160 Project

February 29, 2024

Anonymous

Please take a look at my project description and understand it: Project Title: Simple Music Notation Editor Project Overview: Develop a Simple Music Notation Editor aimed at music students and educators for creating and editing basic musical scores. This Java Swing application will extend the provided simple paint application framework to incorporate musical notation functionalities, offering a straightforward interface for users to input, visualize, and edit notes on a staff. Simplified Objectives: Create a user-friendly interface for displaying and editing musical notes on a staff. Allow users to place, move, and remove notes with mouse interactions. Enable basic playback of the composed music using synthesized instrument sounds. Specific Features to Implement: Musical Staff Display: A panel that graphically represents a musical staff, where users can visually place notes. Note Placement and Editing: Users can click on the staff to add notes. Clicking on an existing note will remove it or allow the user to drag it to a new position for pitch change. Playback Functionality: A simple play button that interprets the notes on the staff and plays back the melody using a basic synthesized sound. Note Duration Selection: A toolbar or menu allowing users to select different note durations (e.g., whole, half, quarter notes) before placing them on the staff.

ChatGPT

Got it! Your project is to develop a Simple Music Notation Editor using Java Swing. The application will allow music students and educators to create and edit basic musical scores. It will extend a provided simple paint application framework to include musical notation functionalities. Here's a breakdown of the objectives and features:

1. **User Interface:**
   - Create an intuitive interface for displaying musical notes on a staff.
2. **Note Placement and Editing:**
   - Enable users to add, move, and remove notes on the staff using mouse interactions.
   - Clicking on an existing note will either remove it or allow for dragging to change its pitch.
3. **Playback Functionality:**
   - Implement a play button to play back the composed music using synthesized instrument sounds.
4. **Note Duration Selection:**
   - Provide a toolbar or menu for users to select different note durations (e.g., whole, half, quarter notes) before placing them on the staff.

To achieve this, you'll need to integrate graphical representation of a musical staff, implement note placement mechanisms, incorporate playback functionality, and allow users to select note durations. The end result should be a user-friendly application that simplifies the process of composing and editing basic musical scores.

Anonymous

Redo this so that it makes a pair of staffs, one for the Treble and one for the Bass. Make it so that there is five lines per staff private class StaffPanel extends JPanel { int startX = 0; int startY = 50; int endX = getWidth() - 50; int endY = 50; @Override protected void paintComponent(Graphics g) { super.paintComponent(g); startX += 50; for (int i = 0; i < 6; i++) { g.drawLine(startX, startY + i * 20, endX, endY + i * 20); } } @Override public Dimension getPreferredSize() { return new Dimension(600, 100); // Example size for the staff panel } }

ChatGPT

Certainly! Below is the updated code for the `StaffPanel` class that generates a pair of staffs, one for the Treble clef and one for the Bass clef, each with five lines:

java Copy code

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

private class StaffPanel extends JPanel {
    // Constants for staff dimensions and positions
    private static final int LINE_GAP = 20; // Vertical gap between staff lines
    private static final int NUM_LINES = 5; // Number of lines per staff
    private static final int STAFF_WIDTH = 500; // Width of each staff
    private static final int STAFF_HEIGHT = NUM_LINES * LINE_GAP; // Height of each staff
    private static final int STAFF_MARGIN = 20; // Margin around each staff
    private static final int PANEL_WIDTH = 2 * (STAFF_MARGIN + STAFF_WIDTH); // Total panel width
    private static final int PANEL_HEIGHT = 2 * (STAFF_MARGIN + STAFF_HEIGHT); // Total panel height

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        // Draw Treble clef staff
        drawStaff(g, STAFF_MARGIN, STAFF_MARGIN, STAFF_WIDTH, STAFF_HEIGHT);

        // Draw Bass clef staff
        drawStaff(g, STAFF_MARGIN, 2 * STAFF_MARGIN + STAFF_HEIGHT, STAFF_WIDTH, STAFF_HEIGHT);
    }

    // Method to draw a single staff
```

```
    private void drawStaff(Graphics g, int x, int y, int width, int height) {
        int lineY = y; // Starting y-coordinate for the lines

        // Draw the staff lines
        for (int i = 0; i < NUM_LINES; i++) {
            g.drawLine(x, lineY, x + width, lineY);
            lineY += LINE_GAP; // Move to the next line position
        }
    }

