Rhythminator Part 3

CSE1102 Project 8, Spring 2016

Bryan Arnold

5/2/16

TA: Zigeng Wang

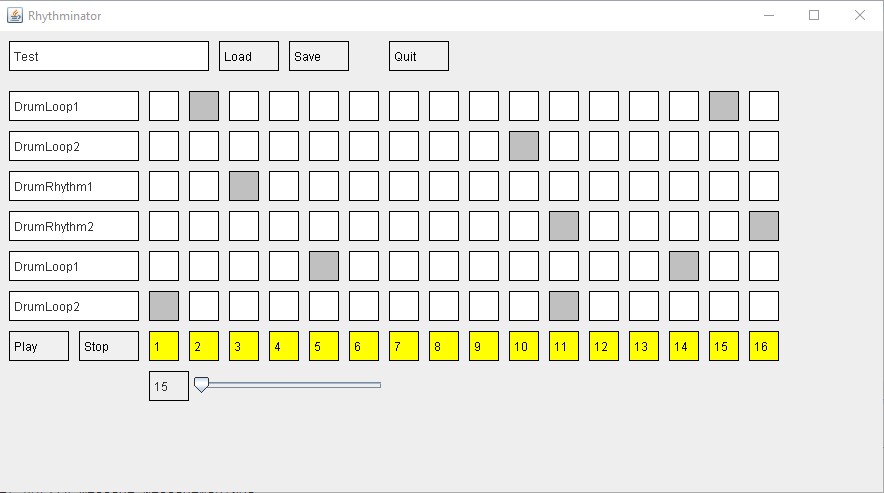
Section: 51

Instructor: Jeffrey A. Meunier

**Introduction:**

In this assignment you will complete the controller in order to connect the view and model together to complete the final part of the Rhythminator Project.

**Output:**

****

Sound.scanSoundDir class found 93 sound files in directory 'sounds'

Sound.scanSoundDir class found 94 sound files in directory 'sounds'

Controller.keyReleased 10

Controller.soundNameSelected for track 1: DrumLoop1

Controller.soundNameSelected for track 2: DrumLoop2

Controller.soundNameSelected for track 3: DrumRhythm1

Controller.soundNameSelected for track 4: DrumRhythm2

Controller.soundNameSelected for track 5: DrumLoop1

Controller.soundNameSelected for track 6: DrumLoop2

Controller.noteSquareClicked NoteSquare(track = 0, beat = 1)

Controller.noteSquareClicked NoteSquare(track = 2, beat = 2)

Controller.noteSquareClicked NoteSquare(track = 1, beat = 9)

Controller.noteSquareClicked NoteSquare(track = 0, beat = 14)

Controller.noteSquareClicked NoteSquare(track = 4, beat = 13)

Controller.noteSquareClicked NoteSquare(track = 5, beat = 10)

Controller.noteSquareClicked NoteSquare(track = 4, beat = 4)

Controller.noteSquareClicked NoteSquare(track = 5, beat = 0)

Controller.noteSquareClicked NoteSquare(track = 3, beat = 10)

Controller.noteSquareClicked NoteSquare(track = 3, beat = 15)

Controller.sliderChange null = 25

Controller.sliderChange null = 26

Controller.sliderChange null = 27

Controller.sliderChange null = 28

Controller.sliderChange null = 29

Controller.sliderChange null = 30

Controller.sliderChange null = 29

Controller.sliderChange null = 28

Controller.sliderChange null = 27

Controller.sliderChange null = 26

Controller.sliderChange null = 25

Controller.sliderChange null = 24

Controller.sliderChange null = 23

Controller.sliderChange null = 22

Controller.sliderChange null = 21

Controller.sliderChange null = 20

Controller.sliderChange null = 19

Controller.sliderChange null = 18

Controller.sliderChange null = 17

Controller.sliderChange null = 16

Controller.sliderChange null = 15

Controller.sliderChange null = 15

Save Button Pressed

Rhythm name: Test

Tempo: 15

Load Button Pressed

Controller.buttonPressed got Play button

Beat: 1

Controller.notify message.Message@a016da

Beat: 2

Controller.notify message.Message@a016da

Beat: 3

Controller.notify message.Message@a016da

Beat: 4

Controller.notify message.Message@a016da

Beat: 5

Controller.notify message.Message@a016da

Beat: 6

Controller.notify message.Message@a016da

Beat: 7

Controller.notify message.Message@a016da

Beat: 8

Controller.notify message.Message@a016da

Beat: 9

Controller.notify message.Message@a016da

Beat: 10

Controller.notify message.Message@a016da

Beat: 11

Controller.notify message.Message@a016da

Beat: 12

Controller.notify message.Message@a016da

Beat: 13

Controller.notify message.Message@a016da

Beat: 14

Controller.notify message.Message@a016da

Beat: 15

Controller.notify message.Message@a016da

Beat: 16

Controller.notify message.Message@a016da

Beat: 1

Controller.notify message.Message@a016da

Controller.buttonPressed got Stop button

**Source Code:**

**Default Package:**

**Rythminator Class:**

**import** javax.swing.SwingUtilities;

**import** model.Clock;

**import** model.Model;

**import** model.Sound;

**import** view.View;

**import** controller.Controller;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

**public** **class** Rhythminator{

**private** **static** **final** **int** ***NUM\_TRACKS*** = 6;

**private** **static** **final** **int** ***NUM\_BEATS*** = 16;

**public** **static** **void** main(String[] args){

Sound.*scanSoundDir*("sounds");

Sound s1 = **new** Sound("CymbalCrash1");

**for**(**int** n=0; n<4; n++){

s1.play();

**try**

{

Thread.*sleep*(250);

}

**catch**(InterruptedException e){

e.printStackTrace();

}

}

// In order to use Swing graphics effectively, the program should be started

// running in a concurrent thread controlled by Swing. This is how you do it:

SwingUtilities.*invokeLater*(**new** Runnable(){

@Override

**public** **void** run(){

Rhythminator.*\_main*();

}

});

}

// I put all the interesting stuff in a separate method here so that you don't

// have to keep looking at the Swing threading stuff in the main method above.

