GamrCorps / **GameOverWindow.java** Secret

Created a minute ago

APCSP Code

GameOverWindow.java

```java
package apcsp.create.mastermind.window;

import apcsp.create.mastermind.util.MastermindGame;
import apcsp.create.mastermind.util.Util;

import javax.swing.*;
import java.awt.*;

class GameOverWindow extends JFrame {

    GameOverWindow(MastermindWindow parentWindow, MastermindGame game) {
        super();
        this.setTitle("Game Over");
        this.setSize(400, 300);
        this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

        Box container = Box.createVerticalBox();
        Util.setSize(container, 400, 300);

        JLabel youWinLabel = new JLabel("You Win!", SwingConstants.CENTER);
        youWinLabel.setFont(new Font(youWinLabel.getFont().getName(), Font.PLAIN, 60));
        youWinLabel.setAlignmentX(CENTER_ALIGNMENT);
        Util.setSize(youWinLabel, 400, 150);
        container.add(youWinLabel);

        JLabel moveCountLabel = new JLabel("You took " + game.moveCount + " moves to break the code!", SwingConstants.CENTER);
        moveCountLabel.setFont(new Font(moveCountLabel.getFont().getName(), Font.PLAIN, 20));
        moveCountLabel.setAlignmentX(CENTER_ALIGNMENT);
        Util.setSize(moveCountLabel, 400, 100);
        container.add(moveCountLabel);

        Box buttonBox = Box.createHorizontalBox();
        Util.setSize(buttonBox, 400, 50);

        JButton quitGameButton = new JButton("Quit Game");
        Util.setSize(quitGameButton, 200, 50);
        quitGameButton.addActionListener(e -> System.exit(0));
        buttonBox.add(quitGameButton);

        JButton newGameButton = new JButton("New Game");
        Util.setSize(newGameButton, 200, 50);
        newGameButton.addActionListener(e -> {
            parentWindow.newGame();
            this.dispose();
        });
        buttonBox.add(newGameButton);

        container.add(buttonBox);

        this.add(container);

        this.revalidate();
        this.repaint();

        this.pack();

        this.setVisible(true);

        this.setLocationRelativeTo(null);
    }

    public static void main(String[] args) {
        new GameOverWindow(new MastermindWindow(), new MastermindGame());
    }
}
```

Main.java

```java
package apcsp.create.mastermind;

import apcsp.create.mastermind.window.MastermindWindow;

import javax.swing.*;

public class Main {

    public static void main(String[] args) {
        // Make sure to set a look and feel that works on all platforms.
        try {
            UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName());
        } catch (Exception e) {
            e.printStackTrace();
        }

        // Create the main Mastermind window.
        new MastermindWindow();
    }
}
```

MastermindGame.java

```java
package apcsp.create.mastermind.util;

import java.util.Random;

public class MastermindGame {

    private PegColor[] solution;
    public Move lastMove;
    public int moveCount;

    private static Random rand = new Random();

    public MastermindGame() {
        solution = getRandomSolution();
        System.out.println();
    }

    private PegColor[] getRandomSolution() {
        PegColor[] sol = new PegColor[4];
        PegColor[] values = PegColor.values();
        for (int i = 0; i < 4; i++) {
            sol[i] = values[rand.nextInt(values.length)];
        }
        return sol;
    }

    public PegColor[] submitMove(Move move) {
        Move newMove = calculateResponsePegs(move);
        lastMove = newMove;
        moveCount++;
        assert newMove != null;
        return newMove.responsePegs;
    }


    private Move calculateResponsePegs(Move move) {
        if (move.responsePegs == null) {
            int num_correct = 0;
            int num_almost_correct = 0;
            boolean[] partOfSolution = new boolean[4];
            boolean[] checked = new boolean[4];
            for (int i = 0; i < 4; i++) {
                if (move.pegs[i] == solution[i]) {
                    num_correct++;
                    partOfSolution[i] = true;
                    checked[i] = true;
                }
            }
            for (int i = 0; i < 4; i++) {
                for (int j = 0; j < 4; j++) {
                    if (j != i && !checked[i] && !partOfSolution[j]) {
                        if (move.pegs[i] == solution[j]) {
                            partOfSolution[j] = true;
                            checked[i] = true;
                            num_almost_correct++;
                            break;
                        }
                    }
                }
            }
            PegColor[] result = new PegColor[4];
            int index = 0;
            while (num_correct > 0) {
                result[index++] = PegColor.RED;
                num_correct--;
            }
            while (num_almost_correct > 0) {
                result[index++] = PegColor.BLACK;
                num_almost_correct--;
            }
            move.responsePegs = result;
            return move;
        }
        return null;
    }

    public MastermindGame restart() {
        MastermindGame game = new MastermindGame();
        game.solution = this.solution;
        return game;
    }

    public boolean wasWon() {
        for (int i = 0; i < 4; i++) {
            if (lastMove.pegs[i] != solution[i]) return false;
        }
        return true;
    }
}
```

