

Java Swing Graphical User Interface (GUI) Design and Construction

Assignments 4 and 5 for CSC 243, Spring, 2010, Dr. Dale E. Parson

<http://faculty.kutztown.edu/parson/spring2010/CSC243Spring2010.html>

Assignment 4 is due 11:59 PM on April 20. Use **gmake turnitin** to turn it in.

Assignment 5 is due 11:59 PM on May 2. Use **gmake turnitin** to turn it in.

COME TO CLASS! I will be giving interactive demos and explanations that you will need.

ASSIGNMENT 4:

Perform the following steps to copy and inspect my initial code handout.

```
cp ~parson/JavaLang/game_assign_4_5.zip ~/JavaLang
cd ~/JavaLang
/bin/unzip game_assign_4_5.zip
cd ./games2010rev2
gmake clean test
```

Unzipping creates two Java package hierarchies in your JavaLang directory, **games2010rev4** that holds the game, and **gamestomidi2010rev4** that holds the game-to-MIDI translator. **Your work for this assignment will occur in file gamestomidi2010rev4/scrabble/ScrabbleToMidiBoardNoteVisualizer.**

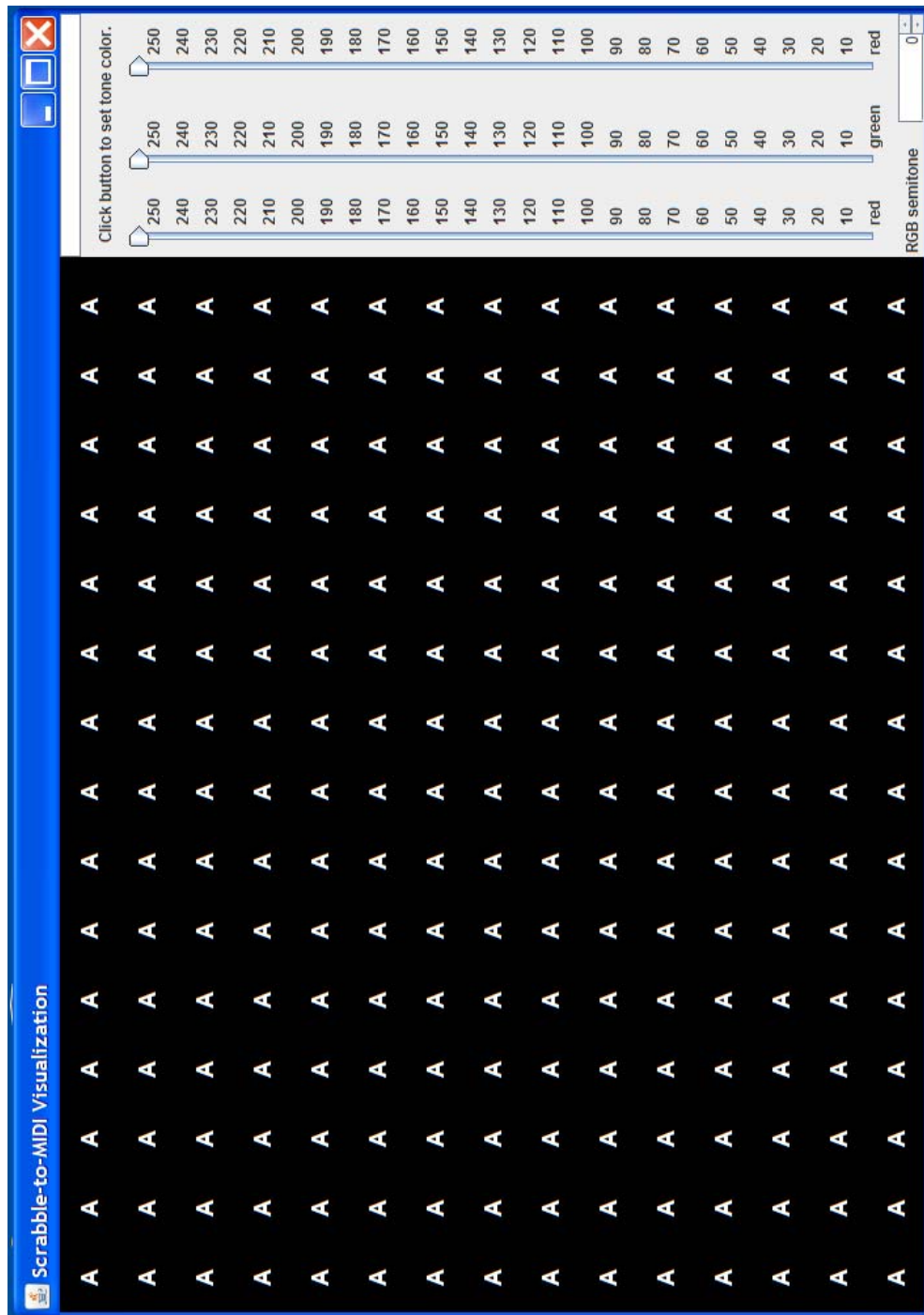
Running **gmake test** invokes the following test driver:

```
java gamestomidi2010rev4.scrabble.ScrabbleToMidiBoardNoteVisualizer
```

