Muusika

Helilõigud, MIDI, saatehääl, kolmkõlad, kvanditud heli, stereo

Klippide mängimine

Esimesest versioonist peale suudavad java programmid mängida au lihtsamas formaadis faile. Versioonist 1.2 on juurde liidetud ka muude (wav, midi) helifailitüüpide mängimise võimalus. Muusika mängimiseks tuleb luua AudioClip. Selle meetoditega play, loop ning stop saab panna klipi mängima ning mängimine katkestada. Loop tähendab, et klippi mängitakse pidevalt, s.t. pärast lõpetamist hakatakse mängimisega uuesti algusest peale.

```
import java.applet.*;
public class Muusika1 extends Applet{
 public void start(){
   AudioClip lugu=getAudioClip(getCodeBase(), "spacemusic.au");
   lugu.play();
 }
}
```

Järgneva näite puhul hakatakse korduvalt mängima lehele tulles ning lehelt lahkudes lõpetatakse mängimine.

```
import java.applet.*;
public class Muusika2 extends Applet{
 AudioClip lugu;
 public void init(){
  lugu=getAudioClip(getCodeBase(), "spacemusic.au");
 }
 public void start(){
   lugu.loop();
 }
 public void stop(){
   lugu.stop();
 }
}
```

Iseseisva programmi puhul saab klipi kätte klassi `Applet` staatilise meetodi `newAudioClip` abil, meetodi parameetriks on helifaili URL.

```
import java.applet.*;
import java.net.URL;
public class Muusika3a{
  public static void main(String argumendid[]) throws Exception{
    AudioClip lugu=Applet.newAudioClip(
      new URL("http://minitorn.tpu.ee/~jaagup/kool/java/"+
               "abiinfo/tutorial/sound/example-1dot2/bottle-open.wav")
    );
    lugu.play();
    Thread.sleep(5000);
  }
}
```

Kui soovitakse mängida lugu samast masinast, siis tuleb ka see kohalik aadress enne URLiks muuta.

```
import java.applet.*;
import java.net.URL;
public class Muusika3{
  public static void main(String argumendid[]) throws Exception{
    AudioClip lugu=Applet.newAudioClip(
      new File("spacemusic.au").toURL()
    );
    lugu.play();
    Thread.sleep(5000);
  }
}
```

MIDI

Lisaks varemvalminud lugude esitamisele saab ka ise meloodiaid kokku panna. MIDI abil saame määrata, millist nooti millal mängida. Salvestatud pillide helinäidete järgi luuakse selle tulemusena meie poolt küsitud hääl. Kui soovitakse samaaegselt kuulda mitme pilli meloodiat, tuleb need paigutada eri kanalitele. Iga kanal võib korraga teha ühe pilli häält ning harilikult on kanaleid kokku kuni 16.

Kanalist väljuvat heli saab kontrollida sinna saadetavate käskudega. Enamkasutatavad on `noteOn` ja `noteOff`. Esimene neist palub nooti mängima hakata, teine mängimise lõpetada. Meetod `noteOn` vajab kahte parameetrit: heli kõrgust ja valjust, noodi kõlamise lõpetamiseks piisab noodi numbrist. MIDI standardi järgi on igal pooltoonil oma number vahemikus 0-127. Esimese oktaavi C väärtuseks on näiteks 60, sealt saab siis vastavalt poole tooni kaupa üles ja allapoole arvutada. Valjust tähistab samuti number samast vahemikust. MIDI vahendid asuvad paketis `javax.sound`, mis kuulub standardvahendite hulka alates JDK versioonist 1.3. Operatsiooni muusikavahendite poole pöördumiseks saab kasutada klassi MidiSystem.

Üksik noot

Järgnevas näites küsitakse selle klassi kaudu helitekitamise seade ehk süntesaator ning avatakse. Viimase käest küsitakse tema külge kuuluv kanalite massiiv ning sealt omakorda kanal nr. 0. Järgnevalt palun vastaval kanalil mängida A nooti (noot nr. 69) valjusega 65. Ootan sekundi ning siis lasen noodi kõlamise lõpetada. System.exit viimase käsuna on tarvilik, kuna MIDI vahendite tarvitamisega virtuaalmasina poolt loodud lõim ei oska nootide lõpuga oma tööd lõpetada ning programm jääks viimasele reale rippuma. Analoogiline olukord on ka graafikakomponentide juures, kus programmi töö lõpetamiseks tuleb kirjutada `System.exit(0).`

```
import javax.sound.midi.*;
public class Noot{
  public static void main(String argumendid[]) throws Exception{
    Synthesizer synthesizer=MidiSystem.getSynthesizer();
    synthesizer.open();
    MidiChannel kanal=synthesizer.getChannels()[0]; //kanal 0;
    int korgus=69; //A
    int valjus=65; //keskmine
    kanal.noteOn(korgus, valjus);
    Thread.sleep(1000);
    kanal.noteOff(korgus);
    System.exit(0);
  }
}
```

