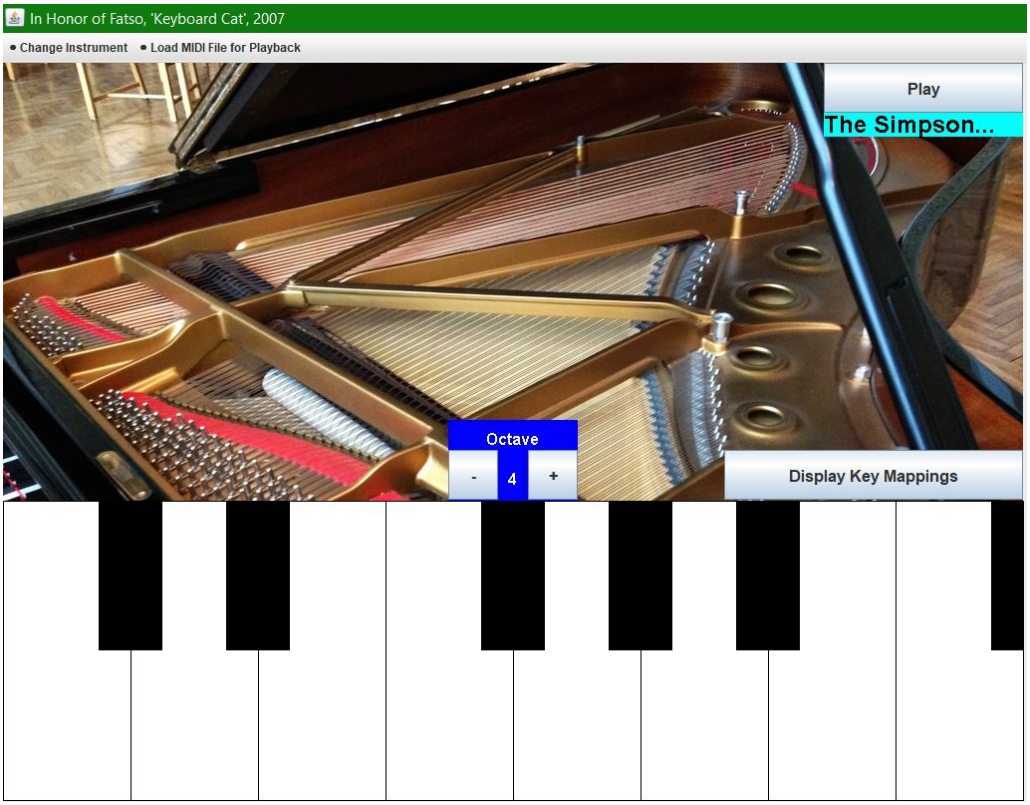
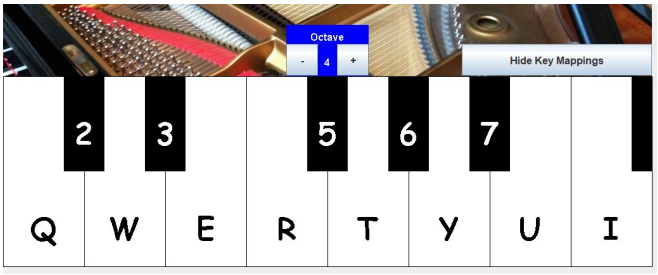
Creators: Nat Roth, Sabir Meah, David Shuster
For COSC-112, Prof. Alfeld
Last Updated May 4, 2020

Description of Project

Our application enables the user to play on a musical keyboard.