When running on bill make sure to run Exceed on your PC and use a putty session with X11 enabled.

Class `ScrabbleToMidiBoardNoteVisualizer` is a `JFrame`-derived class that implements the game board-to-music visualization interface that we have been discussing in class. The **main** method in `ScrabbleToMidiBoardNoteVisualizer` is a test driver that constructs an object of this class for purposes of testing. **Your goal in this assignment** is to construct and display the graphical components of this class as driven by the stand-alone tests in its main method. Your goal in Assignment 5 will be to construct event handlers for the event-producing GUI components of this class as explained in the “ASSIGNMENT 5” section below. The current section focuses on ASSIGNMENT 4.

Figure 1 shows a `ScrabbleToMidiBoardNoteVisualizer` object as displayed by its main test driver. Your goal in Assignment 4 is to write class `ScrabbleToMidiBoardNoteVisualizer`'s fields and constructor so that it appears similar to Figure 1 or better. Figure 1 shows a static display with each note location on the virtual board set to a white letter “A.” For assignment 4 you need to match or better this Figure 1. For assignment 5 there will not be a matrix of white A. Instead the letters will flash on and off according to location and with a color according to the note played. For now we will concentrate on achieving the static display of Figure 1.



15 x 15 matrix of currently playing notes as a matrix of JLabel objects (They will not be all white in Assignment 5. They will display only when their notes are being played, in a color indexed by the note number as explained below. This illustration is for Assignment 4.)

Figure 1: Class ScrabbleToMidiBoardNoteVisualizer as a displayed JFrame

In addition to the overall JFrame (using BorderLayout) of class ScrabbleToMidiBoardNoteVisualizer, there is a JPanel for the 15 x 15 board in the **Center** with the black background, a JPanel for all the controls in the **East** side of Figure 1 (**BorderLayout**), within which there are 3 nested JPanels. The top nested JPanel holds the color button (blank and white in Figure 1) and the “Click button to set tone color.” JLabel, the middle nested JPanel holds the 3 labeled JSliders, and the bottom nested JPanel holds the “RGB semitone” JLabel and the JSpinner. You must construct this display in Assignment 4 by constructing a 2D matrix of JLabel objects for the tiles, with the background set to black as displayed, with each holding a white letter “A” for now. You must also construct the color-indicating JButton at the top right (white and blank for assignment 4), the three JSliders, the JSpinner and the JLabels on the right side of Figure 1. The components on the right side use default foreground and background color. The components in the tile matrix use a background of black. The foreground is white for Assignment 4. **The point size of the JLabel objects in the 2D matrix must be 18.** You can set this point size by constructing the JLabels, then using getFont() to get Font for one of them, then constructing a new Font object with the same name and style but with a size of 18, and then using setFont to apply this font to the 15 x 15 JLabels. Fonts are covered in portions of the textbook that we went over, as are colors. See java.awt.Font and java.awt.Color.