Kromaatiline heliredel

Kromaatilist heliredelit võib mängida tsükli abil:

```
for(int i=40; i<120; i++){
      kanal.noteOn(i, 60);
      Thread.sleep(200);
      kanal.noteOff(i);
}
```

Kui soovida, et samal kanalil mängiks korraga mitu nooti, tuleb lihtsalt ükshaaval määrata, millised helikõrgused peavad kõlama. Kõikide mängimise saab korraga lõpetada käsuga `allNotesOff ().`

Helikõrguse ujumine

Kuigi MIDI puhul öeldakse helikõrgus numbriga ette, on ka siin võimalik toonil ujuda lasta. Seda saab käsuga `setPitchBend`, andes ette numbri, palju kõrgust muuta. Vaikimisi väärtuseks on 8192, sellisel juhul vastab noodi number tema helikõrgusele. Iga number sellest ülespoole viib

helikõrgust kõrgemale, allapoole aga madalamaks. Vaikiva kokkuleppe järgi tähistab number 0 tooni võrra madalamat ning 16363 tooni jagu kõrgemat heli, kuid see kõikumise piirkond võib ka erineda. Järgnevas näites peaks tooni ülalt alla ujumine kuulda olema.

```
    kanal.noteOn(60, 70);
    kanal.noteOn(64, 70);
    for(int korgus=16383; korgus>0; korgus-=500){
      Thread.sleep(200);
      kanal.setPitchBend(korgus);
    }
    kanal.allNotesOff();
```

Pillide loetelu

Kanalil mängivat instrumenti saab muuta käsuga `programChange`, andes parameetritena ette uue pilli helipanga ning panga sees sellele pillile vastava programmijupi järjenumbri. Süntesaatorile kättesaadavad pillid saab küsida `getDefaultSoundbank().getInstruments()` abil. All näites paiknevad trükitakse tsüklis järgemööda välja pillide nimed ning mängitakse igal pillil noot.

```
Instrument[] pillid=synthesizer.getDefaultSoundbank().
                getInstruments();
    MidiChannel kanal=synthesizer.getChannels()[0];
    for(int i=0; i<pillid.length; i++){
      System.out.println(i+": "+pillid[i]);
      kanal.programChange(pillid[i].getPatch().getBank(),
                          pillid[i].getPatch().getProgram());
      kanal.noteOn(60, 50);
      Thread.sleep(500);
      kanal.noteOff(60);
    }
```

osa väljundist:

```
12: Instrument Marimba (bank 0 program 12)
13: Instrument Xylophone (bank 0 program 13)
14: Instrument Tubular Bell (bank 0 program 14)
15: Instrument Dulcimer (bank 0 program 15)
16: Instrument Hammond Organ (bank 0 program 16)
```

Rajad

Kanalitele käske andes saab edukalt mängida programmi sees üksikuid noote ja luua kõlaefekte, kuid meloodiate ja viisijuppide esitamiseks on loodud täiendavad abivahendid: sekventser, sekventsid ja rajad. Rajal (track) on kirjas saadetavad teated koos ajatemplitega. Ühele rajale võib kanda näiteks ühe pillimehe mängitavad noodid. Sekvents on radade kogumik (nagu partituur). Sekventser on programmilõik, kes sekventsi kirjutatud käsklused õigel ajal süntesaatorile heli tekitamiseks edasi saadab.

