```
/*
      © Ali Rıza SARAL
      Attribution-NonCommercial-NoDerivatives 4.0 International
 */
package jfuguetestars;

import java.io.File;
import java.io.IOException;
import static java.lang.Math.exp;
import static java.lang.Math.sin;
import java.util.Random;

import javax.sound.midi.InvalidMidiDataException;
import javax.sound.midi.MidiSystem;
import javax.xml.parsers.ParserConfigurationException;
import nu.xom.ParsingException;
import org.jfugue.midi.MidiParser;
import org.staccato.StaccatoParserListener;
import org.jfugue.midi.MidiFileManager;
import org.jfugue.pattern.Pattern;

import org.jfugue.integration.MusicXmlParser;
import org.jfugue.integration.MusicXmlParserListener;
import org.jfugue.player.Player;
import org.w3c.dom.Node;
//import org.jfugue

public class algorithmTST7 {
    public static void main(String[] args) throws
InvalidMidiDataException, IOException, ParserConfigurationException,
ParsingException {
        MidiParser parser = new MidiParser();
        StaccatoParserListener listener = new StaccatoParserListener();
        parser.addParserListener(listener);
       // parser.parse(MidiSystem.getSequence(new
File("C:\\Users\\ars\\Documents\\MuseScore2\\Scores\\My_First_Score.mid")
));
        //parser.parse(MidiSystem.getSequence(new
File("C:\\Users\\ars\\Documents\\MuseScore2\\Scores\\Cello_Guitar_Sonata_
Mvmnt1.mid")));

        Pattern pattern = listener.getPattern();
        //System.out.println(staccatoPattern);

        Player player = new Player();

pattern.clear();
System.out.println("AlgorithmTST1()------------------------------");
pattern.add(performAlgorithmTST7());

 player.play(pattern);


         player = new Player();
```

```
        try {
            MidiFileManager
                    .savePatternToMidi(pattern, new
File("C:\\\\Users\\\\ars\\\\Documents\\\\MuseScore3\\\\Scores\\\\algorith
mTST7-12sqrtx.mid"));
        } catch (IOException ex) {
        }

        //player.play(pattern);

    }
    public static int noteCount=88;
    public static String[] chromatic_scale_piano= {
        //"C0","C#0","D0","D#0","E0","F0","F#0","G0","G#0",
        "A0","A#0","B0",
        "C1","C#1","D1","D#1","E1","F1","F#1","G1","G#1","A1","A#1","B1",
        "C2","C#2","D2","D#2","E2","F2","F#2","G2","G#2","A2","A#2","B2",
        "C3","C#3","D3","D#3","E3","F3","F#3","G3","G#3","A3","A#3","B3",
        "C4","C#4","D4","D#4","E4","F4","F#4","G4","G#4","A4","A#4","B4",
        "C5","C#5","D5","D#5","E5","F5","F#5","G5","G#5","A5","A#5","B5",
        "C6","C#6","D6","D#6","E6","F6","F#6","G6","G#6","A6","A#6","B6",
        "C7","C#7","D7","D#7","E7","F7","F#7","G7","G#7","A7","A#7","B7",
        "C8"
    };
    public static String performAlgorithmTST7(){
        String [] score = generateAlgorithmTST1score();
        //String m = printAlgorithmTST7sin(score);
        String m ="";
        m="T120 V0 I[Piano] Rw ";
       //m= printAlgorithmTST7sin1divx(score);
       //m= printAlgorithmTST7expASC(score);
//       m= printAlgorithmTST7expDESC(score);
//       m= printAlgorithmTST7expDESCNEG(score);
      //m= printAlgorithmTST7tan(score);
      //m= printAlgorithmTST7onedivx(score);
      //m= printAlgorithmTST7onedivsqrtx(score);
      m= printAlgorithmTST7sqrtx(score);


        return m;
    }

    public static String [] generateAlgorithmTST1score(){
        String [] score = new String[noteCount];
        for (int i=0;i<noteCount;i++){
            score[i]=chromatic_scale_piano[i];
        }
        return score;
    }

  public static String printAlgorithmTST7sin1divx(String [] score){
//failed
        String m ="";
        double j=0;
```

```
        int Number1=0;
        for (int i=0;i<48;i++){
          j=j+0.4;
            Number1=(int) (Math.sin(1/j)*10);
            if (Number1>87||Number1<0) break;
           m+=score[Number1+30]+"s ";
            System.out.println(Number1);