Outer JFrame

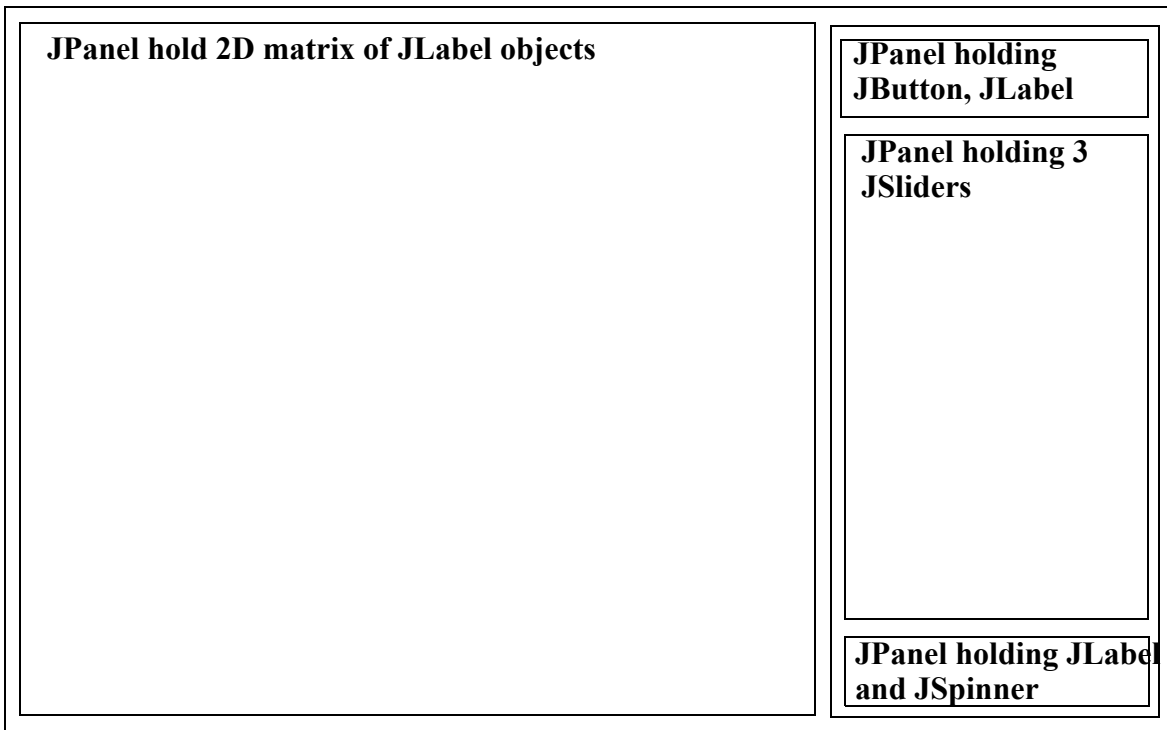


Figure 2: Nested JPanel objects of Figure 1

NOTES ON THE COMPONENTS IN FIGURES 1 AND 2:

JSLIDER: Use the constructor that takes parameters “JSlider(int orientation, int min, int max, int value)” and set orientation to **SwingConstants.VERTICAL**. Min is 0 and max is 255 for each

color. The value for Assignment 4 is 255 for each (all white), but the value for each of the RGB sliders for Assignment 5 must match your color for the default color of the RGB semitone on the JSpinner (semitone 0).

Getting the labels on each JSpinner takes a bit of work. Assuming you have a JSpinner field called redSlider, **here is the code for installing the labels**. We will go over this in class.

```
redSlider.setPaintLabels(true);
Hashtable<Integer, JComponent> rtab = (Hashtable<Integer, JComponent>)
    redSlider.createStandardLabels(10); // label it in increments of 10
rtab.put(new Integer(0), new JLabel("red")); // replace 0 with the slider's color
redSlider.setLabelTable(rtab);
```

JSPINNER: Use the constructor that takes parameters “public JSpinner(SpinnerModel model)” where the model is a *SpinnerNumberModel* constructed with “SpinnerNumberModel(int value, int minimum, int maximum, int stepSize) “. The value range is 0 .. 11 for the twelve semitones in a Western scale, the stepSize is 1, and the initial value is 0.

Make sure to **CENTER** the text on all of your JLabels including the matrix using the JLabel constructor that takes **SwingConstants.CENTER** as its second argument.

On components in the 2D matrix and on the JButton, use methods **setForeground** and **setBackground** to set colors and **setText** to change displayed text. Use **getFont** and **setFont** on the matrix labels to set point size to 18 as previously discussed.

REREAD these instructions as a checklist before you turn in Assignment 4.