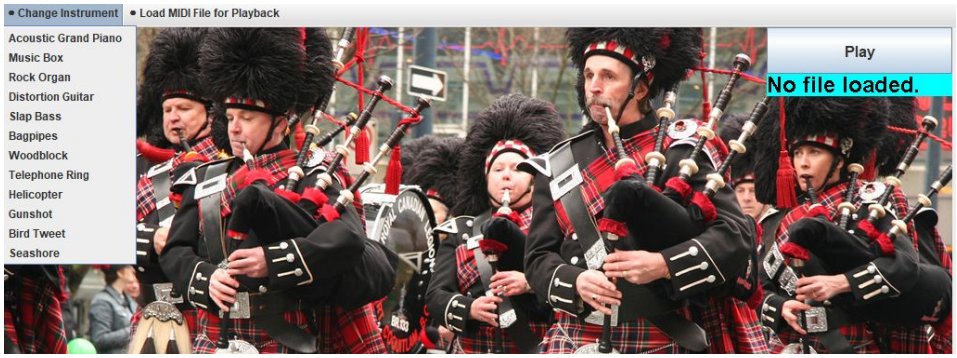


The computer keyboard keys 'q', '2', 'w', '3', 'e', 'r', '5', 't', '6', 'y', '7', 'u', 'i' correspond to C4 (middle C on a piano) through C5 (one octave higher than middle C). If the user pushes ‘q’ on his keyboard, our program will play a C4, and so on. The number keys (2, 3, 5, 6, 7) play black keys while the letters (q, w, e, r, t, y, u, i) play white keys on the piano.



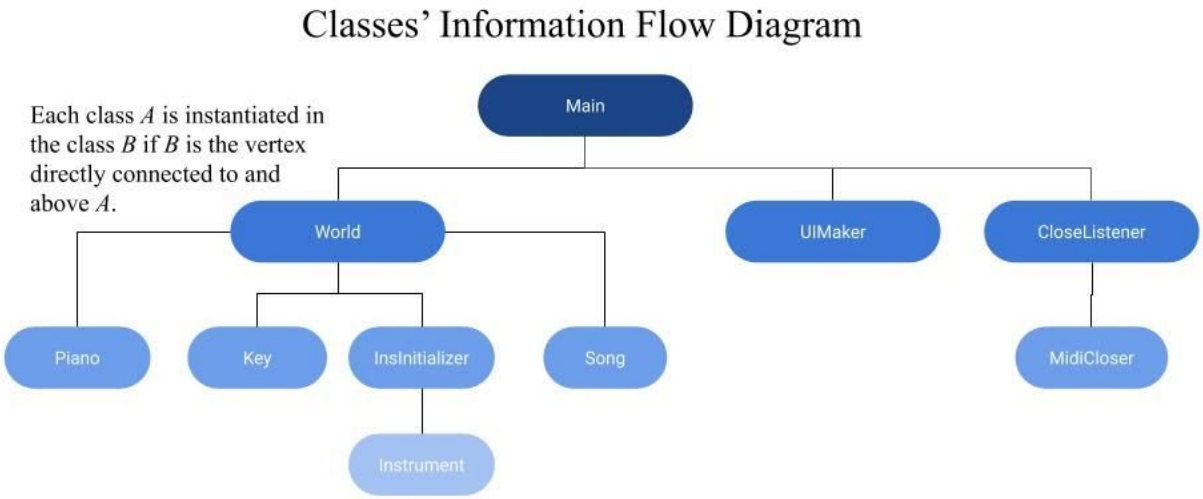
To change octave from the default octave of 4 (which is roughly in the middle of a regular piano), the user can press - or +, or use the left or right arrow keys.

To change the active instrument, the user can use the "Change Instrument" menu in the top left to select a new instrument. For example, bagpipes!



To load a backing track to play along with, the user can open the “Load MIDI File” menu and select a MIDI track from the “Tracks” folder. The user can simply press “Play” to play, and “Pause” to pause the song. Reselect the song in the menu to restart it.

Below is a diagram of how the classes come together and meet in Main.java.