**private** **static** **void** \_main(){

Sound.*scanSoundDir*();

Model model = **new** Model(***NUM\_TRACKS***, ***NUM\_BEATS***);

Clock clock = model.getClock();

Controller controller = **new** Controller(clock, model);

View view = **new** View(controller, ***NUM\_TRACKS***, ***NUM\_BEATS***);

controller.setView(view);

}

}

**Controller Package:**

**Controller Class:**

**package** controller;

**import** gui.Button;

**import** gui.Dialog;

**import** message.ISubscriber;

**import** message.Message;

**import** model.Clock;

**import** model.Model;

**import** view.NoteSquare;

**import** view.View;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

**public** **class** Controller **implements** ISubscriber

{

**private** Model \_model;

**private** Clock \_clock;

**private** View \_view;

**public** Controller(Clock clock, Model model)

{

**this**.\_model = model;

**this**.\_clock = clock;

clock.subscribe(**this**);

}

**public** **void** buttonPressed(Button button)

{

**if**(button.getText().equals("Save"))

\_buttonSave();

**else** **if**(button.getText().equals("Load"))

\_buttonLoad();

**else** **if**(button.getText().equals("Play"))

{

**this**.\_model.setNote(0, 1, **false**);

**this**.\_model.startPlaying();

System.***out***.println("Controller.buttonPressed got Play button");

}

**else** **if**(button.getText().equals("Stop"))

{

**this**.\_model.stopPlaying();

**this**.\_view.clearBeatNumbers();

System.***out***.println("Controller.buttonPressed got Stop button");

}

**else** **if**(button.getText().equals("Quit"))

\_buttonQuit();

**else**

System.***out***.println("Controller.buttonPressed " + button + " pressed");

}

**private** **void** \_buttonQuit()

{

**if**(Dialog.*askYesNo*("Exiting program", "Really quit?"))

System.*exit*(0);

}

**private** **void** \_buttonLoad()

{

System.***out***.println("Load Button Pressed");

}

**private** **void** \_buttonSave()

{

System.***out***.println("Save Button Pressed");

System.***out***.println("Rhythm name: " + **this**.\_view.getHead().getFile());

**long** tempo = 15000/(**this**.\_model.getClock().getDelay());

System.***out***.println("Tempo: " + tempo);

}

**public** **void** keyPressed(**int** keyCode)

{

System.***out***.println("Controller.keyPressed " + keyCode);

}

**public** **void** keyReleased(**int** keyCode)

{

System.***out***.println("Controller.keyReleased " + keyCode);

}

**public** **void** keyTyped(**char** keyChar)

{

System.***out***.println("Controller.keyTyped '" + keyChar + "'");

}

**public** **void** noteSquareClicked(NoteSquare noteSquare) {

**int** trackNum = noteSquare.getTrack();

**int** beatNum = noteSquare.getBeat();

**int** value = noteSquare.getValue();

**if**(value == 0) {

\_model.setNote(trackNum, beatNum, **false**);

} **else** {

\_model.setNote(trackNum, beatNum, **true**);

}

System.***out***.println("Controller.noteSquareClicked " + noteSquare);

}

@Override

**public** **void** notify(Message message)

{

**this**.\_view.setBeatNumber(**this**.\_model.getBeatNumber() - 1);

System.***out***.println("Beat: " + **this**.\_model.getBeatNumber());

System.***out***.println("Controller.notify " + message);

}

**public** **void** soundNameSelected(**int** trackNumber, String soundName)

{

**this**.\_model.setSoundName(trackNumber, soundName);

**this**.\_view.setSoundName(trackNumber, soundName);

System.***out***.println("Controller.soundNameSelected for track " + trackNumber + ": " + soundName);

}

**public** **void** setView(View view)

{

\_view = view;

}

**public** **void** sliderChange(String name, **int** \_value)

{

**this**.\_clock.setDelay(15000/\_value);

System.***out***.println("Controller.sliderChange " + name + " = " + \_value);

}

}

**GUI Package:**

**Box Class:**

**package** gui;

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.awt.event.MouseEvent;

**import** java.awt.event.MouseListener;

**import** javax.swing.BorderFactory;

**import** javax.swing.JPanel;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

**public** **class** Box **extends** JPanel **implements** MouseListener{

**private** String \_text = "";

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** Box(){

**this**("");

}

**public** Box(String text){

**this**.setBorder(BorderFactory.*createLineBorder*(Color.***black***));

**this**.addMouseListener(**this**);

\_text = text;

}

**public** String getText(){

**return** \_text;

}

**public** **void** paintComponent(Graphics g){

**super**.paintComponent(g);

g.drawString(\_text, 5, 20);

}

@Override

**public** **void** mouseClicked(MouseEvent mev){}

@Override

**public** **void** mouseEntered(MouseEvent mev){}

@Override

**public** **void** mouseExited(MouseEvent mev){}

@Override

**public** **void** mousePressed(MouseEvent mev)

{}

@Override

**public** **void** mouseReleased(MouseEvent mev){}

**public** **void** setText(String text){

\_text = text;

**this**.repaint();

}

@Override

**public** String toString(){

**return** "Square(" + \_text + ")";

}

}

**Button Class:**

**package** gui;

**import** java.awt.event.MouseEvent;

**import** controller.Controller;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

**public** **class** Button **extends** Box{

**private** Controller \_controller;

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** Button(Controller controller, String text){

**super**();

\_controller = controller;

**this**.setText(text);

**this**.setForeground(Colors.***BUTTON\_FG***);

**this**.setBackground(Colors.***BUTTON\_BG***);

}

@Override

**public** **void** mousePressed(MouseEvent mev){

\_controller.buttonPressed(**this**);

}

@Override

**public** String toString(){

**return** "Button(" + getText() + ")";

}

}

**Colors Class:**

**package** gui;

**import** java.awt.Color;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

/\*\*

\* These are color definitions for the GUI elements.

\* **@author** jeff

\*/

**public** **class** Colors {

**public** **static** **final** Color ***BUTTON\_FG*** = Color.***BLACK***;

**public** **static** **final** Color ***BUTTON\_BG*** = **new** Color(240, 240, 240);

**public** **static** **final** Color ***WINDOW\_BG*** = Color.***WHITE***;

}

**Dialog Class:**

**package** gui;

**import** javax.swing.JDialog;

**import** javax.swing.JOptionPane;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

**public** **class** Dialog{

**public** **static** **boolean** askYesNo(String title, String question){

**final** String YES = "Yes";

**final** String NO = "No";

JOptionPane pane = **new** JOptionPane(question);

Object[] options = **new** String[] {NO, YES};

pane.setOptions(options);

JDialog dialog = pane.createDialog(**null**, title);

dialog.setVisible(**true**);

Object obj = pane.getValue();