Running **gmake test** for Assignment 4 should give the display of Figure 1 with a JButton, JSliders and a JSpinner that display and move through the correct values. I will take off points if the value ranges are incorrect. **Wait until Assignment 5 to write and connect event listeners.**

Here is the handout code for `ScrabbleToMidiBoardNoteVisualizer.java`. Note the STUDENT requirements.

gamestomidi2010rev4/scrabble/ScrabbleToMidiBoardNoteVisualizer.java

```
1  /*   ScrabbleToMidiBoardNoteVisualizer.java -- class for visualizing
2      Synth sounds generated from Scrabble. Adapted from
3      ScrabbleToMidiSynthVisualizer Spring 2010 CSC 243 Java Programming
4      for board visualization. See STUDENT requirements below.
5  */
6
7  package gamestomidi2010rev4.scrabble ;
8  import gamestomidi2010rev4.GameToMidiSynthEvent ;
9  import gamestomidi2010rev4.GameToMidiSynthEventListener ;
10 import java.util.Hashtable ;
11 import javax.sound.midi.MidiMessage ;
```

```

12 import javax.sound.midi.ShortMessage ;
13 import java.awt.* ;
14 import java.awt.event.* ;
15 import javax.swing.* ;
16 import javax.swing.event.* ;
17
18 /**
19  * This class provides GUI visualization for text-translated MIDI events
20  * on an game-board-like dynamic display as they are sent to a
21  * javax.sound.midi.Synthesizer for playing. This class lights up letters
22  * at their locations on a black background, using a color determined
23  * by each note's semitone distance (0..11) fro the tonic (DO in the scale)
24  * and a user-adjustable stacked display order for the 16 instrument
25  * voices 0..15.
26  * Note: Not all MIDI events have accompanying text.
27  */
28
29 public class ScrabbleToMidiBoardNoteVisualizer extends JFrame
30     implements GameToMidiSynthEventListener {
31
32     // PARSON symbolic constants. STUDENT -- Please note.
33     private static final int BOARD_SIZE = 15 ; // number of coulms and rows
34     private static final int CHANNELS = 16 ; // MIDI "voices," chan below.
35     // ASSIGNMENT 4 STUDENT
36     // STUDENT -- Declare your private FIELDS here and initialize them
37     // either here or in the constructor below.
38
39
40     // STUDENT -- halftones is used in a debugging print statement below.
41     private static String [] halftones = {
42         "tonic",
43         "minor 2nd",
44         "major 2nd",
45         "minor 3rd",
46         "major 3rd",
47         "perfect 4th",
48         "tritone",
49         "perfect fifth",
50         "minor 6th",
51         "major 6th",
52         "minor 7th",
53         "major 7th"
54     };
55     /**
56     * Construct the JFrame for displaying the visualization board,
57     * constructing its nested JPanel, JButton, JSlider, JSpinner

```

```

58     * and any other GUI widgets and constructing and subscribing any
59     * event listener objects.
60     **/
61     public ScrabbleToMidiBoardNoteVisualizer() {
62         // ASSIGNMENT 4 STUDENT -- Write this constructor to create and
63         // display components as they appear in Figures 1 and 2 of the handout.
64         // Do not hook up the event listeners.
65         // Set the size of this JFrame to 1000 x 700 and set the close
66         // operation to JFrame.EXIT_ON_CLOSE. The textbook's JFrame
67         // code examples will be useful for seeing how to do this.
68
69         // At least some of the widgets should be referenced by FIELDS
70         // in this object rather than by local variables in this methods.
71         // Declare the fields where noted above.
72
73         // ASSIGNMENT 5 STUDENT: Add the ChangeListener(s) for
74         // the JSliders and JSpinner and the ActionListener(s) for the
75         // JButton. The JSliders affect the color of displayed JButton,
76         // which transfers its color to the semitone selected by the
77         // JSpinner only when the JButton is clicked. See Assignment 5
78         // part of the handout for details. See additional STUDENT
79         // instructions below for Assignment 5.
80
81     }
82
83     /**
84     * Notify this event listener of a new GameToMidiSynthEvent.
85     * This method displays the NOTEON letters and ends display
86     * of the NOTEOFF letters at their board positions with color
87     * according to scale steps. The topmost NOTEON in the
88     * user-adjustable stacking order is the one displayed at that
89     * particular location; when it goes NOTEOFF, any letter beneath it
90     * is displayed, unless there are no notes beneath it, in which case it
91     * goes blank. There are only 15 x 15 JButtons to display, and each
92     * button displays either the letter of the topmost of 16 layers
93     * which is currently NOTEON, or a black background if none are NOTEON.
94     *
95     * @param event the event being sent to a Synthesizer
96     *
97     **/
98     public void notifyMidiSynthEvent(GameToMidiSynthEvent event) {
99         String textword = event.getTextword();
100        int start = event.getStart();
101        int end = event.getEnd();
102        String location = event.getLocation() ;
103        // You can also drill down into the MidiEvent base class to

