول شده است، خانه مطابق با آن را نمایش داده و ۲ امتیاز ویژه از کاربر کسر می‌گردد.

با انتخاب کلید مرحله ۹، پیغام زیر به کابر نمایش داده می‌شود:



به طور خلاصه در این مرحله، در صورتی که امتیاز ویژه بیشتر از ۲ باشد، کاربر می‌تواند، یک سلول را انتخاب کند و با زدن کلید use bonus ، خانه مطابق سلول انتخابی را مشاهده نماید. اگر امتیاز بیشتر از ۱۰ باشد، آنگاه با زدن کلید use bonus به صورت اتوماتیک دو خانه نمایش داده می‌شود.

دانشجو می تواند به سلیقه خود آیتم هاي ديگري (اختياري) به بازي اضافه نماید.

مواردی که اضافه شده است شامل زمان سنج برای نمایش help بازی است. همچنین رنگ و تصاویر بازی تغییر داده شده است. نهایتا صداهایی که موقعه انتخاب درست، انتخاب غلط، برد در بازی و باخت در بازی پخش می‌شوند، تغییر داده شده‌اند تا شکل و عملکرد بازی شکیل تر و زیبا تر باشد.