Classes List & Overview

Class Name/ [Nested Class Name]	Access Level	Parents	Description	Number of Instances	Number of Constructors	Constructor Explanation
InsInitializer	public	N/A	This class exists for the sole purpose of being able to very easily change the list of available instruments. Its only method is static, which is why no instances need to be created.	0	0	N/A
Instrument	public	N/A	This class provides a way to keep track of a given instrument; 12 instances for each of our 12 instruments.	12 (number of instruments)	1	The single constructor is pretty standard, being used to assign values to some or all of the class's member variables.
Key	public	N/A	This class allows you to keep track of the basic properties of a given piano key in an octave.	13 (one for each note from a lower-C to a higher-C of an arbitrary octave)	1	The single constructor is pretty standard, being used to assign values to some or all of the class's member variables.
Main	public	extends JPanel implements KeyListener	This class is our Main class! It runs the entire program, drawing together the other classes into something the user can interact with and enjoy. The application is a piano keyboard whose active instrument can be changed; MIDI files can be played back as well.	1	1	Creates a World as a field, and does standard JPanel constructor stuff.
World	public	N/A	Keeps track of much of the basic information that is used in the user interface, drawing, and audio aspects of the program and provides a drawScreen function to be called in Main's paintComponent.	1 (planet Earth is unique!)	1	Sets up much of the basic information that is used in the user interface, drawing, and audio aspects of the program.
Piano	public	N/A	Draws graphics related to on-screen keyboard in response to user input	1 (since only 1 piano needs to be drawn in the program. This is a field of World)	1	width and height fields are set to the main width and height (i.e. width and height of the panel). Otherwise, the constructuor takes in an array of keys representing the keyboard and the desired height of the on-screen keyboard
Song	public	N/A	Handles acquiring and interacting with the local system MIDI devices needed: the synthesizer, which makes the sound, and the sequencer, which reads the MIDI files. Holds these devices as fields of itself. All other classes invoke methods of this class for everything sound or MIDI-related, basically playing piano notes and songs.	1 instance as a field of World	1	Obtains the default local synth and sequencer, sets the latter to transmit events to the former, and sets the piano volume.
UIMaker	public	N/A	UIMaker has all the tools for making the user interface: the program's menus, buttons, and so on.	1 (in Main's "main" method)	1	The single constructor assigns UIMaker's mainInstance to Main's mainInstance and set's UIMaker's world to mainInstance's world.
CloseListener	public	extends WindowAdapter	Code file borrowed from tips4java.wordpress.com, which gives universal permission for the use of its code. It uses a WindowListener to handle program closing operations, and supports a confirmation dialog box and any Action.	1, just after the window initialization and Main instantiation in main()	3	The main constructor takes the Action and two String values for the dialog box; the others take only Action or Strings. (We use the Action-only constructor.)
MidiCloser	public	extends AbstractAction	Closes the MIDI devices via Song. This is the window closing action used by CloseListener.	1 (see above)	1	Takes the Song instance as a field, to later invoke its close() method.
[MyMenuListener]	private	implements MenuListener	Ensures that, when someone clicks the menu, our keyboard keys are not stuck down.	2 (1 for instrument menu, 1 for MIDI file menu)	0	All constructors are default/inherited.

Classes List & Overview

Class Name/ [Nested Class Name]	Access Level	Parents	Description	Number of Instances	Number of Constructors	Constructor Explanation
[MyButton]	private	extends JButton	This class is similar to JButton, but it ensures that each button has the correct font and is not focusable (i.e. the program goes back to focusing on the user's CPU keyboard inputs after a MyButton is clicked).	5 (Help button, 2 octave changing buttons, 2 toggler buttons)	2	Constructor 1 takes no arguments. Be sure to use "setText" later for the button to label the button. Used to allow the program to set the text of the generalized octave button without setting any initial text. Constructor 2 takes in a String that labels the button.
[OctaveButton]	private	extends MyButton	Depending on boolean input to the constructor, an instance of this class is a button labeled either "+" or "-", with location and ActionListener corresponding to the decreasing and increasing of octaves.	2 (decrease, increase)	1	Depending on boolean input, OctaveButton's constructor creates a button labeled either "+" or "-", with location and ActionListener corresponding to the decreasing and increasing of octaves.
[HelperAL]	private	implements ActionListener	This class gives the reaction to the Help button, namely that of opening up the README.txt with instructions and notes about program use.	1	0	All constructors are default/inherited.
[MidiLoader]	private	implements ActionListener	This class gives the program the appropriate actions when the user clicks any of the items on the MIDI file menu. It loads either the specified file or chooses a random file from our Tracks folder (in the case of "Load Random File").	5 (1 for the random track loader, 4 for the 4 MIDI files in "Tracks")	2	The buttons that specify the MIDI file name only take in the file name as a String; the button that picks a random MIDI file to load must take in an ArrayList<String> that represents a list of the possible MIDI files to randomly choose from.
[InsChanger]	private	implements ActionListener	This class instructs the program to change the user's keyboard sound to an indicated instrument when the menu item is pressed.	12 (number of instruments)	1	Takes in an integer, which is assigned to the member variable n. This represents the index in the list of instruments that corresponds to our selected instrument.
[OctaveChanger]	private	implements ActionListener	This class instructs the program to increment the octave by +/-1 when the corresponding button is pressed.	2 (1 for octave increase button, 1 for octave decrease button)	1	Takes in an integer, which is assigned to the member variable n. This represents the amount (+/-1) by which the octave will be changed.
[Toggler]	private	implements ActionListener	This class contains a MyButton as one of its member variables; this class also allows one to Toggle the button and its corresponding variables to true or false, updating the button's displayed text as necessary.	2 (1 for play/pause button, 1 for display key mappings button)	1	Sets the location of the Toggler's MyButton, labels it, and assigns it its corresponding ActionListener.