```

```

104 // get MIDI data such as command, channel, data1 and data2
105 // if useful.
106 int chan = -1 ;
107 MidiMessage mmsg = event.getMessage();
108 ShortMessage smsg = null ;
109 if (mmsg instanceof ShortMessage) {
110     smsg = (ShortMessage) mmsg ;
111     chan = smsg.getChannel();
112 }
113 // New stuff added Oct. 2009 for showing interval from scale's root.
114 String interval = null ;
115 boolean isDisplayChange = false ;
116 boolean isNoteOn = false ;
117 boolean areAllNotesOff = false ;
118 int noteOffsetFromTonic = -1 ;
119 int octave = -1, tonic = event.getTonic();
120 if (smsg != null) {
121     int cmd = smsg.getCommand();
122     if (tonic > -1) {
123         int basetonic = tonic % 12 ; // place in the chromatic scale
124         if (cmd == ShortMessage.NOTE_ON
125             || cmd == ShortMessage.NOTE_OFF) {
126             isDisplayChange = true ;
127             interval = (cmd == ShortMessage.NOTE_ON) ? " ON " : " OFF ";
128             isNoteOn = (cmd == ShortMessage.NOTE_ON) ? true : false ;
129             int note = smsg.getData1();
130             int noffset = note % 12 ;
131             if (noffset < basetonic) {
132                 // Offset must be above the tonic.
133                 noffset += 12 ;
134             }
135             octave = note / 12 ;
136             noteOffsetFromTonic = noffset - basetonic ;
137             interval = interval + halftones[noteOffsetFromTonic]
138                 + " ";
139         }
140     }
141     if (cmd == ShortMessage.CONTROL_CHANGE
142         && smsg.getData1() == 123) {
143         // ALL NOTES OFF! MESSAGE
144         areAllNotesOff = true ;
145         isNoteOn = false ;
146         isDisplayChange = true ;
147     }
148 }
149 String locreport = "" ;

```

```

150     if (location != null) {
151         locreport = "\tloc = " + location ;
152     }
153     if (textword != null && start > -1 && end > -1) {
154         String noteinfo = new String(); ;
155         if (interval != null && octave > -1) {
156             noteinfo = interval + "\toctave = " + octave ;
157         }
158         //** DEBUGGING INFO
159         System.err.println("DEBUG VISUAL " + event.getTick() + " "
160             + chan + "\t"
161             + textword.substring(start, end) + " "
162             + textword + "\t" + noteinfo + locreport) ;
163         //**/
164         if (isDisplayChange && chan > -1) {
165             // STUDENT SPRING 2010 ASSIGNMENT 5 CODE GOES BELOW:
166             // Your must use the values in these predefined variables:
167             // 1. chan -- is the MIDI channel 0 .. 15 inclusive, also
168             // know as the instrument being played or silenced.
169             // It determines the levels 0..15 of board tile being
170             // displayed/modified.
171             // 2. areAllNotesOff -- if this is true, it is equivalent
172             // to setting all notes ON ITS CHANNEL (instrument level)
173             // to NOTEOFF.
174             // 3. isNoteOn -- if true then this message is turning a
175             // NOTEON, else it is turning a NOTEOFF. Note that
176             // areAllNotesOff is the same as turning all notes
177             // on this channel off.
178             // 4. noteOffsetFromTonic -- if this number is 0..11,
179             // then it is the number to select your color. If
180             // areAllNotesOff is true then this number will be -1.
181             // 5. letterToDisplay -- this is a single-letter String that
182             // is the letter displayed on the board.
183             // 6. columnHeading is the single-letter char 'A'..'O'
184             // representing the column on the board, where 'A' is the
185             // leftmost column and 'O' is the rightmost.
186             // Note that it is possible to subtract like this:
187             //     columnHeading - 'A'
188             // to get an integer offset 0..14 from 'A'.
189             // 7. rowHeading is the String "1" .. "15" that gives
190             // the row of letterToDisplay on the board, where "15" is
191             // the TOP row of the board and "1" is the bottom row.
192             String letterToDisplay ;
193             char columnHeading ;
194             String rowHeading ;
195             if (areAllNotesOff) {