MastermindWindow.java

```java
package apcsp.create.mastermind.window;

import apcsp.create.mastermind.util.*;

import javax.swing.*;
import java.awt.*;

public class MastermindWindow extends JFrame {

    private PegColor currentColor = PegColor.RED;
    private PegColor[] moveColors;
    private PegColor[] responseColors;

    private Box historyPanel;
    private JScrollPane historyScrollPane;
    private JButton[] moveButtons;
    private JButton[] responseButtons;
    private JButton submitButton;
    private int submitButtonState = 0;

    private MastermindGame game;
    private boolean moveButtonsEnabled = true;
    private GameOverWindow gameOverWindow;

    public MastermindWindow() {
        super();
        this.setTitle("Mastermind");
        this.setSize(620, 500);
        this.setJMenuBar(this.createMenuBar());
        Box container = Box.createVerticalBox();


        Box topPanel = Box.createHorizontalBox();

        this.historyPanel = Box.createVerticalBox();
        this.historyPanel.setPreferredSize(new Dimension(520, 0));

        historyScrollPane = new JScrollPane(this.historyPanel, JScrollPane.VERTICAL_SCROLLBAR_ALWAYS, JScrollPane.HORIZONTAL_SCROLLBAR_NEVE
        historyScrollPane.setPreferredSize(new Dimension(520, 400));
        topPanel.add(historyScrollPane);

        Box colorPanel = Box.createVerticalBox();
        colorPanel.setPreferredSize(new Dimension(100, 400));

        JLabel label = new JLabel("Colors", SwingConstants.CENTER);
        label.setMaximumSize(new Dimension(100, 50));
        colorPanel.add(label);

        for (int i = 0; i < PegColor.values().length; i++) {
            JButton button = new JButton();
            button.setBackground(PegColor.values()[i].color);
            button.setMaximumSize(new Dimension(100, 50));
            final int j = i;
            button.addActionListener(e -> this.currentColor = PegColor.values()[j]);
            colorPanel.add(button);
        }
        topPanel.add(colorPanel);
        container.add(topPanel);


        Box bottomPanel = Box.createHorizontalBox();

        this.moveButtons = new JButton[4];
        for (int i = 0; i < 4; i++) {
            JButton button = new JButton(new PegIcon(PegIcon.LARGE, Color.WHITE));
            Util.setSize(button, 100, 100);
            final int j = i;
            button.addActionListener(e -> {
                this.moveColors[j] = this.currentColor;
                updateMoveButtons();
            });
            bottomPanel.add(button);
            this.moveButtons[i] = button;
        }

        JPanel responsePegPanel = new JPanel();
        responsePegPanel.setLayout(new GridLayout(2, 2));
        Util.setSize(responsePegPanel, 100, 100);
        this.responseButtons = new JButton[4];
        for (int i = 0; i < 4; i++) {
            JButton button = new JButton(new PegIcon(PegIcon.SMALL, Color.WHITE));
            Util.setSize(button, 50, 50);
            button.setEnabled(false);
            responsePegPanel.add(button);
            this.responseButtons[i] = button;
        }
        bottomPanel.add(responsePegPanel);

        bottomPanel.add(Box.createHorizontalStrut(10));

        this.submitButton = new JButton("Submit");
        Util.setSize(this.submitButton, 100, 50);
        this.submitButton.setMargin(new Insets(2, 10, 2, 10));
        this.submitButton.addActionListener(e -> {
            switch (this.submitButtonState) {
                case 0:
                    Move move = new Move(this.moveColors);
                    this.responseColors = this.game.submitMove(move);
                    this.moveButtonsEnabled = false;
                    updateMoveButtons();
                    if (this.game.wasWon()) {
                        gameOverWindow = new GameOverWindow(this, game);
                        this.submitButton.setText("New Game");
                        this.submitButtonState = 2;
                    } else {
                        this.submitButton.setText("Next Guess");
                        this.submitButtonState = 1;
                    }
                    break;
                case 1:
                    this.submitButton.setText("Submit");
                    this.responseColors = null;
                    this.moveButtonsEnabled = true;
                    updateMoveButtons();
                    this.submitButtonState = 0;
                    Box box = createPreviousMoveBox(this.game.lastMove);
