```
/*
      © 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 algorithmTST2 {
    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("AlgorithmTST2()------------------------------");
//pattern.add(performAlgorithmTST2Asc());

//pattern.add(performAlgorithmTST2Desc()); // 3
//pattern.add(performAlgorithmTST2AscsingleSTEP(3)); // 3
//pattern.add(performAlgorithmTST2DescsingleSTEP(5)); // 3
//pattern.add(performAlgorithmTST2AscsingleSTEP(7)); // 3
//pattern.add(performAlgorithmTST2DescsingleSTEP(4)); // 3
//pattern.add(performAlgorithmTST2DescsingleSTEP(3)); // 3
```

```
//pattern.add(performAlgorithmTST2DescsingleSTEP(11)); // 3
//pattern.add(performAlgorithmTST2DescsingleSTEP(10)); // 3
//pattern.add(performAlgorithmTST2DescsingleSTEP(12)); // 3
//pattern.add(performAlgorithmTST2Desc()); // 3

//pattern.add(performAlgorithmTST2AscallSTEPS());
//pattern.add(performAlgorithmTST2DescallSTEPS()); //
//pattern.add(performAlgorithmTST2invAscallSTEPS());
//pattern.add(performAlgorithmTST2ADscallSTEPS());  //
//pattern.add(performAlgorithmTST2AscallSTEPSblurred());
pattern.add(performAlgorithmTST2ADscallSTEPSblurred());  //4


 player.play(pattern);


         player = new Player();
        try {
            MidiFileManager
                    .savePatternToMidi(pattern, new
File("C:\\\\Users\\\\ars\\\\Documents\\\\MuseScore3\\\\Scores\\\\algorith
mTST1-2x.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 performAlgorithmTST2Asc(){
        String [] score = generateAlgorithmTST2score();
        //String m = printAlgorithmTST2scoreASC(score);
        String m ="";
        m="T120 V0 I[Piano] Rw ";
        m= printAlgorithmTST2scoreASC(score);
        return m;
    }
    public static String performAlgorithmTST2Desc(){
        String [] score = generateAlgorithmTST2score();
        //String m = printAlgorithmTST2scoreASC(score);
        String m ="";
        m="T120 V0 I[Piano] Rw ";
```

```
    m= printAlgorithmTST2scoreDESC(score);
    return m;
}
public static String performAlgorithmTST2AscsingleSTEP(int j){
    String [] score = generateAlgorithmTST2score();
    //String m = printAlgorithmTST2scoreASC(score);
    String m ="";
    m= printAlgorithmTST2scoreAscWithSTEP(score, j);
    return m;
}
public static String performAlgorithmTST2DescsingleSTEP(int j){
    String [] score = generateAlgorithmTST2score();
    //String m = printAlgorithmTST2scoreASC(score);
    String m ="";
    m= printAlgorithmTST2scoreDescWithSTEP(score, j);
    //m+="Rw A0o ";
    return m;
}
public static String performAlgorithmTST2AscallSTEPS(){
    String [] score = generateAlgorithmTST2score();
    //String m = printAlgorithmTST2scoreASC(score);
    String m ="";
    for (int j=1;j<13;j++){
    m+= printAlgorithmTST2scoreAscWithSTEP(score, j);
    m+="Rq ";
    System.out.print("Rq ");
    }
    m+="Rw A0o ";
    //m+="Rw A6w ";
    return m;
}
public static String performAlgorithmTST2DescallSTEPS(){
    String [] score = generateAlgorithmTST2score();
    //String m = printAlgorithmTST2scoreASC(score);
    String m ="";
    for (int j=13;j>0;j--){
    m+= printAlgorithmTST2scoreDescWithSTEP(score, j);
    m+="Rq ";
    System.out.print("Rq ");
    }
    m+="Rw A0o ";
    //m+="Rw A6w ";
    return m;
}
public static String performAlgorithmTST2ADscallSTEPS(){
    String [] score = generateAlgorithmTST2score();
    //String m = printAlgorithmTST2scoreASC(score);
    String m ="";
    for (int j=13;j>0;j--){
    m+= printAlgorithmTST2scoreADscWithSTEP(score, j);
    m+="Rq ";
    System.out.print("Rq ");
    }
    m+="Rw A0o ";
```

```
        //m+="Rw A6w ";
        return m;
    }
    public static String performAlgorithmTST2invAscallSTEPS(){
