```
/* This project is a demo showing a sound card selector and tester.
 * The UI would look nice and work better, but I've decided
 * that's outside the scope of this demo.
 *
 * Code from this demo will eventually be reused in
 * a Java version of my ClipBoard reader.
 *
 * NOTE: It seems to take a moment for a digital sound (S/PDIF)
 * to start playing.
 *
 * NOTE: In this demo, two sound cards with the same
 * name are not supported. Only the top one in the
 * combo box can be selected.  (Whichever the computer sees first.)
 *
 * NOTE: This demo needs DirectX on Windows and SDL on Linux
 * since the demo accesses the hardware directly.
 *
 * NOTE: I can't test on macOS since I don't have macOS.
 *
 * Project copyright follows
 *
 *
 * Copyright (c) 2017, James Marchant
 * All rights reserved.
 * Redistribution and use in source and binary forms, with or without
```

```
 * modification, are permitted provided that the following
 * conditions are met:
 * 1. Redistributions of source code must retain
 *    the above copyright notice, this list of conditions
 *    and the following disclaimer.
 * 2. Redistributions in binary form must reproduce
 *    the above copyright notice, this list of conditions
 *    and the following disclaimer in the documentation
 *    and/or other materials provided with the distribution.
 *    THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
 *    CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
 *    INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
 *    AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
 *    IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
 *    ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 *    (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 *    LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 *    ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TOR
 *    (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 *    SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *    The views and conclusions contained in the software and documentation are those
 *    of the authors and should not be interpreted as representing official policies,
 *    either expressed or implied, of the FreeBSD Project.
 */
package info.jamiemarchant.test.ui;
```

```java
import java.awt.EventQueue;


public class MainWindow {
    private JFrame _jfrmSoundCardSelector;
    private JButton _jbtnTest;
    private boolean _startingUp = true;
    private Thread _audioThread;
    private WhiteNoiseRunnable _wnrunnable;
    private JComboBox<String> _jcbSoundCardNames;
    private ArrayList<Line> _allLines = new ArrayList<Line>();
    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    MainWindow window = new MainWindow();
                    window._jfrmSoundCardSelector.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
```

```
    });
}

/**
 * Create the application.
 */
public MainWindow() {
    initialize();
}
/**
 * Initialize the contents of the frame.
 */
private void initialize() {
    _wnrunnable = new WhiteNoiseRunnable();
    _audioThread = null;
    _jfrmSoundCardSelector = new JFrame();
    _jfrmSoundCardSelector.setTitle("Sound Card Selector");
    _jfrmSoundCardSelector.setBounds(100, 100, 954, 97);
    _jfrmSoundCardSelector.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JButton _btnOk = new JButton("Ok");
    _btnOk.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent arg0) {
            JOptionPane.showMessageDialog(_btnOk,"Ok hit. In a larger program " +
                                                 " the results would be saved.");
```

```
            System.exit(1);
        }
    });

    _jfrmSoundCardSelector.getContentPane().add(_btnOk, BorderLayout.EAST);
    _jcbSoundCardNames = new JComboBox<String>();
    _jcbSoundCardNames.addItemListener(new ItemListener() {
        public void itemStateChanged(ItemEvent arg0) {
            if(!_startingUp){
                stopSound();
            }
        }
    });

    _jcbSoundCardNames.addPropertyChangeListener(new PropertyChangeListener() {
        public void propertyChange(PropertyChangeEvent arg0) {
            // No longer used; wrong event.
            //This part might be needed for the designer.
        }
    });

    _jfrmSoundCardSelector.getContentPane().add(_jcbSoundCardNames,
                                                BorderLayout.CENTER);
    // Initialize the mixer.
    Info[] infoAboutMixers = AudioSystem.getMixerInfo();
```

```
for(javax.sound.sampled.Mixer.Info infoAboutMixer : infoAboutMixers){
    Mixer aMixer = AudioSystem.getMixer(infoAboutMixer);
    try {
        aMixer.open();
        javax.sound.sampled.Line.Info[] infoLines = aMixer.getSourceLineInfo();
        System.out.println(infoLines.length);
        int portCounter = 1;
        for(javax.sound.sampled.Line.Info infoLine : infoLines){
            Line line = aMixer.getLine(infoLine);
            String niceOutput = "";
            //niceOutput += infoAboutMixer.getVendor();
            niceOutput += infoAboutMixer.getName();
            niceOutput += infoAboutMixer.getDescription();
            niceOutput += ": port#" + portCounter;
            /* To be meaningful name is needed for Linux, but description is
             * needed for Windows. As I said at the top, I can't  test macOS.
             * */

            portCounter++;
            // Only add "SourceDataLines". At the moment,
            // we only handle these objects.
            if(line instanceof SourceDataLine){
                _allLines.add(line);// all lines for all mixers.
                _jcbSoundCardNames.addItem(niceOutput);
            }
```

```
                /* Right now, this is limited to
                 * sound devices(all speakers, S/PDIF, HDMI).