**if**(obj != **null** && obj.toString().equals(YES))

**return** **true**;

**return** **false**;

}

**public** **static** String getString(String title, String prompt, String initialValue){

Object value = JOptionPane.*showInputDialog*(**null**, prompt, title, JOptionPane.***PLAIN\_MESSAGE***, **null**, **null**, initialValue);

**if**(value != **null**)

**return** value.toString();

**return** **null**;

}

**public** **static** String selectOption(String title, String prompt, String[] options){

Object value = JOptionPane.*showInputDialog*(**null**, prompt, title, JOptionPane.***PLAIN\_MESSAGE***, **null**, options, **null**);

**if**(value != **null**)

**return** value.toString();

**return** **null**;

}

}

**NumberSlider Class:**

**package** gui;

**import** javax.swing.JPanel;

**import** javax.swing.JSlider;

**import** javax.swing.event.ChangeEvent;

**import** javax.swing.event.ChangeListener;

**import** controller.Controller;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

**public** **class** NumberSlider **extends** JPanel **implements** ChangeListener{

**private** Controller \_controller;

**private** String \_name;

**private** JSlider \_slider;

**private** Box \_numberBox;

**private** **int** \_value;

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** NumberSlider(Controller controller, String name, **int** low, **int** high, **int** deflt){

**this**.setLayout(**null**);

\_name = name;

\_numberBox = **new** Box();

\_numberBox.setLocation(0, 0);

\_numberBox.setSize(40, 30);

\_numberBox.setText("" + deflt);

**this**.add(\_numberBox);

\_slider = **new** JSlider(JSlider.***HORIZONTAL***, low, high, deflt);

\_slider.setFocusable(**false**);

\_slider.addChangeListener(**this**);

\_slider.setLocation(\_numberBox.getWidth(), 0);

\_slider.setSize(200, 30);

**this**.add(\_slider);

\_value = deflt;

\_controller = controller;

**this**.setSize(\_slider.getX() + \_slider.getWidth(), 100);

}

@Override

**public** **void** stateChanged(ChangeEvent ce){

JSlider slider = (JSlider)ce.getSource();

\_value = slider.getValue();

\_numberBox.setText("" + \_value);

\_controller.sliderChange(\_name, \_value);

}

}

**TextBox Class:**

**package** gui;

**import** java.awt.event.MouseEvent;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

**public** **class** TextBox **extends** Box{

**private** String \_dialogTitle;

**private** String \_dialogPrompt;

**private** **static** **final** **long** ***serialVersionUID*** = 1;

**public** TextBox(){

**this**("Click to enter text");

}

**public** TextBox(String text){

**this**(text, "Enter new text");

}

**public** TextBox(String text, String dialogPrompt){

**this**(text, dialogPrompt, "Question");

}

**public** TextBox(String text, String dialogPrompt, String dialogTitle){

**super**(text);

\_dialogPrompt = dialogPrompt;

\_dialogTitle = dialogTitle;

}

@Override

**public** **void** mousePressed(MouseEvent mev){

String initialValue = **this**.getText();

String text = Dialog.*getString*(\_dialogTitle, \_dialogPrompt, initialValue);

**if**(text != **null**)

setText(text);

}

}

**Window Class:**

**package** gui;

**import** java.awt.event.KeyEvent;

**import** java.awt.event.KeyListener;

**import** java.awt.event.WindowEvent;

**import** java.awt.event.WindowListener;

**import** javax.swing.JFrame;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

**public** **class** Window **extends** JFrame **implements** KeyListener, WindowListener{

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** Window(String name){

**super**(name);

**this**.setLayout(**null**);

**this**.addWindowListener(**this**);

**this**.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

**this**.addKeyListener(**this**);

**this**.setFocusable(**true**); // allows key listener to capture events on this frame

**this**.setSize(400, 400);

**this**.setBackground(Colors.***WINDOW\_BG***);

}

/\*\*

\* Call this method to make the application window full-screen.

\* This hides the title bar.

\*/

**public** **void** fullscreen(){

**this**.maximize();

**this**.setUndecorated(**true**);

}

@Override

**public** **void** keyPressed(KeyEvent kev){}

@Override

**public** **void** keyReleased(KeyEvent kev){}

@Override

**public** **void** keyTyped(KeyEvent kev){}

/\*\*

\* Call this method to maximize the application window.

\* It's different from full-screen.

\*/

**public** **void** maximize(){

**this**.setSize(-1, -1);

}

@Override

**public** **void** windowActivated(**final** WindowEvent wev){}

@Override

**public** **void** windowClosed(**final** WindowEvent wev){}

@Override

**public** **void** windowClosing(**final** WindowEvent wev){

//System.exit(0); // not needed if JFrame.EXIT\_ON\_CLOSE is used in the constructor

}

@Override

**public** **void** windowDeactivated(**final** WindowEvent wev){}

@Override

**public** **void** windowDeiconified(**final** WindowEvent wev){}

@Override

**public** **void** windowIconified(**final** WindowEvent wev){}

@Override

**public** **void** windowOpened(**final** WindowEvent wev){}

}

**Message Package:**

**Broadcaster Class:**

**package** message;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

/\*\*

\* Broadcasts a message to multiple subscribers.

\*/

**public** **class** Broadcaster **implements** IPublisher, ISubscriber{

**private** ISubscriber[] \_subscribers;

**private** **int** \_nextSubscriber = 0;

/\*\*

\* Creates an instance of a Broadcaster that can hold up to some number of

\* subscribers.

\* **@param** numSubscribers The maximum number of subscribers that this

\* Broadcaster can hold.

\*/

**public** Broadcaster(**int** numSubscribers){

\_subscribers = **new** ISubscriber[numSubscribers];

}

@Override

**public** **void** notify(Message message){

**for**(ISubscriber subscriber : \_subscribers)

**if**(subscriber != **null**)

subscriber.notify(message);

}

@Override

**public** **void** subscribe(ISubscriber subscriber){

\_subscribers[\_nextSubscriber++] = subscriber;

}

@Override

**public** **void** unsubscribe(ISubscriber subscriber){}

}

**IPublisher Interface:**

**package** message;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

**public** **interface** IPublisher {

**public** **void** subscribe(ISubscriber subscriber);

**public** **void** unsubscribe(ISubscriber subscriber);

}

**ISubscriber Interface:**

**package** message;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

**public** **interface** ISubscriber{

**public** **void** notify(Message message);

}

**Message Class:**

**package** message;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

**public** **class** Message{

**private** IPublisher \_publisher;

**public** Message(IPublisher publisher){

\_publisher = publisher;