Classes' Member Variables

Class/[Nested Class]	Variable Name	Description	Type	Type Explanation	Access Level	Access Level Explanation
Main	WIDTH	width of the Main	int	N/A	public static final	Main extends JPanel
	HEIGHT	height of the Main	int	N/A	public static final	Main extends JPanel
	world	Keeps track of much of the basic information that is used in the user interface, drawing, and audio aspects of the program and provides a drawScreen function to be called in Main's paintComponent.	World	N/A	public	world is accessed in the UIMaker's constructor
World	displayMapping	True if displaying "q" on the lower C-key and "w" on the D-key and so on	boolean	N/A	public	Accessed in UIMaker
	playback	True if currently playing back a MIDI file	boolean	N/A	public	Accessed in UIMaker
	loadedFileName	The name of the current loaded MIDI file, if any.	String	N/A	public	Accessed in UIMaker
	height	As received from Main's HEIGHT	int	N/A	public	Accessed in UIMaker
	width	As received from Main's WIDTH	int	N/A	public	Accessed in UIMaker
	background	The image in the background, corresponding to the active instrument.	BufferedImage	N/A	public	Accessed in UIMaker
	mySong	The object that keeps track of the MIDI devices for sound and file reading.	Song	N/A	public	Accessed in UIMaker
	instruments	This allows one to keep track of the available instrument options.	ArrayList<Instrument>	N/A	public	Accessed in UIMaker
	keyboard	This field keeps track of the 13 Key objects in our arbitrary octave.	Key[]	N/A	public	Accessed in Main
	octave	From 0 through 8, this keeps track of the octave. Middle C is played at the leftmost C when the octave is 4, corresponding to a piano's C4.	int	N/A	public	Accessed in UIMaker
	keyBook	This takes in the user's CPU keyboard action's key character, such as q, w, e, 2, 3, etc. and outputs the index in the keyboard of the musical Key that corresponds to the CPU key.	HashMap<Character, Integer>	A switch statement could have been used, but this seemed to be the most intuitive and efficient method of mapping input chars (q, e, etc.) to the index of the Key being played in keyboard.	public	Accessed in Main
	piano	Piano object that is used to tdraw graphics related to the on-screen keyboard	Piano	N/A	private	Only accessed in World
UIMaker	world	Many member variables of world are accessed. This is primarily here to avoid the necessity of typing out "mainInstance.world...." repeatedly.	World	N/A	private	UIMaker's world is only referenced within UIMaker, as Main's world can be referenced elsewhere.
	mainInstance	This enables the UI's buttons and menu bar to be added to the instance of Main, as well as enabling us to use Main's refresh() and repaint() methods as responses to the user's use of the UI.	Main	N/A	private	This particular reference to mainInstance is only used within the class.
[MidiLoader]	rand	This is used for picking a random MIDI file name from an ArrayList of MIDI file names in order to load a random track.	Random	N/A	private	This field is used only in the MidiLoader class.
	fileName	The name of the MIDI file that would be loaded if the corresponding menu item were clicked.	String	N/A	private	This field is used only in the MidiLoader class.