```

```

196         letterToDisplay = “ “;
197         columnHeading = ‘A’; // dummy value, not needed
198         rowHeading = “0” ; // dummy value, not needed
199         noteOffsetFromTonic = 0 ; // dummy value
200     } else {
201         letterToDisplay = (isNoteOn ?
202             textword.substring(start, start+1) : “ “);
203         columnHeading = location.charAt(0);
204         rowHeading = location.substring(1);
205     }
206     // STUDENT ASSIGNMENT 5
207     // STUDENT: Put your MidiEvent-driven display changes here:
208 }
209
210 }
211 }
212
213 public static void main(String [] args) {
214     // STUDENT: Do not add anything to this main() function.
215     // It is here strictly as a test driver.
216     // All the work of initializing this JFrame goes in the
217     // constructor above. That constructor will normally be called
218     // from the Scrabble-to-MIDI framework. For assignment 3 we are
219     // calling it from here. MAKE NO CHANGES TO MAIN!
220     ScrabbleToMidiBoardNoteVisualizer testobject
221         = new ScrabbleToMidiBoardNoteVisualizer();
222 }
223 }

```

ASSIGNMENT 5:

Assignment 5 handles events from two sources. Events from the JButton, JSliders and JSpinner go to event listeners that you must write. MIDI note events come in through method notifyMidiSynthEvent above; your changes go into the STUDENT section at its bottom.

To test Assignment 5, run “**gmake test**” after every change. On a PC you can use Eclipse, or you can just run “javac *.java” from within the scrabble and gamestomidi2010rev4 directories, and then run **java gamestomidi2010rev4.scrabble.ScrabbleToMidiBoardNoteVisualizer** from within the gamestomidi2010rev4 directory. Make sure that this test does not blow up.

To test Assignment 5 event handling with a Scrabble game running, use **gmake testgui4** or its equivalent from the gamestomidi2010rev4 directory (on a command line):

```

java games2010rev4.GameMain
gamestomidi2010rev4.scrabble.ScrabbleToMultiMidiEngineLocalWithConfigUI
gamestomidi2010rev4.scrabble.ScrabbleToMultiMidiFrameUI 2

```

This test dynamically loads your class and sets it up as an event handler for MIDI events after invoking your constructor. A lot of text prints out. You can reduce some of this text by commenting out the `System.err.println` at line 159 above. If you want to capture the `System.err` output from your program on bill or another UNIX system, run this:

```
gmake testgui4 2> junk
```

and then examine the junk file to see any errors. **When running on bill make sure to run Exceed on your PC and use a putty session with X11 enabled.**

Here is what should happen with GUI events:

JSpinner: A `ChangeEvent` from each `JSpinner` should update the color `JButton` at the top right of Figure 1. This color does not transfer to the current semitone being colored until the `JButton` is clicked. The `JSpinners` must be initialized to hold values according to semitone 0's color at construction time (its default value).

JButton: An `ActionEvent`'s listener should transfer the `JButton`'s color (which is set by the three `JSliders`) onto that semitone. Initialize the `JButton` and `JSliders` in the constructor for the default color on semitone 0. Also, **UPDATE THE MATRIX IF APPROPRIATE**, i.e., if that semitone is currently being played by any topmost letters on the matrix. See below.

JSpinner: When a `ChangeEvent` shows a semitone change in the spinner's range 0 .. 11, set the `JSliders` and the `JButton` to show the color previously saved (or initialized) for that semitone. Initialize the `JSpinner` to 0 for semitone 0. The `JSpinner` must be read when the `JButton` `ActionEvent` occurs in order to determine the semitone to update, using `JSpinner`'s `getValue()` cast to an `Integer`. `JSlider` also has a `getValue()` method.