}

**public** IPublisher getPublisher(){

**return** \_publisher;

}

}

**Sequencer Class:**

**package** message;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

**public** **class** Sequencer **implements** IPublisher, ISubscriber{

**private** SubscriberList \_subscribers;

**private** **int** \_nextNotify = 0;

**public** Sequencer(**int** numSubscribers){

\_subscribers = **new** SubscriberList();

}

**public** **int** getStepNumber(){

**return** \_nextNotify;

}

@Override

**public** **void** notify(Message message){

**if**(\_nextNotify == \_subscribers.count())

\_nextNotify = 0;

SequencerMessage msg = **new** SequencerMessage(**this**, \_nextNotify);

//System.out.println("Sequencer.notify " + \_nextNotify);

\_subscribers.notify(\_nextNotify++, msg);

}

**public** **void** reset(){

\_nextNotify = 0;

}

**public** **void** setStep(**int** stepNumber, ISubscriber subscriber){

\_subscribers.subscribe(subscriber);

}

@Override

**public** **void** subscribe(ISubscriber subscriber){

\_subscribers.subscribe(subscriber);

}

@Override

**public** **void** unsubscribe(ISubscriber subscriber){

\_subscribers.unsubscribe(subscriber);

}

}

**SequencerMessage Class:**

**package** message;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

**public** **class** SequencerMessage **extends** Message{

**private** **int** \_stepNumber;

**public** SequencerMessage(IPublisher publisher, **int** stepNumber){

**super**(publisher);

\_stepNumber = stepNumber;

}

**public** **int** getStepNumber(){

**return** \_stepNumber;

}

}

**SubscriberList Class:**

**package** message;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

**import** java.util.ArrayList;

/\*\*

\* This is a convenient class to use to keep track of a list of subscribers. The list

\* of subscribers can grow without bound, so this is easier to use than an array.

\* **@author** jeff

\*/

**public** **class** SubscriberList **implements** IPublisher, ISubscriber{

**private** ArrayList<ISubscriber> \_subscribers = **new** ArrayList<ISubscriber>();

**public** SubscriberList(){

// nothing to see here

}

**public** **int** count(){

**return** \_subscribers.size();

}

@Override

**public** **void** notify(Message message){

**for**(ISubscriber subscriber : \_subscribers)

**if**(subscriber != **null**)

subscriber.notify(message);

}

**public** **void** notify(**int** subscriberNumber, Message message){

ISubscriber subscriber = \_subscribers.get(subscriberNumber);

**if**(subscriber != **null**)

subscriber.notify(message);

}

@Override

**public** **void** subscribe(ISubscriber subscriber){

\_subscribers.add(subscriber);

}

@Override

**public** **void** unsubscribe(ISubscriber subscriber){

\_subscribers.remove(subscriber);

}

}

**Model Package:**

**Chord Class:**

**package** model;

**import** message.ISubscriber;

**import** message.Message;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

**public** **class** Chord **implements** ISubscriber{

**private** SoundBank \_soundBank;

**private** **boolean**[] \_notes;

**public** Chord(SoundBank soundBank, **int** numNotes){

\_soundBank = soundBank;

\_notes = **new** **boolean**[numNotes];

}

@Override

**public** **void** notify(Message message){

**this**.play();

}

**public** **void** play(){

\_soundBank.play(\_notes);

}

**public** **void** setNote(**int** noteNumber, **boolean** noteValue){

\_notes[noteNumber] = noteValue;

}

}

**Clock Class:**

**package** model;

**import** java.util.Timer;

**import** java.util.TimerTask;

**import** message.IPublisher;

**import** message.ISubscriber;

**import** message.Message;

**import** message.SubscriberList;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

/\*\*

\* A Clock instance is a free-running asynchronous timer. It conforms to the

\* IPublisher interface.

\* **@author** jeff

\*/

**public** **class** Clock **extends** TimerTask **implements** IPublisher{

**private** SubscriberList \_subscribers = **new** SubscriberList();

**private** Message \_msg = **new** Message(**this**); // create one message to keep re-using

**private** Timer \_timer;

**private** **long** \_delay = 250;

**private** **int** \_numTicksLeft = 0;

**public** Clock(){

// nothing to see here

}

/\*\*

\* This method is called automatically by the timer at each clock tick.

\* If you're thinking about calling this method, don't. Call tick() instead.

\* (Yes, it's like having two doorways that lead into the same room. This

\* doorway is not for you.)

\*/

@Override

**public** **void** run(){

tick();

}

/\*\*

\* Sets the delay of this clock. By default, the clock is set to tick

\* once every 250 mS (which is every 1/4 second).

\* **@param** delay The number of milliseconds to delay between ticks.

\*/

**public** **void** setDelay(**long** delay){

\_delay = delay;

}

**public** **long** getDelay(){

**return** **this**.\_delay;

}

/\*\*

\* Starts the clock running for an unbounded number of ticks.

\* To stop the clock, call the stop() method.

\*/

**public** **void** start(){

**this**.start(-1);

}

/\*\*

\* Starts the clock running for a specific number of ticks.

\* **@param** numTicks

\*/

**public** **void** start(**int** numTicks){

**if**(\_timer == **null**){

\_numTicksLeft = numTicks;

\_timer = **new** Timer();

ClockTask clockTask = **new** ClockTask();

\_timer.schedule(clockTask, 0, \_delay);

}

}

/\*\*

\* Stops a running clock.

\*/

**public** **void** stop(){

**if**(\_timer != **null**){

\_timer.cancel();

\_timer = **null**;

}

}

/\*\*

\* Adds a subscriber to the list of subscribers for this clock.

\* Each time the clock ticks, all subscribers will be notified of the clock tick.

\* **@param** subscriber

\*/

**public** **void** subscribe(ISubscriber subscriber){

\_subscribers.subscribe(subscriber);

}

/\*\*

\* This method is called by the run() method.

\* You may call this method to test or simulate a single clock tick.

\*/

**public** **void** tick(){

**if**(\_numTicksLeft == 0)

**this**.stop();

**else**

{

// you can un-comment this next line if you want to see when the clock ticks

//System.out.println("Clock.tick(" + \_numTicksLeft + ")");

\_subscribers.notify(\_msg);

**if**(\_numTicksLeft > 0)

\_numTicksLeft--;

}

}

/\*\*

\* Removes a subscriber from the list of subscribers for this clock.

\* **@param** subscriber

\*/

**public** **void** unsubscribe(ISubscriber subscriber){

\_subscribers.unsubscribe(subscriber);

}

/\*\*

\* Every time the timer starts, a new instance of a TimerTask must be given to it.