Classes' Member Variables

Class/[Nested Class]	Variable Name	Description	Type	Type Explanation	Access Level	Access Level Explanation
	isRandom	True if this MidiLoader corresponds to the menu item that allows the user to load a random track. This allows the code to streamline the actionPerformed method by checking quickly if the program must do the additional work of fetching a random MIDI file name to be sent to Song's loading functionality.	boolean	N/A	private	This field is used only in the MidiLoader class.
	ls	When choosing a random file, MidiLoader uses to chooses a random file name from this list of names of available MIDI files.	ArrayList<String>	N/A	private	This field is used only in the MidiLoader class.
[InsChanger]	n	Denotes the index of the instrument in world's ArrayList of instruments	int	N/A	private	This field is used only in the InsChanger class.
[OctaveChanger]	n	amount by which world.octave is being incremented when an octave changing button is pressed, namely +/- 1	int	Use of boolean was an option too, but int is simpler	private	This field is used only in the OctaveChanger class.
[Toggler]	b	The MyButton allows one to toggle the button and its corresponding booleans to true or false, updating the button's displayed text as necessary within Toggler's actionPerformed method.	MyButton	See class MyButton for more info.	private	This field is used only in the Toggler class.
	toggledItemName	The toggledItemName is used to check, inside Toggler, whether the button is a Display Key Mappings button or a play/pause button.	String	It is theoretically possible to use a boolean instead, since we have only two buttons. However, the Toggler class was set up to facilitate the addition of further Toggler instances, such as on/off Toggler objects for sound effect pedals (e.g. reverb, delay, echo) or a boolean indicating whether the user is currently recording the music they are playing on the musical keyboard.	private	This field is used only in the Toggler class.
Instrument	instrumentName	The name of the instrument, e.g. "Acoustic Grand Piano" or "FX 6 (goblins)," the latter of which is unfortunately not included in our program.	String	N/A	public final	The name of the instrument is accessed in UIMaker; it is final because instruments do not often change register legal name changes.
	midiNumber	The corresponding MIDI code number of the given instrument.	int	Song's note-channeling methods require an integer to indicate the active instrument.	public final	The corresponding MIDI code number of a given instrument is fixed.
Key	isBlack	True if the corresponding key on the keyboard is supposed to be a black key.	boolean	A piano key is either black or it is not (i.e. it is white).	public final	Used in class Piano. Final because piano keys do not typically change color...not that it wouldn't be cool.
	pressing	True if the key is currently being pressed down. Used in drawing key depressions on the piano keyboard.	boolean	N/A	public	Accessed in Main's KeyListener functionalities to set pressing to true when a Key's corresponding CPU key is being pressed. It is also accessed in World to draw the depressions of each Key that is being pressed down.
Piano	width	Width of the panel, and therefore the width of the keyboard as well	int	Pixels are integers and can't have fractions	private	Isn't accessed outside class, and defined based on another public variable anyway

Classes' Member Variables

Class/[Nested Class]	Variable Name	Description	Type	Type Explanation	Access Level	Access Level Explanation
	height	Height of the panel	int	Pixels are integers and can't have fractions	private	Isn't accessed outside class, and defined based on another public variable anyway
	keyboardheight	Height of the keyboard	int	Pixels are integers and can't have fractions	public	Not currently being accessed outside class, but other graphics may want to know the keyboard height with additional modifications to the program
	keyboard	Array of keys representing the keyboard	Key[]	N/A	private	Only accessed in the class
Song	_synth	The default local MIDI synthesizer's Java address, to which sound messages can be sent	Synthesizer	N/A	public	Just in case more functionality is added, but currently only accessed through Song methods
	_pianoChannel	The synth channel on which the piano is played. The synth has 16 channels, 0-15; this one is 15 so as not to interfere with the sequencer messages.	MidiChannel	N/A	public	Just in case more functionality is added, but currently only accessed through Song methods
	_sequencer	The default local MIDI sequencer's Java address, to which sequences and commands can be sent	Sequencer	N/A	public	Just in case more functionality is added, but currently only accessed through Song methods
	_volume	The value used as the velocity for the piano key sound messages; currently 80	int	N/A	public	Just in case more functionality is added, but currently only accessed through Song methods
	_fileLoaded	Used internally to keep track of whether a sequence is loaded in the sequencer (even though that's probably also handled elsewhere)	boolean	N/A	private	Used internally, in loadFile and play methods, and probably also redundant
CloseListener	message	The message for the (unused) closing dialog box	String	N/A	private	Used only in the dialog box
	title	The title for the (unused) closing dialog box	String	N/A	private	Used only in the dialog box
	closeAction	The action to complete on close	Action	N/A	private	Used only during window closing
	disposeOnClose	Sets whether to use a dispose action rather than an exit action	boolean	N/A	private	Used only internally
MidiCloser	song	The original Song instance, used to close its MIDI device fields	Song	N/A	public	Everybody has access to this anyway

