为Java程序中添加播放MIDI音乐功能 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/271/2021_2022__E4_B8_BAJava_E7_A8_8B_c104_271657.htm Java在多媒体处理方面的确优势不大，但是我们在程序中有些时候又需要一些音乐做为点缀，如果播放的音乐是wav等波形音频文件,又挺大,所以背景音乐最好就是MIDI了，可是网上很多播放MIDI的教程都是简单的几句话的例子，并且没有考虑资源的释放问题，如果程序长久运行的话,就会出现内存越耗越多的情况，以至于最后抛出一个java.lang.OutOfMemoryError，整个程序就挂了。 在MIDI的播放中，一个类是比较重要的，那就是MidiSystem类，它负责整个MIDI播放设备等的管理，其实就是Seqencer，它就是一个MIDI播放设置，用于播放MIDI序列的，还有一个类叫Seqence，它就是MIDI的序列,MIDI的序列可以自己由程序生成，也可以从文件中或者URL中读取。 下面我们来看一个例子吧：/* * Test5.java * * Created on 2007-9-22, 11:16:22 * * To change this template, choose Tools | Templates * and open the template in the editor. */package test1.import java.io.File.import java.io.IOException.import java.io.InputStream.import java.util.Hashtable.import java.util.Map.import java.util.logging.Level.import java.util.logging.Logger.import javax.sound.midi.InvalidMidiDataException.import javax.sound.midi.MidiSystem.import javax.sound.midi.MidiUnavailableException.import javax.sound.midi.Sequence.import javax.sound.midi.Sequencer./** * * @author hadeslee */public class Test5 implements Runnable{

private Sequencer midi. private String[] names={"1.mid","2.mid","3.mid","4.mid","5.mid"}. private int i. private MapString,Sequence> map. public Test5(){ initMap(). new Thread(this).start(). } private void initMap(){ try { map = new HashtableString, Sequence>(). midi = MidiSystem.getSequencer(false). midi.open(). for (String s : names) { try { Sequence s1 = MidiSystem.getSequence(new File(s)). map.put(s, s1). } catch (InvalidMidiDataException ex) { Logger.getLogger(Test5.class.getName()).log(Level.SEVERE, null, ex). } catch (IOException ex) { Logger.getLogger(Test5.class.getName()).log(Level.SEVERE, null, ex). } } } catch (MidiUnavailableException ex) { Logger.getLogger(Test5.class.getName()).log(Level.SEVERE, null, ex). } } private void createPlayer(String name){ try { Sequence se=map.get(name). midi.setSequence(se). midi.start(). }catch (InvalidMidiDataException ex) { Logger.getLogger(Test5.class.getName()).log(Level.SEVERE, null, ex). } } public void run(){ while(true){ try { System.out.println("换文件了." (i)). String name=names[(int)(Math.random()*names.length)]. createPlayer(name). Thread.sleep(10000). } catch (InterruptedException ex) { Logger.getLogger(Test5.class.getName()).log(Level.SEVERE, null, ex). } } } public static void main(String[] args) { new Test5(). }} 在这里有很重要的一点，那就是在程序运行的时候，只要一个Seqencer就可以了，我以前在程序里面每次播放的时候都生

成了一个Seqencer，因为那个时候我想，我都调用它的close()方法了，它还能被打开吗？其实它还可以再度被打开的，就是这样一种惯性思维使得程序最终因内存溢出而崩溃。 现在按我这种方式播,哪怕10毫秒换一次MIDI都可以，换个几万次内存一点都没有加，呵呵,真是防不胜防啊。