\* This is a total pain in the rear. So here's a class that I use for that purpose.

\* This is a good example of why you'd want to put a class within a class. This

\* class is used \*only\* by the Clock class, and furthermore, users of the Clock

\* class should not even know that this auxiliary class is needed to make the Clock

\* work. Thankfully, Java lets me hide the class here inside the Clock class.

\* **@author** jeff

\*/

**private** **class** ClockTask **extends** TimerTask{

@Override

**public** **void** run(){

Clock.**this**.tick();

}

}

}

**Model Class:**

**package** model;

**import** message.ISubscriber;

**import** message.Sequencer;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

@SuppressWarnings("unused")

**public** **class** Model{

**public** **int** \_numTracks;

**public** **int** \_numBeats;

**private** SoundBank \_soundBank;

**private** Sequencer \_sequencer;

**private** Chord[] \_chords;

**private** Clock \_clock;

/\*\*

\* The model sets up the clock.

\* Most of the work of running the application happens in the clock.

\* **@param** numTracks

\* **@param** numBeats

\*/

**public** Model(**int** numTracks, **int** numBeats)

{

**this**.\_numTracks = numTracks;

**this**.\_numBeats = numBeats;

Sequencer sequencer = **new** Sequencer(numBeats);

**this**.\_sequencer = sequencer;

SoundBank soundbank = **new** SoundBank(**new** Sound[numTracks]);

**this**.\_soundBank = soundbank;

Chord[] chords = **new** Chord[numBeats];

**this**.\_chords = chords;

**for**(**int** i = 0; i < **this**.\_chords.length; i++){

chords[i] = **new** Chord(soundbank, numTracks);

sequencer.subscribe(chords[i]);

}

Clock clock = **new** Clock();

**this**.\_clock = clock;

clock.subscribe(sequencer);

}

**public** **int** getBeatNumber()

{

**return** \_sequencer.getStepNumber();

}

**public** Clock getClock()

{

**return** \_clock;

}

**public** **int** getTracks(){

**return** **this**.\_numTracks;

}

**public** **void** startPlaying()

{

**this**.\_clock.start();

}

**public** **void** stopPlaying()

{

**this**.\_clock.stop();

**this**.\_sequencer.reset();

}

**public** **void** setNote(**int** trackNum, **int** beatNum, **boolean** value)

{

**this**.\_chords[beatNum].setNote(trackNum, value);

}

**public** **void** setSoundName(**int** trackNum, String soundName)

{

Sound sound1 = **new** Sound(soundName);

**this**.\_soundBank.setSound(trackNum - 1, sound1);

}

**public** SoundBank getSoundBank(){

**return** **this**.\_soundBank;

}

}

**Sound Class:**

**package** model;

**import** java.io.File;

**import** java.io.IOException;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.Hashtable;

**import** java.util.Set;

**import** javax.sound.sampled.AudioInputStream;

**import** javax.sound.sampled.AudioSystem;

**import** javax.sound.sampled.Clip;

**import** javax.sound.sampled.LineUnavailableException;

**import** javax.sound.sampled.UnsupportedAudioFileException;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

/\*\*

\* This class represents sounds that can be played. The sounds are stored

\* in sound files on the disk, but get loaded into memory automatically.

\* **@author** jeff

\*/

**public** **class** Sound{

**private** String \_name;

**private** Clip \_clip;

**public** **static** **final** String ***SOUND\_DIR*** = "sounds";

**private** **static** Hashtable<String, File> *\_soundFiles* = **new** Hashtable<String, File>();

**private** **static** String[] *\_extensions* = {".aiff", ".wav"};

/\*\*

\* Creates a new sound from a sound file on disk. The sound file must be present

\* in the sounds directory.

\* **@param** soundName The base of the file name from which to create the sound.

\*/

**public** Sound(String soundName){

**this**(soundName, *\_soundFiles*.get(soundName));

}

/\*\*

\* Creates a new sound from a sound file. The file can be located anywhere on disk.

\* **@param** soundName What to call the sound.

\* **@param** soundFile The file that refers to the sound file on disk.

\*/

**public** Sound(String soundName, File soundFile){

**if**(soundFile == **null**)

**throw** **new** RuntimeException("there is no sound file having name " + soundName);

\_name = soundFile.getName();

**try**{

\_clip = AudioSystem.*getClip*();

AudioInputStream ais = AudioSystem.*getAudioInputStream*(soundFile);

\_clip.open(ais);

}

**catch**(LineUnavailableException exn){

**throw** **new** RuntimeException("Unable to open the line for audio output. Is another program using it?");

}

**catch**(UnsupportedAudioFileException exn){

System.***out***.println("Unsupported audio format for that file: " + soundFile.toString());

**throw** **new** RuntimeException("Unsupported audio format for that file: " + soundFile.toString());

}

**catch**(IOException exn){

System.***out***.println("Unable to open file: " + soundFile.toString());

**throw** **new** RuntimeException("Unable to open file: " + soundFile.toString());

}

}

/\*\*

\*

\* **@return** Array of all sound files that were found in the sound directory.

\*/

**public** **static** String[] getAllSoundFileNames(){

Set<String> nameSet = *\_soundFiles*.keySet();

String[] names = **new** String[nameSet.size()];

nameSet.toArray(names);

Arrays.*sort*(names);

**return** names;

}

/\*\*

\*

\* **@return** The name of this sound instance.

\*/

**public** String getName(){

**return** \_name;

}

/\*\*

\* Plays the sound. Make sure your speakers are turned up.

\*/

**public** **void** play(){

\_clip.setFramePosition(0); // necessary for re-playing an already-played clip

\_clip.start();

}

/\*\*

\* Scans the default sound directory for sound files.

\*/

**public** **static** **void** scanSoundDir(){

*scanSoundDir*(***SOUND\_DIR***);

}

/\*\*

\* Scans the specified sound directory for sound files.