Classes' Methods

Class/[Nested Class]	Method Name (Arguments)	Description (No Recursive Methods)	Access Level Explanation
InsInitializer	public static ArrayList <Instrument> getInsList()	This method creates an ArrayList of instruments; this ArrayList is used to set up the ArrayList of instruments in World.	This method is called in the World class.
Main	public static void main(String [] args)	This method is at the core of the program; it runs when the user opens up our application!	This ensures that the user can actually run our program.
	public void paintComponent(Graphics g)	This does the drawing of things such as the piano graphic and updates the display to show keys when they are pressed down.	The access level is determined because the method is inherited from JPanel.
	public void keyPressed (KeyEvent e)	When a key is pressed, this method runs to channel the corresponding note on the musical keyboard.	The access level is determined because the method is inherited from KeyListener.
	public void keyReleased (KeyEvent e)	When a key is released, this method stops channeling the corresponding note (and redraws the updated corresponding unpressed key on the piano); or, if the left and right arrow keys were pressed, it decreases and increases the octave respectively.	The access level is determined because the method is inherited from KeyListener.
	public void keyTyped (KeyEvent e)	Empty; present to override method from parent.	The access level is determined because the method is inherited from KeyListener.
	public void addNotify()	Calls parent's version of this method and requests focus.	The access level is determined because the method is inherited from KeyListener.
	public void refresh()	Turns of all notes that were played by the user; for each Key instance, sets pressing to false; repaints the screen to display these updates. This method is called when the user clicks certain buttons or opens certain menus that momentarily remove focus from the KeyListener such that keys would otherwise be stuck in a constant state of being pressed down.	This method is called in UIMaker to ensure that, when the user clicks something on the menu bar, the keyboard keys being played at the time do not remain stuck in that state.
Piano	public void initialize (Graphics g)	Draws the default keyboard with nothing pressed and no key mappings, based on the height, width, and keyboardheight fields of the Piano object.	The method is called in drawScreen() in World to draw the initial on-screen keyboard
	public void drawKey (Graphics g, int key, boolean pressed)	This method draws an individual key, either as pressed (and therefore a different color) or unpressed. In practical use, this only needs to be called when a key is pressed because of the previous initialize() method.	This method is called in drawScreen() in World to draw pressed keys