                    addToHistoryPanel(box);
                    break;
                case 2:
                    if (gameOverWindow != null) gameOverWindow.dispose();
                    newGame();
                    break;
            }
        });
        bottomPanel.add(this.submitButton);

        container.add(bottomPanel);


        this.add(container);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setVisible(true);

        this.moveColors = new PegColor[]{PegColor.RED, PegColor.BLUE, PegColor.GREEN, PegColor.YELLOW};
        this.game = new MastermindGame();
        this.responseColors = null;
        updateMoveButtons();

        this.setLocationRelativeTo(null);
    }

    private JMenuBar createMenuBar() {
        JMenuBar menuBar = new JMenuBar();

        JMenu gameMenu = new JMenu("Game");

        JMenuItem newGame = new JMenuItem("New Game");
        newGame.addActionListener(e -> {
            resetHistoryPanel();
            this.game = new MastermindGame();
        });
        gameMenu.add(newGame);

        JMenuItem restartGame = new JMenuItem("Restart Game");
        restartGame.addActionListener(e -> {
            resetHistoryPanel();
            this.game = this.game.restart();
        });
        gameMenu.add(restartGame);

        JMenuItem endGame = new JMenuItem("Quit Game");
        endGame.addActionListener(e -> System.exit(0));
        gameMenu.add(endGame);

        menuBar.add(gameMenu);


        JMenu helpMenu = new JMenu("Help");

        JMenuItem usage = new JMenuItem("How to Use");
        usage.addActionListener(e -> new UsageWindow());
        helpMenu.add(usage);

        JMenuItem rules = new JMenuItem("Rules");
        rules.addActionListener(e -> new RulesWindow());
        helpMenu.add(rules);

        menuBar.add(helpMenu);

        return menuBar;
    }

    private static Box createPreviousMoveBox(Move move) {
        Box panel = Box.createHorizontalBox();

        for (PegColor peg : move.pegs) {
            JButton button = new JButton(peg.icon_large);
            Util.setSize(button, 100, 100);
            button.setEnabled(false);
            panel.add(button);
        }

        JPanel responsePegPanel = new JPanel();
        responsePegPanel.setLayout(new GridLayout(2, 2));
        Util.setSize(responsePegPanel, 100, 100);

        for (PegColor responsePeg : move.responsePegs) {
            JButton button;
            if (responsePeg == null) {
                button = new JButton(PegIcon.WHITE_ICON_SMALL);
            } else {
                button = new JButton(responsePeg.icon_small);
            }
            Util.setSize(button, 50, 50);
            button.setEnabled(false);
            responsePegPanel.add(button);
        }

        panel.add(responsePegPanel);
        Util.setSize(panel, 500, 100);

        return panel;
    }

    private void addToHistoryPanel(JComponent component) {
        component.setAlignmentX(LEFT_ALIGNMENT);
        this.historyPanel.setPreferredSize(new Dimension(520, this.historyPanel.getPreferredSize().height + 100));
        this.historyPanel.add(component);
        this.revalidate();
        this.repaint();
        JScrollBar bar = historyScrollPane.getVerticalScrollBar();
        bar.setValue(bar.getMaximum());
    }

    private void resetHistoryPanel() {
        this.historyPanel.removeAll();
        this.historyPanel.setPreferredSize(new Dimension(520, 0));
        this.revalidate();
        this.repaint();
    }

    private void updateMoveButtons() {
        for (int i = 0; i < 4; i++) {
            this.moveButtons[i].setIcon(this.moveColors[i].icon_large);
            this.moveButtons[i].setEnabled(this.moveButtonsEnabled);
        }
        if (this.responseColors != null) {
            for (int i = 0; i < 4; i++) {
                if (this.responseColors[i] == null) continue;
                this.responseButtons[i].setIcon(this.responseColors[i].icon_small);
            }
        } else {
            for (int i = 0; i < 4; i++) {
                this.responseButtons[i].setIcon(PegIcon.WHITE_ICON_SMALL);
            }
        }
    }

    void newGame() {
        resetHistoryPanel();
        this.game = new MastermindGame();
        this.submitButton.setText("Submit");
        this.responseColors = null;
        this.moveButtonsEnabled = true;
        updateMoveButtons();
        this.submitButtonState = 0;
    }
}
```