        String [] score = generateAlgorithmTST2score();
        //String m = printAlgorithmTST2scoreASC(score);
        String m ="";
        for (int j=1;j<13;j++){
        m+= printAlgorithmTST2scoreAscWithSTEP(score, j);
        m+= printAlgorithmTST2scoreAscWithSTEP(score, 14-j);
        m+="Rq ";
        System.out.print("Rq ");
        }
        m+="Rw A0o ";
        //m+="Rw A6w ";
        return m;
    }
    public static String performAlgorithmTST2AscallSTEPSblurred(){
        String [] score = generateAlgorithmTST2score();
        String m ="";
        for (int j=1;j<13;j++){
        m+= printAlgorithmTST2scoreWithSTEPblurred(score, j);
        m+="Rq ";
        System.out.print("Rq ");
        }
        m+="Rw A0o ";
        //m+="Rw A6w ";
        return m;
    }
    public static String performAlgorithmTST2ADscallSTEPSblurred(){
        String [] score = generateAlgorithmTST2score();
        String m ="";
        for (int j=1;j<13;j++){
        m+= printAlgorithmTST2scoreAdscWithSTEPblurred(score, j);
        m+="Rq ";
        System.out.print("Rq ");
        }
        m+="Rw A0o ";
        //m+="Rw A6w ";
        return m;
    }
    public static String [] generateAlgorithmTST2score(){
        String [] score = new String[noteCount];
        for (int i=0;i<noteCount;i++){
            score[i]=chromatic_scale_piano[i];
        }
        return score;
    }
public static String printAlgorithmTST2scoreASC(String [] score){
        String m ="";
        for (int i=0;i<score.length;i++){
            m+=score[i]+"s ";
        }
        System.out.println(m);
```

```
        return m;
    }
  public static String printAlgorithmTST2scoreDESC(String [] score){
        String m ="";
        for (int i=score.length-1;i>-1;i--){
            m+=score[i]+"s ";
        }
        System.out.println(m);
        return m;
    }
public static String printAlgorithmTST2scoreAscWithSTEP(String [] score,
int step){
        String m ="";
        for (int i=0;i<score.length/step;i++){
            m+=score[i*step]+"s ";
        }
        System.out.println(m);
        return m;
    }
public static String printAlgorithmTST2scoreADscWithSTEP(String [] score,
int step){
        String m ="";
        for (int i=0;i<score.length/step;i++){
            m+=score[i*step]+"s ";
        }

        for (int i=(score.length-1)/step;i>-1;i--){
            m+=score[i*step]+"s ";
        }
        System.out.println(m);
        return m;
    }
public static String printAlgorithmTST2scoreDescWithSTEP(String [] score,
int step){
        String m ="";
        for (int i=(score.length-1)/step;i>-1;i--){
            m+=score[i*step]+"s ";
        }
        System.out.println(m);
        return m;
    }
public static String printAlgorithmTST2scoreWithSTEPblurred(String []
score, int step){
        String m ="";
        int dev=0;

        for (int i=0;i<score.length/step;i++){
            dev=randomInteger(-5,5);
            if (i*step+dev < 0 || i*step+dev > 87) dev=0;
            m+=score[i*step+dev]+"s ";
        }
        System.out.println(m);
        return m;
    }
```

```
public static String printAlgorithmTST2scoreAdscWithSTEPblurred(String []
score, int step){
        String m ="";
        int dev=0;

        for (int i=0;i<score.length/step;i++){
            dev=randomInteger(-5,5);
            if (i*step+dev < 0 || i*step+dev > 87) dev=0;
            m+=score[i*step+dev]+"s ";
        }

        for (int i=(score.length-1)/step;i>-1;i--){
            dev=randomInteger(-5,5);
            if (i*step+dev < 0 || i*step+dev > 87) dev=0;
            m+=score[i*step+dev]+"s ";
        }
        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;
  }

}
```