    @Override
    public Dimension getPreferredSize() {
        return new Dimension(PANEL_WIDTH, PANEL_HEIGHT);
    }
}
```

This code creates a `StaffPanel` class that draws two staffs, one above the other, representing the Treble clef and the Bass clef, respectively. Each staff consists of five horizontal lines, with appropriate spacing between them. The dimensions and positions of the staffs are calculated based on predefined constants. Finally, the `paintComponent()` method draws the staffs on the panel, and `getPreferredSize()` method sets the preferred size of the panel to accommodate both staffs.

Anonymous

Can you add a gap at the end after drawing the Bass clef staff so that there is space after every pair of staffs? protected void paintComponent(Graphics g) { super.paintComponent(g); // Draw Treble clef staff drawStaff(g, STAFF_MARGIN, STAFF_MARGIN, STAFF_WIDTH, STAFF_HEIGHT); // Draw Bass clef staff drawStaff(g, STAFF_MARGIN, 2 * STAFF_MARGIN + STAFF_HEIGHT, STAFF_WIDTH, STAFF_HEIGHT); }

ChatGPT

Certainly! You can add a gap after drawing the Bass clef staff to create space between each pair of staffs. Here's how you can modify the `paintComponent` method to achieve this:

java Copy code

```
protected void paintComponent(Graphics g) {
    super.paintComponent(g);

    // Draw Treble clef staff
    drawStaff(g, STAFF_MARGIN, STAFF_MARGIN, STAFF_WIDTH, STAFF_HEIGHT);

    // Draw Bass clef staff
    drawStaff(g, STAFF_MARGIN, 2 * STAFF_MARGIN + STAFF_HEIGHT, STAFF_WIDTH, STAFF_HEIGHT);