Move.java

```java
package apcsp.create.mastermind.util;

public class Move {
    public PegColor[] pegs;
    public PegColor[] responsePegs;

    public Move (PegColor[] pegs) {
        this(pegs, null);
    }

    private Move (PegColor[] pegs, PegColor[] responsePegs) {
        this.pegs = pegs;
        this.responsePegs = responsePegs;
    }
}
```

PegColor.java

```java
package apcsp.create.mastermind.util;

import java.awt.*;

public enum PegColor {
    RED(Color.RED),
    BLUE(Color.BLUE),
    GREEN(Color.GREEN),
    YELLOW(Color.YELLOW),
    ORANGE(Color.ORANGE),
    BLACK(Color.BLACK);

    public Color color;
    public PegIcon icon_small;
    public PegIcon icon_large;

    PegColor(Color color) {
        this.color = color;
        this.icon_small = new PegIcon(PegIcon.SMALL, color);
        this.icon_large = new PegIcon(PegIcon.LARGE, color);
    }
}
```

PegIcon.java

```java
package apcsp.create.mastermind.util;

import javax.swing.*;
import java.awt.*;
import java.awt.geom.Ellipse2D;

public class PegIcon implements Icon {

    public static final int SMALL = 32;
    public static final int LARGE = 64;
    public static final PegIcon WHITE_ICON_SMALL = new PegIcon(SMALL, Color.WHITE);
    @SuppressWarnings("UnusedDeclaration")
    public static final PegIcon WHITE_ICON_LARGE = new PegIcon(LARGE, Color.WHITE);

    private int size;
    private Color color;

    public PegIcon(int size, Color color) {
        this.size = size;
        this.color = color;
    }

    @Override
    public void paintIcon(Component c, Graphics g, int x, int y) {
        Graphics2D canvas = (Graphics2D) g.create();

        canvas.setColor(color);
        canvas.fill(new Ellipse2D.Double(x, y, this.size, this.size));

        canvas.dispose();
    }

    @Override
    public int getIconWidth() {
        return this.size;
    }

    @Override
    public int getIconHeight() {
        return this.size;
    }
}
```

RulesWindow.java

```java
package apcsp.create.mastermind.window;

import apcsp.create.mastermind.util.Util;

import javax.swing.*;

class RulesWindow extends JFrame {

    RulesWindow() {
        super();
        this.setTitle("Mastermind Rules");
        this.setSize(400, 400);
        this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

        JTextArea textArea = new JTextArea();
        Util.setSize(textArea, 600, 200);
        textArea.setEditable(false);
        textArea.setLineWrap(true);
        textArea.setWrapStyleWord(true);
        textArea.setText("Welcome to the game of Mastermind!\n" +
                "\n" +
                " - The goal of the game is to break the code in as few guesses as possible.\n" +
                " - Use the four large pegs to choose a code to submit.\n" +
                " - Once you submit a code, you are given a response of four RED, WHITE, and BLACK pegs. These pegs may be in an arbitrary
                " - RED response pegs indicate that one of your pegs is the right color and in the right location.\n" +
                " - BLACK response pegs indicate that one of your pegs is the right color, but in the wrong location.\n" +
                " - WHITE response pegs indicate that one of your pegs is the wrong color.");

        this.add(textArea);

        this.pack();

        this.setVisible(true);

        this.setLocationRelativeTo(null);
    }
}
```

UsageWindow.java

```java
package apcsp.create.mastermind.window;

import apcsp.create.mastermind.util.Util;

import javax.swing.*;

class UsageWindow extends JFrame {

    UsageWindow() {
        super();
        this.setTitle("How to Use");
        this.setSize(400, 400);
        this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

        JTextArea textArea = new JTextArea();
        Util.setSize(textArea, 600, 200);
        textArea.setEditable(false);
        textArea.setLineWrap(true);
        textArea.setWrapStyleWord(true);
        textArea.setText("How to Use\n" +
                "\n" +
                " - The four large buttons on the bottom allow you to enter your guess. Click on them to change the color of that peg to th
                " - To change the currently selected color, click one of the colored buttons on the right.\n" +
                " - When you are ready to submit your guess, press the submit button. This will lock in your guess and show you the respons
                " - Then, press \"Next Guess\" to try and guess again. This will put your previous guess into the history panel above the g
                " - You can scroll through the history panel to see your previous guesses.");

        this.add(textArea);

        this.pack();

        this.setVisible(true);

        this.setLocationRelativeTo(null);
    }
}
```

Util.java

```java
package apcsp.create.mastermind.util;

import javax.swing.*;
import java.awt.*;

public class Util {
    @SuppressWarnings("UnusedReturnValue")
    public static <T extends JComponent> T setSize(T comp, int width, int height) {
        Dimension dim = new Dimension(width, height);
        comp.setMinimumSize(dim);
        comp.setPreferredSize(dim);
        comp.setMaximumSize(dim);
        return comp;
    }
}
```