\* **@param** dirName

\*/

**public** **static** **void** scanSoundDir(String dirName)

{

ArrayList<String> soundFileNames = **new** ArrayList<String>();

File directory = **new** File(dirName);

File[] files = directory.listFiles();

**for**(File file : files){

String fileName = file.getName();

**for**(String ext : *\_extensions*){

**if**(fileName.endsWith(ext)){

soundFileNames.add(fileName);

String shortName = file.getName();

shortName = shortName.substring(0, shortName.lastIndexOf('.'));

*\_soundFiles*.put(shortName, file);

**break**;

}

}

}

**int** count = *\_soundFiles*.size();

System.***out***.println("Sound.scanSoundDir class found " + count + " sound files in directory '" + ***SOUND\_DIR*** + "'");

*\_soundFiles*.put("(none)", **new** File(""));

}

@Override

**public** String toString(){

**return** "Sound(" + \_name + ")";

}

}

**SoundBank Class:**

**package** model;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

**public** **class** SoundBank{

**private** Sound[] \_sounds;

**public** SoundBank(Sound[] sounds){

\_sounds = sounds;

}

**public** **void** play(**boolean**[] noteValues){

**for**(**int** n=0; n<noteValues.length; n++)

**if**(noteValues[n]){

Sound sound = \_sounds[n];

**if**(sound != **null**)

\_sounds[n].play();

}

}

**public** **void** play(**int** soundNumber){

\_sounds[soundNumber].play();

}

**public** **void** setSound(**int** soundNumber, Sound sound){

\_sounds[soundNumber] = sound;

}

**public** Sound[] getSound(){

**return** **this**.\_sounds;

}

}

**SoundPlayer Class:**

**package** model;

**import** message.ISubscriber;

**import** message.Message;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

**public** **class** SoundPlayer **implements** ISubscriber{

**private** SoundBank \_soundBank;

**private** **int** \_soundNumber;

**public** SoundPlayer(SoundBank soundBank, **int** soundNumber){

\_soundBank = soundBank;

\_soundNumber = soundNumber;

}

@Override

**public** **void** notify(Message message){

\_soundBank.play(\_soundNumber);

}

}

**View Package:**

**BeatNumbers Class:**

**package** view;

**import** javax.swing.JPanel;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

**public** **class** BeatNumbers **extends** JPanel {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**private** BeatNumberSquare[] \_beats;

**public** BeatNumbers(**int** beat){

**this**.setLayout(**null**);

**this**.\_beats = **new** BeatNumberSquare[beat];

**int** n = 0;

**for**(**int** i = 0; i < beat; i++){

BeatNumberSquare beats = **new** BeatNumberSquare(i + 1);

**this**.add(beats);

\_beats[i] = beats;

beats.setLocation(n, 0);

**if** (i < beat - 1){

n += NoteSquare.***SIZE*** + Tracks.***GAP\_SIZE***;

}

}

**this**.setSize((beat \* NoteSquare.***SIZE***) + (beat \* Tracks.***GAP\_SIZE***), NoteSquare.***SIZE***);

}

**public** **void** setBeatNumber(**int** beat){

**for**(**int** i = 0; i < **this**.\_beats.length; i++){

**if**(\_beats[i].getState() == **true**){

\_beats[i].setState(**false**);

}

}

\_beats[beat].setState(**true**);

}

**public** **void** clear(){

**for**(**int** i = 0; i < **this**.\_beats.length; i++){

**if**(\_beats[i].getState() == **true**){

\_beats[i].setState(**false**);

}

}

}

}

**BeatNumberSquare Class:**

**package** view;

**import** gui.Box;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

**public** **class** BeatNumberSquare **extends** Box {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**private** **int** \_beat;

**private** **boolean** \_state;

**public** BeatNumberSquare(**int** beat){

**super**();

**this**.\_beat = beat;

**this**.\_state = **false**;

setText(Integer.*toString*(beat));

setForeground(Colors.***BEATNUMBER\_OFF\_FG***);

setBackground(Colors.***BEATNUMBER\_OFF\_BG***);

setSize(NoteSquare.***SIZE***, NoteSquare.***SIZE***);

}

**public** **boolean** getState(){

**return** **this**.\_state;

}

**public** **int** getBeat(){

**return** **this**.\_beat;

}

**public** **void** setState(**boolean** state) {

**this**.\_state = state;

**if**(**this**.\_state == **true**){

setForeground(Colors.***BEATNUMBER\_ON\_FG***);

setBackground(Colors.***BEATNUMBER\_ON\_BG***);

} **else** **if**(**this**.\_state == **false**){

setForeground(Colors.***BEATNUMBER\_OFF\_FG***);

setBackground(Colors.***BEATNUMBER\_OFF\_BG***);

}

invalidate();

}

}

**Colors Class:**

**package** view;

**import** java.awt.Color;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

/\*\*

\* These are the color definition for the view elements.

\* **@author** jeff

\*/

**public** **class** Colors{

**public** **static** **final** Color ***BEATNUMBER\_OFF\_FG*** = Color.***BLACK***;

**public** **static** **final** Color ***BEATNUMBER\_OFF\_BG*** = Color.***YELLOW***;

**public** **static** **final** Color ***BEATNUMBER\_ON\_FG*** = Color.***BLACK***;

**public** **static** **final** Color ***BEATNUMBER\_ON\_BG*** = Color.***GREEN***;

**public** **static** **final** Color ***NOTESQUARE\_OFF*** = Color.***WHITE***;

**public** **static** **final** Color ***NOTESQUARE\_ON*** = Color.***LIGHT\_GRAY***;

}

**ControlButtons Class:**

**package** view;

**import** javax.swing.JPanel;

**import** controller.Controller;

**import** gui.Button;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

**public** **class** ControlButtons **extends** JPanel {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** ControlButtons(Controller controller){

setLayout(**null**);

Button play = **new** Button(controller, "Play");

Button stop = **new** Button(controller, "Stop");

add(play);

add(stop);

play.setSize(60, 30);

stop.setSize(60, 30);

play.setLocation(0, 0);

stop.setLocation(70, 0);

setSize(130, 30);

}

}

**Header Class:**

**package** view;

**import** javax.swing.JPanel;

**import** controller.Controller;

**import** gui.Button;

**import** gui.TextBox;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

**public** **class** Header **extends** JPanel{

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**private** TextBox \_text;

**public** Header(Controller controller){

setLayout(**null**);

TextBox text = **new** TextBox("Unnamed", "Enter a new name: ", "Change name");

**this**.\_text = text;

text.setSize(200, 30);

text.setLocation(0, 0);

text.setBackground(Colors.***NOTESQUARE\_OFF***);

Button load = **new** Button(controller, "Load");

load.setSize(60, 30);

load.setLocation(210, 0);

Button save = **new** Button(controller, "Save");

save.setSize(60, 30);

save.setLocation(280, 0);

Button quit = **new** Button(controller, "Quit");

quit.setSize(60, 30);

quit.setLocation(380, 0);

**this**.setSize(440, 30);

**this**.add(text);

**this**.add(load);

**this**.add(save);

**this**.add(quit);

}

**public** String getFile(){

String text = **this**.\_text.getText();