        }
        System.out.println(m);
        return m;
    }
 public static String printAlgorithmTST7expASC(String [] score){
//failed
        String m ="";
        double j=0;
        int Number1=0;
        for (int i=0;i<120;i++){
          j=j+0.1;
            Number1=(int) (exp(j));
            if (Number1>87||Number1<0) break;
           m+=score[Number1]+"s ";
            System.out.println(Number1);
        }
        System.out.println(m);
        return m;
    }
 public static String printAlgorithmTST7expDESC(String [] score){
//failed
        String m ="";
        double j=0;
        int Number1=0;
        for (int i=0;i<120;i++){
          j=j+0.1;
            Number1=(int) (exp(j));
            if (Number1>87||Number1<0) break;
           m+=score[88-Number1]+"s ";
            System.out.println(Number1+" "+j);
        }
        System.out.println(m);
        return m;
    }
  public static String printAlgorithmTST7expDESCNEG(String [] score){
//failed
        String m ="";
        double j=4.4;
        int Number1=0;
        for (int i=0;i<120;i++){
          j=j-0.1;
            Number1=(int) (exp(j));
            if (Number1>87||Number1<0) break;
           m+=score[Number1+14]+"s ";
            System.out.println(Number1+" "+j);
        }
        System.out.println(m);
```

```
        return m;
    }
 public static String printAlgorithmTST7tan(String [] score){  //failed
        String m ="";
        double j=40;
        int Number1=1;
        for (int i=0;i<240;i++){
          j=j+0.7;
            Number1=(int) (Math.tan(j*Math.PI/180));
            if (Number1>87||Number1<0) break;
           m+=score[Number1]+"s ";
            System.out.println(Number1+" "+j);
        }
        System.out.println(m);
        return m;
    }
 public static String printAlgorithmTST7onedivx(String [] score){
//failed
        String m ="";
        double j=1;
        int Number1=1; double k=0;
        for (int i=1;i<240;i++){
            j=j+1;
          k=1/Math.sqrt(j);
          k=10000/j;
            Number1=(int) ((k*Math.PI/180));
            if (Number1>87||Number1<0) break;
           m+=score[Number1]+"s ";
            System.out.println(Number1+" "+j+" "+k);
        }
        System.out.println(m);
        return m;
    }
  public static String printAlgorithmTST7onedivsqrtx(String [] score){
//failed
        String m ="";
        double j=1;
        int Number1=1; double k=0;
        for (int i=1;i<240;i++){
            j=j+1;
          k=6300/Math.sqrt(j);
            Number1=(int) ((k*Math.PI/180));
            if (Number1>87||Number1<0) break;
           m+=score[Number1]+"s ";
            System.out.println(Number1+" "+j+" "+k);
        }
        System.out.println(m);
        return m;
    }
   public static String printAlgorithmTST7sqrtx(String [] score){
//failed
        String m ="";
        double j=1;
        int Number1=1; double k=0;
```

```
        for (int i=1;i<240;i++){
            j=j+1;
          k=700*Math.sqrt(j);
            Number1=(int) ((k*Math.PI/180));
            if (Number1>87||Number1<0) break;
           m+=score[Number1]+"s ";
            System.out.println(Number1+" "+j+" "+k);
        }
        System.out.println(m);
        return m;
    }
  public static int randomInteger(int min, int max) {

    Random rand = new Random();

    // nextInt excludes the top value so we have to add 1 to include the
top value
    int randomNum = rand.nextInt((max - min) + 1) + min;

    return randomNum;
    }
  public static boolean existsPrev(String[] playedNotes,String note, int
depthRepeat, int currentPOS){

      //for (int i=0;i<playedNotes.length;i++){
      for (int i=0;i<depthRepeat;i++){
          if (currentPOS-i>-1&&playedNotes[currentPOS-i]!=null &&
playedNotes[currentPOS-i].indexOf(note)>-1) return true;
      }
      return false;
  }

}
```