Classes' Methods

Class/[Nested Class]	Method Name (Arguments)	Description (No Recursive Methods)	Access Level Explanation
	public void drawMapping (Graphics g, World w)	This method is called to draw strings representing the cooresponding keys of the computer keyboard on top of each of the on-screen keys. One call of the method draws the mappings on all of the keys.	This method is called in drawScreen() in World to draw computer keyboard mappings on top of the on-screen keyboard
	private void drawMappingHelper(Graphics g, World w, int i, String name, boolean isBlack)	This method is a helper method to simplify drawing mappings on the keys inside the previous drawMapping() method. Thus, it is only called within that method.	Only called within drawMapping(), which is another method within the same class
Song	public void noteOn(int note)	Interfaces to the synth's noteOn command	Called from Main
	public void noteOff(int note)	Interfaces to the synth's noteOff command	Called from Main
	public void allNotesOff()	Interfaces to the synth's allNotesOff command	Called from Main
	public void changeInstrument(int instrument)	Interfaces to the synth's programChange command	Called from InsChanger (in UIMaker)
	public void changeVolume (int volume)	Changes the volume field	Currently unused, but would be called outside Song
	public void loadFile(String path)	Reads a MIDI sequence from the specified MIDI file and loads it into the sequencer	Called from MidiLoader (in UIMaker)
	public void play()	Interfaces to the sequencer's start command, checking first that it's ready to go	Called from Toggler (in UIMaker)
	public void stopPlaying()	Interfaces to the sequencer's stop command	Called from Toggler (in UIMaker)
	public void close()	Closes the synth and sequencer, which is required to deallocate those resources for other programs	Called from MidiCloser
UIMaker	public void makeUI()	Creates and activates the UI on the UIMaker's instance of Main.	This method is called in Main's main method.
	private JMenuBar getMenuBar()	Returns the JMenuBar that is displayed at the top of the program.	This method is called only within UIMaker.
	private JMenu getInstrumentMenu()	Returns the menu from which the user can switch instruments.	This method is called only within UIMaker.
	private JMenu getMIDIMenu()	Returns the menu from which the user can load a MIDI file.	This method is called only within UIMaker.
[MyMenuListener]	public void menuSelected (MenuEvent e)	Ensures that, when someone clicks the menu, our keys are not stuck down.	The access level is determined because the method is inherited from MenuListener.

Classes' Methods

Class/[Nested Class]	Method Name (Arguments)	Description (No Recursive Methods)	Access Level Explanation
	public void menuDeselected(MenuEvent e)	Empty; present to override method from parent.	The access level is determined because the method is inherited from MenuListener.
	public void menuCanceled(MenuEvent e)	Empty; present to override method from parent.	The access level is determined because the method is inherited from MenuListener.
[MidiLoader]	public void actionPerformed(ActionEvent e)	Sends the String containing the path to the given MIDI file to world.mySong's loadFile method, i.e. reacts to the clicking of a MIDI menu item by loading the corresponding MIDI file.	The access level is determined because the method is inherited from ActionListener.
[InsChanger]	public void actionPerformed(ActionEvent e)	Sets the background to correspond to the chosen instrument and changes the instrument, in accordance with the item clicked on the instrument menu.	The access level is determined because the method is inherited from ActionListener.
[OctaveChanger]	public void actionPerformed(ActionEvent e)	Increments the octave by +/- 1 to some octave in range [0, 8] in accordance with the button clicked.	The access level is determined because the method is inherited from ActionListener.
[Toggler]	public void actionPerformed(ActionEvent e)	If the user is toggling the display of key mappings, this method updates the key mapping display. If the user is toggling the playback, this method updates world.playback and either pauses or plays the loaded track.	The access level is determined because the method is inherited from ActionListener.
World	public void drawScreen (World w, Graphics g)	This method is called in Main's paintComponent to draw things in correspondence with the state of our world - e.g., if a key is being pressed down, and so on.	Called in the class Main
	public void drawStringCentered(Graphics g, String s, int x0, int x1, int y0, int y1)	Draws a String centered within given boundaries.	Used in the Piano class as well as in World.
CloseListener	public void setDisposeOnClose(boolean disposeOnClose)	Sets the field used to determine the final closing action	Unused here, but would be called from outside CloseListener
	public void windowClosing (WindowEvent e)	Overrides WindowAdapter's windowClosing method to trigger the dialog box or closing action at the right time	Must be public because its superclass method is public; called from elsewhere in the JFrame workings
	private boolean confirmWindowClosing(JFrame frame)	Actually opens the confirmation dialog box (JOptionPane) and returns the yes/no result	Only accessed from windowClosing

Classes' Methods

Class/[Nested Class]	Method Name (Arguments)	Description (No Recursive Methods)	Access Level Explanation
MidiCloser	public void actionPerformed(ActionEvent e)	Overrides AbstractAction's actionPerformed method to call Song's close method when this action is invoked	Must be public because its superclass method is public; called from CloseListener