**return** text;

}

}

**NoteSquare Class:**

**package** view;

**import** java.awt.event.MouseEvent;

**import** controller.Controller;

**import** gui.Box;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

**public** **class** NoteSquare **extends** Box{

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** **static** **final** **int** ***SIZE*** = 30;

**private** Controller \_controller;

**private** **int** \_track;

**private** **int** \_beat;

**private** **int** \_value;

**public** NoteSquare(Controller controller, **int** track, **int** beat){

**super**();

**this**.\_controller = controller;

**this**.\_track = track;

**this**.\_beat = beat;

**this**.\_value = 0;

setSize(***SIZE***, ***SIZE***);

setBackground(Colors.***NOTESQUARE\_OFF***);

}

**public** **int** getTrack(){

**return** **this**.\_track;

}

**public** **int** getBeat(){

**return** **this**.\_beat;

}

**public** **int** getValue(){

**return** **this**.\_value;

}

**public** **void** setValue(**int** value){

**this**.\_value = value;

**if**(**this**.\_value == 0){

**this**.setBackground(Colors.***NOTESQUARE\_OFF***);

} **else** **if**(**this**.\_value == 1){

**this**.setBackground(Colors.***NOTESQUARE\_ON***);

}

repaint();

}

@Override

**public** **void** mousePressed(MouseEvent mev){

**if**(**this**.getValue() == 0){

**this**.setValue(1);

} **else** **if**(**this**.getValue() == 1) {

**this**.setValue(0);

}

**this**.\_controller.noteSquareClicked(**this**);

}

@Override

**public** String toString(){

**return** "NoteSquare(track = " + **this**.getTrack() + ", beat = " + **this**.getBeat() + ")";

}

}

**SoundBank Class:**

**package** view;

**import** javax.swing.JPanel;

**import** controller.Controller;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

**public** **class** SoundBank **extends** JPanel {

**private** **static** **final** **long** ***serialVersionUID*** = 6249914173769272977L;

**private** SoundNameBox[] \_soundNameBoxes;

**public** SoundBank(Controller aController, **int** numTracks) {

setLayout(**null**);

**int** width = 0;

\_soundNameBoxes = **new** SoundNameBox[numTracks];

**int** y = 0;

**for** (**int** i=0; i < numTracks; i++) {

SoundNameBox soundNameBox = **new** SoundNameBox(aController, i+1);

soundNameBox.setLocation(0, y);

\_soundNameBoxes[i] = soundNameBox;

\_soundNameBoxes[i].setBackground(Colors.***NOTESQUARE\_OFF***);

y += Tracks.***GAP\_SIZE*** + soundNameBox.getSize().getHeight();

add(soundNameBox);

}

**if** (\_soundNameBoxes[0] != **null**)

width = (**int**) \_soundNameBoxes[0].getSize().getWidth();

setSize(width, y);

}

**public** **void** setSoundName(**int** trackNum, String soundName) {

\_soundNameBoxes[trackNum - 1].setText(soundName);

}

}

**SoundChooser Class:**

**package** view;

**import** java.awt.BorderLayout;

**import** java.awt.GridLayout;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.event.KeyEvent;

**import** java.awt.event.KeyListener;

**import** java.awt.event.MouseEvent;

**import** java.awt.event.MouseListener;

**import** javax.swing.JButton;

**import** javax.swing.JDialog;

**import** javax.swing.JList;

**import** javax.swing.JPanel;

**import** javax.swing.JScrollPane;

**import** model.Sound;

**import** controller.Controller;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

**public** **class** SoundChooser **extends** JDialog **implements** ActionListener, KeyListener, MouseListener {

**private** JButton \_play, \_ok, \_cancel;

@SuppressWarnings("rawtypes")

**private** JList \_soundNames;

**private** Controller \_controller;

**private** **int** \_trackNumber;

**private** **static** SoundChooser *\_theSoundChooser* = **null**;

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** **static** **void** open(Controller controller, **int** trackNumber, String initialValue){

**if**(*\_theSoundChooser* == **null**)

*\_theSoundChooser* = **new** SoundChooser(controller, trackNumber);

**else**

*\_theSoundChooser*.\_trackNumber = trackNumber;

*\_theSoundChooser*.\_setInitialValue(initialValue);

*\_theSoundChooser*.setVisible(**true**);

}

@SuppressWarnings({ "unchecked", "rawtypes" })

**private** SoundChooser(Controller controller, **int** trackNumber){

**this**.setTitle("Sound chooser");

**this**.setDefaultCloseOperation(JDialog.***HIDE\_ON\_CLOSE***);

**this**.setModal(**true**);

\_controller = controller;

\_trackNumber = trackNumber;

**this**.setLayout(**new** BorderLayout());

String[] soundNames = Sound.*getAllSoundFileNames*();

\_soundNames = **new** JList(soundNames);

\_soundNames.addKeyListener(**this**);

\_soundNames.addMouseListener(**this**);

JScrollPane scrollPane = **new** JScrollPane(\_soundNames);

**this**.add(scrollPane, BorderLayout.***CENTER***);

JPanel buttonPanel = **new** JPanel();

buttonPanel.setLayout(**new** GridLayout(1, 3));

\_ok = **new** JButton("Use this sound");

\_play = **new** JButton("Play");

\_cancel = **new** JButton("Cancel");

\_play.addActionListener(**this**);

\_ok.addActionListener(**this**);

\_cancel.addActionListener(**this**);

buttonPanel.add(\_ok);

buttonPanel.add(\_play);

buttonPanel.add(\_cancel);

**this**.add(buttonPanel, BorderLayout.***SOUTH***);

**this**.setSize(400, 400);

**this**.setLocationRelativeTo(**null**);

}

@Override

**public** **void** actionPerformed(ActionEvent aev){

Object source = aev.getSource();

**if**(source == \_play)

\_play();

**else** **if**(source == \_ok)

\_select();

**else**

\_cancel();

}

@Override

**public** **void** keyPressed(KeyEvent kev){

**int** keyCode = kev.getKeyCode();

**switch**(keyCode){

**case** 10: // enter

\_select();

**break**;

**case** 27: // escape

\_cancel();

**break**;

**case** 32: // space

\_play();

**break**;

}

}

@Override

**public** **void** keyReleased(KeyEvent kev){}

@Override

**public** **void** keyTyped(KeyEvent kev){}

**private** **void** \_cancel(){

**this**.setVisible(**false**);

}

**private** **void** \_play(){

String soundName = (String)\_soundNames.getSelectedValue();