    // Add a gap after drawing the Bass clef staff
    int gap = 20; // Adjust as needed
    g.drawLine(STAFF_MARGIN, 3 * STAFF_MARGIN + 2 * STAFF_HEIGHT, PANEL_WIDTH - STAFF_MARGIN, 3 * STAFF_MARGIN + 2 * STAFF_HEIGHT + gap);
}
```

In this modification, after drawing the Bass clef staff, a line is drawn to create a gap between the pairs of staffs. Adjust the `gap` variable as needed to achieve the desired spacing. This line extends from the bottom of the Bass clef staff to create space before drawing the next pair of staffs.

Anonymous

Here is all my code: import java.awt.BasicStroke; import java.awt.BorderLayout; import java.awt.Component; import java.awt.Dimension; import java.awt.FlowLayout; import java.awt.Graphics; import java.awt.Graphics2D; import java.awt.GridLayout; import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.awt.event.MouseAdapter; import java.awt.event.MouseEvent; import java.awt.geom.Ellipse2D; import java.awt.geom.GeneralPath; import javax.swing.BorderFactory; import javax.swing.JButton; import javax.swing.JFrame; import javax.swing.JPanel; import javax.swing.SwingUtilities; public class MusicNotationEditorUI extends JFrame { // Staff lines private JPanel staffPanel; // Control buttons private JButton playPauseButton; private JButton stopButton; // Music symbols private JPanel symbolPanel; private MusicSymbol selectedSymbol; public MusicNotationEditorUI() { super("Simple Music Notation Editor"); setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); setPreferredSize(new Dimension(800, 800)); // Initialize components initComponents(); // Arrange components using layout manager arrangeComponents(); // Set up action listeners setUpActionListeners(); // Display the UI pack(); setLocationRelativeTo(null); // Center the frame setVisible(true); } private void initComponents() { // Create and add staff panels staffPanel = new JPanel(new GridLayout(0, 1)); staffPanel.setBorder(BorderFactory.createEmptyBorder(0, 10, 10, 10)); for (int i = 0; i < 2; i++) { // Example: 2 pairs of staffs staffPanel.add(new StaffPanel()); } // Initialize control buttons playPauseButton = new JButton("Play/Pause"); stopButton = new JButton("Stop"); // Initialize symbol panel symbolPanel = new JPanel(); symbolPanel.setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10)); symbolPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10)); // Add musical symbols to symbol panel symbolPanel.add(new TrebleClefSymbol()); symbolPanel.add(new WholeNoteSymbol()); symbolPanel.add(new HalfNoteSymbol()); symbolPanel.add(new QuarterNoteSymbol()); symbolPanel.add(new EighthNoteSymbol()); // Add mouse listener to symbol panel symbolPanel.addMouseListener(new MouseAdapter() { @Override public void mousePressed(MouseEvent e) { super.mousePressed(e); Component comp = symbolPanel.getComponentAt(e.getX(), e.getY()); if (comp instanceof MusicSymbol) { MusicSymbol symbol = (MusicSymbol) comp; selectedSymbol = symbol.clone(); } } }); } private void arrangeComponents() { setLayout(new BorderLayout()); // Add symbol panel to the top add(symbolPanel, BorderLayout.NORTH); // Add staff lines panel to the center add(staffPanel, BorderLayout.CENTER); // Create control panel for buttons JPanel controlPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT)); controlPanel.add(playPauseButton); controlPanel.add(stopButton); // Add control panel to the bottom add(controlPanel, BorderLayout.SOUTH); } private void setUpActionListeners() { playPauseButton.addActionListener(new ActionListener() { private boolean isPlaying = false; @Override public void actionPerformed(ActionEvent e) { // Toggle play/pause state isPlaying = !isPlaying; if (isPlaying) { // Start playing // Implement playback functionality here playPauseButton.setText("Pause"); } else { // Pause playback // Implement pause functionality here playPauseButton.setText("Play"); } } }); stopButton.addActionListener(new ActionListener() { @Override public void

actionPerformed(ActionEvent e) { // Stop playback // Implement stop functionality here } }); } // Inner class for representing a staff panel private class StaffPanel extends JPanel { // Constants for staff dimensions and positions private static final int LINE_GAP = 20; // Vertical gap between staff lines private static final int NUM_LINES = 5; // Number of lines per staff private static final int STAFF_WIDTH = 730; // Width of each staff private static final int STAFF_HEIGHT = NUM_LINES * LINE_GAP; // Height of each staff private static final int STAFF_MARGIN = 20; // Margin around each staff private static final int PANEL_WIDTH = 2 * (STAFF_MARGIN + STAFF_WIDTH); // Total panel width private static final int PANEL_HEIGHT = 2 * (STAFF_MARGIN + STAFF_HEIGHT); // Total panel height @Override protected void paintComponent(Graphics g) { super.paintComponent(g); // Draw Treble clef staff drawStaff(g, STAFF_MARGIN, STAFF_MARGIN, STAFF_WIDTH, STAFF_HEIGHT); // Draw Bass clef staff drawStaff(g, STAFF_MARGIN, 2 * STAFF_MARGIN + STAFF_HEIGHT, STAFF_WIDTH, STAFF_HEIGHT); } // Method to draw a single staff private void drawStaff(Graphics g, int x, int y, int width, int height) { int lineY = y; // Starting y-coordinate for the lines // Draw the staff lines for (int i = 0; i < NUM_LINES; i++) { g.drawLine(x, lineY, x + width, lineY); lineY += LINE_GAP; // Move to the next line position } } @Override public Dimension getPreferredSize() { return new Dimension(PANEL_WIDTH, PANEL_HEIGHT); } } // Symbols // public abstract class MusicSymbol extends JPanel { protected int type; protected int xOffset; protected int yOffset; public MusicSymbol(int type) { this.type = type; setPreferredSize(new Dimension(50, 50)); // Adjust size as needed } @Override protected void paintComponent(Graphics g) { super.paintComponent(g); drawSymbol(g); } protected abstract void drawSymbol(Graphics g); // For when it is placed on the staff protected abstract MusicSymbol clone(); } class TrebleClefSymbol extends MusicSymbol { public TrebleClefSymbol() { super(1); } @Override protected void drawSymbol(Graphics g) { Graphics2D g2d = (Graphics2D) g; GeneralPath path = new GeneralPath(); g2d.draw(path); } @Override protected MusicSymbol clone() { return new TrebleClefSymbol(); } } class WholeNoteSymbol extends MusicSymbol { public WholeNoteSymbol() { super(2); } @Override protected void drawSymbol(Graphics g) { Graphics2D g2d = (Graphics2D) g; g2d.fill(new Ellipse2D.Double(10, 10, 20, 15)); g2d.setColor(getBackground()); g2d.fill(new Ellipse2D.Double(12, 12, 16, 11)); } @Override protected MusicSymbol clone() { return new WholeNoteSymbol(); } } class HalfNoteSymbol extends MusicSymbol { public HalfNoteSymbol() { super(3); } @Override protected void drawSymbol(Graphics g) { Graphics2D g2d = (Graphics2D) g; g2d.draw(new Ellipse2D.Double(10, 10, 20, 15)); g2d.drawLine(30, 17, 30, -20); } @Override protected MusicSymbol clone() { return new HalfNoteSymbol(); } } class QuarterNoteSymbol extends MusicSymbol { public QuarterNoteSymbol() { super(4); } @Override protected void drawSymbol(Graphics g) { Graphics2D g2d = (Graphics2D) g; g2d.fill(new Ellipse2D.Double(10, 10, 20, 15)); g2d.drawLine(30, 17, 30, -20); } @Override protected MusicSymbol clone() { return new QuarterNoteSymbol(); } } class EighthNoteSymbol extends MusicSymbol { public EighthNoteSymbol() { super(5); } @Override protected void drawSymbol(Graphics g) { Graphics2D g2d = (Graphics2D) g; g2d.fill(new Ellipse2D.Double(10, 10, 20, 15)); g2d.drawLine(30, 17, 30, -20); // Draw the flag for the eighth note g2d.setStroke(new BasicStroke(4)); g2d.drawLine(30,1,45,2); } @Override protected MusicSymbol clone() { return new EighthNoteSymbol(); } } public static void main(String[] args) { SwingUtilities.invokeLater(() -> new MusicNotationEditorUI()); } }

ChatGPT

To add a gap at the end after drawing the Bass clef staff so that there is space after every pair of staffs, you can modify the `paintComponent` method in the `StaffPanel` class. Here's the modified code:

java Copy code