                 */
                /* In the future, I want it to work on a port level
                 * (front speakers of Sound Blaster Live)*/
                /*
                 * Just a reminder that descriptions come
                 * from the OS and how they are named are
                 * beyond my control.
                 */
            }
            aMixer.close();
        } catch (LineUnavailableException e) {
            System.err.println(e.getMessage());
            JOptionPane.showMessageDialog(_jfrmSoundCardSelector,
            "The sound card '" + aMixer.getMixerInfo().getDescription() +
            "' Could not be accessed, is it busy?", "warning",
            JOptionPane.WARNING_MESSAGE);
        }

    }
    _jbtnTest = new JButton("Test");

    _jbtnTest.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent arg0) {
```

```
            if(_jbtnTest.getText().equals("Stop")){
                stopSound();
            }
            else{
                int selectedIndex = _jcbSoundCardNames.getSelectedIndex();
                Line theLine = _allLines.get(selectedIndex);
                System.out.println(theLine.getClass().getName());
                _wnrunnable.setDateLine((SourceDataLine) theLine);
                _wnrunnable.setKeepLooping(true);
                _audioThread = new Thread(_wnrunnable);
                // You can't restart a thread once it's stopped,
                // so we need to make a new thread.
                _audioThread.start();
                _jbtnTest.setText("Stop");
            }
        }
    });
    _jfrmSoundCardSelector.getContentPane().add(_jbtnTest, BorderLayout.WEST);

    _startingUp = false;
}
protected void stopSound() {
    if(_audioThread == null){
        return; // No sound has started.
    }
```

```
            System.out.println("Stop sound called");
            _jbtnTest.setText("Test");
            _wnrunnable.setKeepLooping(false);
            System.out.println("Joing the endless loop");
            try {
                _audioThread.join();// Wait for the thread to stop.
            } catch (InterruptedException e) {
                System.err.println("Failed to naturally stop thread");
                int selectedIndex = _jcbSoundCardNames.getSelectedIndex();
                _allLines.get(selectedIndex).close();
                /* The above may cause issues if a sound card is disconnected,
                 * but that's outside the scope of this demo. The
                 * order of items in the list will remain the same,
                 * but they may not be alphabetical.
                 * Sorting them is outside the scope of this demo.
                 */
                //theLine.close();
                JOptionPane.showMessageDialog(_jbtnTest, "The program has encountered an" +
                " error and must quit.");
                // Probably safer than stopping threads. I need to make sure resources are free.
                System.exit(1);
            }
        }
    }
```

```
package info.jamiemarchant.test.threading;
import java.nio.ByteBuffer;
```

```java
import java.util.Random;
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.SourceDataLine;
import javax.swing.JOptionPane;
/***
 * Runnable that creates white noise
 * @author Jamie Marchant
 *
 */

public class WhiteNoiseRunnable implements Runnable {

    private volatile boolean _keepLooping = true;
    private volatile ByteBuffer _buffer;
    private volatile SourceDataLine _line = null;
    public void setSourceDataLineAndPort(SourceDataLine lvar){
        _line = lvar;
    }
    public void setDateLine(SourceDataLine theLine){
        _line = theLine;
    }
    /***
     * The packet size used to generate the noise.
     */
```

```java
// Note sure why this is 5000, since it was not explained in the
// sample.
final static public int PACKET_SIZE = 5000;
final static public int SAMPLE_SIZE = 2;
@Override
/***
 * Creates white noise on the audio mixer.
 * @throws NullPointerException if an audio line has not been set.
 */
public void run() {
    if(_line != null){
         // code from:
         // https://stackoverflow.com/questions/26963342/generating-colors-of-noise-in-java
         AudioFormat format = new AudioFormat(44100, 16, 1, true, true);
         //DataLine.Info info = new DataLine.Info(SourceDataLine.class, format, PACKET_SIZE * 2);
        try {
            _line.open(format);
            _line.start();
            _buffer = ByteBuffer.allocate(PACKET_SIZE);
            // NOTE: New IO thing, so Java 7?
        } catch (LineUnavailableException e) {
            // Must be caught here but that's ok.
            System.err.println(e.getMessage());
            JOptionPane.showMessageDialog(null, "Can't open audio device, is it busy?");
            return;
```

```
            }
        }else{
            throw new NullPointerException("Line to use is now null");
        }
        Random random = new Random();
        while(_keepLooping){
            _buffer.clear();
             for (int i=0; i < PACKET_SIZE /SAMPLE_SIZE; i++) {
                 _buffer.putShort((short) (random.nextGaussian() * Short.MAX_VALUE));
             }
             _line.write(_buffer.array(), 0, _buffer.position());
        }
        _line.stop();
        _line.close();
    }
    /***
     * Sets the value that keeps the thread running. Do this
     * and call "join" when you want to stop the thread.
     * @param var - a true or false value, you probably want to set it false.
     */
    public void setKeepLooping(boolean var){
        _keepLooping = var;
    }
}
```