notifyMidiSynthEvent: STUDENT comments above show where your code must go. When `NOTEON`, `NOTEOFF` and `ALLNOTESOFF` messages arrive you must turn on or off foreground text (using `setText()`) and color (using `setForeground`) for the `JLabels` in the matrix. For this assignment channel 0 (variable `chan` above) is the top layer and channel 15 is the bottom. We are not rotating them. The topmost layer that has a note on must show that note IN ITS COLOR FOR THAT layer. A letter on one of the 16 layers may have a different semitone offset (variable `noteOffsetFromTonic` above) from the same letter on another layer. You must display the color of the topmost layer 0 .. 15 for which that note is currently on. When a `NOTEON` occurs, make sure that the note is not already displaying at a higher level before showing it. When a `NOTEOFF` occurs, do not erase the displayed note if a `NOTEON` should be showing on another layer. **This requirement means that you must keep a 3D matrix of information for currently displayed color information of the dimension `BOARDSIZE x BOARDSIZE x CHANNELS` (see above).**

If you do not get the color of displayed tiles on the 2D matrix perfect, you will still get most of the points for this assignment. Make sure to get the event listeners working, and make sure to display

all NOTEON letters/locations that are currently active and to hide (blacken) all other locations on the 2D matrix.

Assignment 5 is due at the end of the day on May 2. Please get something working and turn it in on time. If you are going to be late, let me know. Invoke **gmake turnitin** after both **gmake test** (retesting Assignment 4) and **gmake testgui4** (testing Assignment 5) are working as well as you can get them to work. May the Force be with You!

Come to class. I will demo aspects of Assignment 5 and will suggest approaches to coding in class.

I have found that the Syntheizer mode used to test Assignment 5 does not generate sounds on some Windows machines while it does on others. It works fine on Macs that I have tried. You do not need sound to work on this assignment.

The `System.err.println` statement in `ScrabbleToMidiBoardNoteVisualizer.java` produces reams of output as shown below, which may be useful in debugging. You can comment that statement out if you do not want all of this terminal output.

```
DEBUG VISUAL 1271120944905 0: E DEEP ON tonic octave = 4 loc = J8
DEBUG VISUAL 1271120945119 1: E DEEP OFF tonic octave = 3 loc = I8
DEBUG VISUAL 1271120945119 1: E DEEP ON tonic octave = 3 loc = J8
DEBUG VISUAL 1271120945119 0: E DEEP OFF tonic octave = 4 loc = I8
DEBUG VISUAL 1271120945119 0: E DEEP OFF tonic octave = 4 loc = J8
DEBUG VISUAL 1271120945119 0: P DEEP ON major 3rd octave = 6 loc = K8
DEBUG VISUAL 1271120945119 0: D DEEP ON major 3rd octave = 4 loc = H8
DEBUG VISUAL 1271120945333 1: E DEEP OFF tonic octave = 3 loc = J8
DEBUG VISUAL 1271120945333 1: P DEEP ON minor 3rd octave = 5 loc = K8
DEBUG VISUAL 1271120945333 0: P DEEP OFF major 3rd octave = 6 loc = K8
DEBUG VISUAL 1271120945333 0: D DEEP OFF major 3rd octave = 4 loc = H8
DEBUG VISUAL 1271120945333 0: E DEEP ON tonic octave = 4 loc = I8
DEBUG VISUAL 1271120945333 0: E DEEP ON tonic octave = 4 loc = J8
DEBUG VISUAL 1271120945547 1: P DEEP OFF minor 3rd octave = 5 loc = K8
```

DEBUG VISUAL 1271120945547 0: E DEEP OFF tonic octave = 4 loc = I8

DEBUG VISUAL 1271120945547 0: E DEEP OFF tonic octave = 4 loc = J8

DEBUG VISUAL 1271120945560 1: D DEEP ON minor 3rd octave = 3 loc = H8

DEBUG VISUAL 1271120945774 1: D DEEP OFF minor 3rd octave = 3 loc = H8

The first field is a timestamp, then a channel number 0 .. 15, then the letter followed by its word, then ON or OFF for NOTEON or NOTEOFF, then its semitone from table halftones in the code listing above, then octave information (which you can ignore) and location (which you cannot ignore). Location coordinates are mapped as follows.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
15	3w			2l			3w			2l		3w		
14	2w			3l			3l			2w				
13		2w			2l		2l			2w				
12	2l		2w			2l			2w		2l			
11			2w				2w							
10		3l			3l			3l			3l			
9		2l			2l		2l			2l				
8	3w		2l			2w			2l		3w			
7		2l			2l		2l			2l				
6		3l			3l			3l			3l			
5			2w				2w							
4	2l		2w			2l			2w		2l			
3		2w			2l		2l			2w				
2	2w			3l			3l			2w				
1	3w		2l			3w			2l		3w			