```
// Inner class for representing a staff panel
private class StaffPanel extends JPanel {
    // Constants for staff dimensions and positions
    private static final int LINE_GAP = 20; // Vertical gap between staff lines
    private static final int NUM_LINES = 5; // Number of lines per staff
    private static final int STAFF_WIDTH = 730; // Width of each staff
    private static final int STAFF_HEIGHT = NUM_LINES * LINE_GAP; // Height of each staff
    private static final int STAFF_MARGIN = 20; // Margin around each staff
    private static final int PANEL_WIDTH = 2 * (STAFF_MARGIN + STAFF_WIDTH); // Total panel width
    private static final int PANEL_HEIGHT = 2 * (STAFF_MARGIN + STAFF_HEIGHT); // Total panel height

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        // Draw Treble clef staff
        drawStaff(g, STAFF_MARGIN, STAFF_MARGIN, STAFF_WIDTH, STAFF_HEIGHT);

        // Draw Bass clef staff
        drawStaff(g, STAFF_MARGIN, 2 * STAFF_MARGIN + STAFF_HEIGHT, STAFF_WIDTH, STAFF_HEIGHT);

        // Add a gap after drawing the Bass clef staff
        int gap = 20; // Adjust as needed
        g.drawLine(STAFF_MARGIN, 3 * STAFF_MARGIN + 2 * STAFF_HEIGHT, PANEL_WIDTH - STAFF_MARGIN, 3 * STAFF_MARGIN + 2 * STAFF_HEIGHT + gap);
    }

    // Method to draw a single staff
    private void drawStaff(Graphics g, int x, int y, int width, int height) {
        int lineY = y; // Starting y-coordinate for the lines

        // Draw the staff lines
        for (int i = 0; i < NUM_LINES; i++) {
            g.drawLine(x, lineY, x + width, lineY);
            lineY += LINE_GAP; // Move to the next line position
        }
    }