**if**(soundName != **null** && !soundName.equals("(none)")){

Sound s = **new** Sound(soundName);

s.play();

}

}

@Override

**public** **void** mouseClicked(MouseEvent mev){

**if**(mev.getButton() == MouseEvent.***BUTTON1*** && mev.getClickCount() == 2)

\_select();

}

@Override

**public** **void** mouseEntered(MouseEvent mev){}

@Override

**public** **void** mouseExited(MouseEvent mev){}

@Override

**public** **void** mousePressed(MouseEvent mev){}

@Override

**public** **void** mouseReleased(MouseEvent mev){}

**private** **void** \_select(){

**if**(\_controller != **null**){

String soundName = (String)\_soundNames.getSelectedValue();

\_controller.soundNameSelected(\_trackNumber, soundName);

}

**this**.setVisible(**false**);

}

**private** **void** \_setInitialValue(String initialValue){

**if**(initialValue == **null** || initialValue.length() == 0)

\_soundNames.setSelectedIndex(0);

**else**

\_soundNames.setSelectedValue(initialValue, **true**);

}

}

**SoundNameBox Class:**

**package** view;

**import** java.awt.event.MouseEvent;

**import** controller.Controller;

**import** gui.Box;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

**public** **class** SoundNameBox **extends** Box{

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** **static** **final** **int** ***WIDTH*** = 130;

**public** **static** **final** **int** ***HEIGHT*** = 30;

**private** Controller \_controller;

**private** **int** \_track;

**public** SoundNameBox(Controller controller, **int** track){

**super**();

**this**.\_track = track;

**this**.\_controller = controller;

setSize(***WIDTH***, ***HEIGHT***);

}

@Override

**public** **void** mousePressed(MouseEvent mev){

String text = getText();

SoundChooser.*open*(\_controller, \_track, text);

}

}

**Tracks Class:**

**package** view;

**import** javax.swing.JPanel;

**import** controller.Controller;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

**public** **class** Tracks **extends** JPanel{

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** **static** **final** **int** ***GAP\_SIZE*** = 10;

**private** NoteSquare[][] \_squares;

**public** Tracks(Controller controller, **int** tracks, **int** beats){

**this**.setLayout(**null**);

NoteSquare [][] squares = **new** NoteSquare[tracks][beats];

**int** n = 0;

**int** m = 0;

**for** (**int** i = 0; i < tracks; i++){

**for** (**int** j = 0; j < beats; j++){

squares[i][j] = **new** NoteSquare(controller, i, j);

**this**.add(squares[i][j]);

squares[i][j].setLocation(n, m);

**if** (j < beats - 1){

n += NoteSquare.***SIZE*** + ***GAP\_SIZE***;

} **else** {

m += NoteSquare.***SIZE*** + ***GAP\_SIZE***;

n = 0;

}

}

}

setSize((beats \* NoteSquare.***SIZE***) + (beats \* ***GAP\_SIZE***), ((tracks \* NoteSquare.***SIZE***) + (tracks \* ***GAP\_SIZE***)));

}

**public** NoteSquare[][] getSquares() {

**return** \_squares;

}

}

**View Class:**

**package** view;

**import** gui.NumberSlider;

**import** controller.Controller;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

**public** **class** View {

**private** BeatNumbers \_numBeats;

**private** SoundBank \_sounds;

**private** Header \_head;

**private** Tracks \_track;

**private** NumberSlider \_slider;

**public** View(Controller controller, **int** numTracks, **int** numBeats){

Window window = **new** Window(controller, "Rhythminator");

window.setSize(900, 500);

Header head = **new** Header(controller);

**this**.\_head = head;

head.setLocation(10, 10);

window.add(head);

SoundBank sounds = **new** SoundBank(controller, numTracks);

**this**.\_sounds = sounds;

sounds.setLocation(10, 60);

window.add(sounds);

Tracks tracks = **new** Tracks(controller, numTracks, numBeats);

**this**.\_track = tracks;

tracks.setLocation(150, 60);

window.add(tracks);

ControlButtons controls = **new** ControlButtons(controller);

controls.setLocation(10, numTracks \* NoteSquare.***SIZE*** + 30 + numTracks \* Tracks.***GAP\_SIZE*** + 30);

window.add(controls);

BeatNumbers beatNum = **new** BeatNumbers(numBeats);

**this**.\_numBeats = beatNum;

beatNum.setLocation(150, numTracks \* NoteSquare.***SIZE*** + 30 + numTracks \* Tracks.***GAP\_SIZE*** + 30);

window.add(beatNum);

NumberSlider slider = **new** NumberSlider(controller, **null**, 10, 200, 25);

slider.setLocation(150, numTracks \* NoteSquare.***SIZE*** + 30 + numTracks \* Tracks.***GAP\_SIZE*** + 70);

window.add(slider);

window.setVisible(**true**);

}

**public** **void** setBeatNumber(**int** beat){

**this**.\_numBeats.setBeatNumber(beat);

}

**public** **void** clearBeatNumbers(){

**this**.\_numBeats.clear();

}

**public** **void** setSoundName(**int** track, String name){

**this**.\_sounds.setSoundName(track, name);

}

**public** Header getHead() {

**return** \_head;

}

**public** Tracks getTrack() {

**return** \_track;

}

**public** NumberSlider getSlider() {

**return** \_slider;

}

**public** BeatNumbers getNumBeats(){

**return** **this**.\_numBeats;

}

}

**Window Class:**

**package** view;

**import** java.awt.event.KeyEvent;

**import** controller.Controller;

//Rhythminator Part 3

//CSE1102 Project 07, Spring 2016

//Bryan Arnold

//5/2/16

//TA: Zigeng Wang

//Section: 51

//Instructor: Jeffrey A. Meunier

/\*\*

\* The view.Window extends the gui.Window by notifying the controller

\* when a key event takes place.

\* **@author** jeff

\*/

**public** **class** Window **extends** gui.Window{

**private** Controller \_controller;

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** Window(Controller controller, String name){

**super**(name);

\_controller = controller;

}

@Override

**public** **void** keyPressed(KeyEvent kev){

\_controller.keyPressed(kev.getKeyCode());

}

@Override

**public** **void** keyReleased(KeyEvent kev){

\_controller.keyReleased(kev.getKeyCode());

}

@Override

**public** **void** keyTyped(KeyEvent kev){

//System.out.println("Window kev = " + kev);

\_controller.keyTyped(kev.getKeyChar());

}

}