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

import java.io.File;
import java.io.IOException;
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 algorithmTST3 {
    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("AlgorithmTST3()------------------------------");
pattern.add(initPerform());



 player.play(pattern);


         player = new Player();
```

```
        try {
            MidiFileManager
                    .savePatternToMidi(pattern, new
File("C:\\\\Users\\\\ars\\\\Documents\\\\MuseScore3\\\\Scores\\\\algorith
mTST1-2.mid"));
        } catch (IOException ex) {
        }

        //player.play(pattern);

    }
    public static int floor =70;
    public static int height=7;
    public static int cycles=3;
    public static int step=1;
    public static String noteVal="s";

    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 [] generateAlgorithmTST3score(int floor, int
height){
        String [] score = new String[height];
        int j=0;
        for (int i=floor;i<floor+height;i++){
            score[j]=chromatic_scale_piano[i];
            j++;
        }
        return score;
    }
public static String initPerform(){
    String m="";
    floor =70; height=7;cycles=3;step=1;noteVal="t";
    m+= performAlgorithmTST3ADscsingleSTEP();
    floor =50; height=9;cycles=4;step=2;noteVal="s";
    m+= performAlgorithmTST3ADscsingleSTEP();
    floor =20; height=12;cycles=2;step=4;noteVal="q";
    m+= performAlgorithmTST3ADscsingleSTEP();
    floor =40; height=9;cycles=5;step=3;noteVal="t";
    m+= performAlgorithmTST3ADscsingleSTEP();
    floor =60; height=7;cycles=5;step=1;noteVal="t";
    m+= performAlgorithmTST3ADscsingleSTEP();
    floor =61; height=7;cycles=5;step=2;noteVal="t";
    m+= performAlgorithmTST3ADscsingleSTEP();
```

```
    floor =65; height=7;cycles=9;step=1;noteVal="o";
    m+= performAlgorithmTST3ADscsingleSTEP();
    floor =0; height=12;cycles=3;step=5;noteVal="t";
    m+= performAlgorithmTST3ADscsingleSTEP();
    m+="Rw A0o ";
    return m;
}
    public static String performAlgorithmTST3ADscsingleSTEP(){
        String [] score = generateAlgorithmTST3score(floor,height);
        String m ="";
        for (int i=0;i<cycles+1;i++){
            m+= printAlgorithmTST3scoreADscWithSTEP(score, step);
        }
        //m+="Rw A0o ";
        return m;
    }
    public static String performAlgorithmTST3ADscsingleSTEPrnd(){
        String [] score = generateAlgorithmTST3score(floor,height);
        String m ="";
        for (int i=0;i<cycles+1;i++){
            m+= printAlgorithmTST3scoreADscWithSTEP(score, step);
        }
        m+="Rw A0o ";
        return m;
    }
    public static String printAlgorithmTST3scoreADscWithSTEP(String []
score, int step){
        String m ="";
        for (int i=0;i<score.length/step;i++){
            m+=score[i*step]+noteVal+" ";
        }
        for (int i=((score.length-1)/step )-1;i>0;i--){
            m+=score[i*step]+noteVal+" ";
        }
        System.out.println(m);
        return m;
    }
// Note letter (C, D, E, F, G, A, or B), accidental (#, b, natural), and
octave
//• Rest is specified with an R • Duration: w, h, q, i, s, t, x, o
(whole, half, etc. down to 128th)
//
    //************************************

  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;
  }

}
```