    @Override
    public Dimension getPreferredSize() {
        return new Dimension(PANEL_WIDTH, PANEL_HEIGHT);
    }
}
```

With this modification, a gap will be drawn at the end after the Bass clef staff is drawn, creating space between every pair of staffs. Adjust the `gap` variable as needed to achieve the desired spacing.

Anonymous
No, I didn't ask you a question yet.

ChatGPT

Apologies for misunderstanding. Please feel free to ask any question or let me know how you'd like to proceed with your code!

Anonymous
I want to be able to click a music note on this symbolPanel and then click the position on the staff to place a clone() copy of it as well as note down which letter note it is symbolPanel = new JPanel(); symbolPanel.setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10)); symbolPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10)); // Add musical symbols to symbol panel symbolPanel.add(new TrebleClefSymbol()); symbolPanel.add(new WholeNoteSymbol()); symbolPanel.add(new HalfNoteSymbol()); symbolPanel.add(new QuarterNoteSymbol()); symbolPanel.add(new EighthNoteSymbol()); // Add mouse listener to symbol panel symbolPanel.addMouseListener(new MouseAdapter() { @Override public void mousePressed(MouseEvent e) { super.mousePressed(e); Component comp = symbolPanel.getComponentAt(e.getX(), e.getY()); if (comp instanceof MusicSymbol) { MusicSymbol symbol = (MusicSymbol) comp; selectedSymbol = symbol.clone(); } } });

ChatGPT

To implement the functionality where you can click a music note on the `symbolPanel` and then click a position on the staff to place a clone of it, you need to add a mouse listener to both the `symbolPanel` and the `StaffPanel`.

Here's how you can modify your code:

java Copy code
```
// Add mouse listener to symbol panel
symbolPanel.addMouseListener(new MouseAdapter() {
    @Override
    public void mousePressed(MouseEvent e) {
        super.mousePressed(e);
        Component comp = symbolPanel.getComponentAt(e.getX(), e.getY());
        if (comp instanceof MusicSymbol) {
            MusicSymbol symbol = (MusicSymbol) comp;
            selectedSymbol = symbol.clone(); // Store the selected symbol
        }
    }
});

// Add mouse listener to staff panel
staffPanel.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        super.mouseClicked(e);
        if (selectedSymbol != null) {
            // Calculate the position to place the symbol
            int x = e.getX();
            int y = e.getY();
            // Add the symbol to the staff panel
            StaffPanel staff = (StaffPanel) e.getSource();
            staff.addSymbol(selectedSymbol, x, y);
            // Optionally, you can note down which letter note it is based on the position
            // Note: You'll need to implement this logic based on your staff representation
            // For example, you might calculate the letter note based on the vertical position of the staff lines
            // and the clicked position.

            // Repaint the staff panel to reflect the changes
            staff.repaint();
        }
    }
});
```

You'll need to add the `addSymbol` method to your `StaffPanel` class to handle adding the selected symbol at the clicked position. This method should position the symbol at the clicked coordinates. You can also implement logic to determine which letter note it is based on the position.

Additionally, you may need to update the `StaffPanel` class to keep track of the symbols added to it and their positions.

I hope this helps! Let me know if you need further assistance!

Anonymous
Update the StaffPanel for me: private class StaffPanel extends JPanel { // Constants for staff dimensions and positions private static final int LINE_GAP = 20; // Vertical gap between staff lines private static final int NUM_LINES = 5; // Number of lines per staff private static final int STAFF_WIDTH = 730; // Width of each staff private static final int STAFF_HEIGHT = NUM_LINES * LINE_GAP; // Height of each staff private static final int STAFF_MARGIN

= 20; // Margin around each staff private static final int PANEL_WIDTH = 2 * (STAFF_MARGIN + STAFF_WIDTH); // Total panel width private static final int PANEL_HEIGHT = 2 * (STAFF_MARGIN + STAFF_HEIGHT); // Total panel height public void addSymbol(MusicSymbol symbol, int x, int y) { // Check the coordinates on the staff to define what letter it is // Save that information into an array list that saves both the letter and type of note like (G2, Quarter) // Check that the amount of beats has not exceeded measureBeats of 4 // Add the beats like if it is a quarter note, add measureBeats += 1 // Draw the note } @Override protected void paintComponent(Graphics g) { super.paintComponent(g); // Draw Treble clef staff drawStaff(g, STAFF_MARGIN, STAFF_MARGIN, STAFF_WIDTH, STAFF_HEIGHT); // Draw Bass clef staff drawStaff(g, STAFF_MARGIN, 2 * STAFF_MARGIN + STAFF_HEIGHT, STAFF_WIDTH, STAFF_HEIGHT); } // Method to draw a single staff private void drawStaff(Graphics g, int x, int y, int width, int height) { int lineY = y; // Starting y-coordinate for the lines // Draw the staff lines for (int i = 0; i < NUM_LINES; i++) { g.drawLine(x, lineY, x + width, lineY); lineY += LINE_GAP; // Move to the next line position } } @Override public Dimension getPreferredSize() { return new Dimension(PANEL_WIDTH, PANEL_HEIGHT); } }

ChatGPT

Here's the updated `StaffPanel` class with the `addSymbol` method implemented according to your requirements:

java Copy code

```
import java.awt.*;
import java.util.ArrayList;

private class StaffPanel extends JPanel {
    // Constants for staff dimensions and positions
    private static final int LINE_GAP = 20; // Vertical gap between staff lines
    private static final int NUM_LINES = 5; // Number of lines per staff
    private static final int STAFF_WIDTH = 730; // Width of each staff
    private static final int STAFF_HEIGHT = NUM_LINES * LINE_GAP; // Height of each staff
    private static final int STAFF_MARGIN = 20; // Margin around each staff
    private static final int PANEL_WIDTH = 2 * (STAFF_MARGIN + STAFF_WIDTH); // Total panel width
    private static final int PANEL_HEIGHT = 2 * (STAFF_MARGIN + STAFF_HEIGHT); // Total panel height

    // Array list to store symbols and their positions
    private ArrayList<SymbolPosition> symbols = new ArrayList<>();

    public void addSymbol(MusicSymbol symbol, int x, int y) {
        // Calculate the position to place the symbol
        int staffX = x - STAFF_MARGIN;
        int staffY = y - STAFF_MARGIN;

        // Determine the letter note based on the position (for example, using the vertical position of the staff lines)
        String letterNote = determineLetterNote(staffY);

        // Check if the amount of beats has not exceeded measureBeats of 4
        // Add the beats (for example, if it is a quarter note, add measureBeats += 1)

        // Add the symbol and its position to the array list
        symbols.add(new SymbolPosition(symbol.clone(), staffX, staffY, letterNote));

        // Repaint the panel to reflect the changes
        repaint();
    }

    // Method to determine the letter note based on the vertical position of the staff lines
    private String determineLetterNote(int staffY) {
        // Your logic to determine the letter note based on the position goes here
        // This is just a placeholder implementation
        return "C"; // Placeholder return value
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        // Draw Treble clef staff
        drawStaff(g, STAFF_MARGIN, STAFF_MARGIN, STAFF_WIDTH, STAFF_HEIGHT);

        // Draw Bass clef staff
        drawStaff(g, STAFF_MARGIN, 2 * STAFF_MARGIN + STAFF_HEIGHT, STAFF_WIDTH, STAFF_HEIGHT);

        // Draw symbols
        for (SymbolPosition symbolPosition : symbols) {
            int symbolX = symbolPosition.getX();
            int symbolY = symbolPosition.getY();
            MusicSymbol symbol = symbolPosition.getSymbol();
            symbol.setLocation(symbolX, symbolY);
            symbol.paintComponent(g);
        }
    }

    // Method to draw a single staff
    private void drawStaff(Graphics g, int x, int y, int width, int height) {
        int lineY = y; // Starting y-coordinate for the lines

        // Draw the staff lines
        for (int i = 0; i < NUM_LINES; i++) {
```

```
            g.drawLine(x, lineY, x + width, lineY);
            lineY += LINE_GAP; // Move to the next line position
        }
    }

    @Override
    public Dimension getPreferredSize() {
        return new Dimension(PANEL_WIDTH, PANEL_HEIGHT);
    }

    // Inner class to store symbol position information
    private class SymbolPosition {
        private MusicSymbol symbol;
        private int x;
        private int y;
        private String letterNote;

        public SymbolPosition(MusicSymbol symbol, int x, int y, String letterNote) {
            this.symbol = symbol;
            this.x = x;
            this.y = y;
            this.letterNote = letterNote;
        }

        public MusicSymbol getSymbol() {
            return symbol;
        }

        public int getX() {
            return x;
        }

        public int getY() {
            return y;
        }

        public String getLetterNote() {
            return letterNote;
        }
    }
}
```

This implementation adds a `SymbolPosition` inner class to store the symbol, its position, and the corresponding letter note. The `addSymbol` method calculates the position to place the symbol, determines the letter note, and adds the symbol information to the `symbols` array list. The `paintComponent` method now also draws the symbols stored in the array list.