Kanalile saadetava teate hoidmiseks on `ShortMessage`. Teate sisuks võib olla nii noodi mängimise algus, selle lõpp, kanalil oleva instrumendi vahetus kui muudki, näiteks klahvivajutuse tugevuse muutus. Rajale lisamiseks tuleb teatele ümber panna MidiEvent, mis lisab sinna aja – MIDI ajaühikutes mõõdetava vahemiku alates raja algushetkest. Esimeseks teateks kanalil peab olema PROGRAM_CHANGE, selle abil määratakse, millise pilliga hakatakse vastaval kanalil häält tegema. Siin näites määratakse kanalile 0 pill number 16, ehk praeguse helipanga järgi Hammondi orel.

Sekventsi puhul tuleb määrata, milliselt seal aega arvatakse: kas kaadrite või löökide järgi. Esimest võimalust kasutatakse video kõrvale heli loomiseks, teist nootide ja taktide järgi lugu seades, viimast tähistab Sequence.PPQ nagu siin näites. Käsk `new Sequence (Sequence.PPQ, 4)` tähistab, et luuakse nootidepõhine sekvents, mille iga löök (ehk veerandnoot 2/4, 3/3 ja 4/4 taktimõõdus) jagatakse neljaks tiksuks. Tiks on vähim ajaühik MIDI

mõõdustikus, seega sellise jaotuse korral on võimalik lühimaks kestuseks panna kuueteistkümnendiknoodid. Kui tahta näiteks ka kolmekümnekahendikke kasutada, siis tuleks algselt veerandnoot mitte neljaks vaid kaheksaks tiksuks jagada.

Enne rajale sündmuste lisamist tuleb rada luua sekventsi käsuga `createTrack()`. Tulemuse kuulamiseks on vaja `MidiSystem`'i käest küsida sekventser, see avada. Siis määrata, et sekventseri poolt mängitavaks sekventsiks oleks (praegu meie üherajaline) sekvents ning käsuga start panna sekventser tööle.

```
import javax.sound.midi.*;
public class Rada1{
 public static void main(String argumendid[]) throws Exception{
         ShortMessage lahti = new ShortMessage();
         ShortMessage kinni = new ShortMessage();
         ShortMessage algus = new ShortMessage();
         algus.setMessage(ShortMessage.PROGRAM_CHANGE, 0, 16, 0);
         lahti.setMessage(ShortMessage.NOTE_ON, 0, 65, 93);
         kinni.setMessage(ShortMessage.NOTE_OFF, 0, 65, 93);
         Sequence sequence=new Sequence(Sequence.PPQ, 4);
         Track track=sequence.createTrack();
         track.add(new MidiEvent(algus, 0));
         track.add(new MidiEvent(lahti, 0));
         track.add(new MidiEvent(kinni, 4));
         track.add(new MidiEvent(lahti, 8));
         track.add(new MidiEvent(kinni, 11));
         track.add(new MidiEvent(lahti, 12));
         track.add(new MidiEvent(kinni, 15));
         track.add(new MidiEvent(lahti, 16));
         track.add(new MidiEvent(kinni, 31));
         Sequencer sequencer=MidiSystem.getSequencer();
         sequencer.open();
         sequencer.setSequence(sequence);
         sequencer.start();
  }
}
```

Loodud sekvetserile saab seada mitmeid parameetreid, samuti küsida andmeid mängitava loo kohta. Kui tahan määrata, et loo mängimise kiirus on 40 lööki minutis, siis tuleb kirjutada
`sequencer.setTempoInBPM(40);`
Kuna ennist olin määranud, et ühes löögis on 4 tiksu, siis eelmainitud reast järeldub, et igas minutis on 4*40 ehk 160 tiksu ning ühe tiksu pikkuseks on 60/160 sekundit. Analoogiliselt on võimalik määrata, millisest tiksust alates edasi mängitakse. Sekventseri käest saab küsida mängimise kiirust, temas sisalduva sekventsi ehk lõigu pikkust tiksudes, parasjagu mängitava tiksu numbrit ja muudki.

Kui soovin, et lõiku mitu korda järjest mängitaks, siis tuleb sekventserile panna külge kuular, mis lõputeate saabumisel paluks lõiku taas otsast mängima hakata. Piiritleja final on sekventseri ette toodud seetõttu, et teda saaks kohaliku muutujana sisemises klassis kasutada. Final ei luba muutujal vahetada isendit millele osutada (st., sellele muutujale ei saa anda uut väärtust). Muul juhul võiks juhtuda, et vahepeal pannakse sekventserile uus väärtus ning sisemise klassi sees lükatakse vale sekventser käima. Kui anonüümses sisemises klassis kasutada isendi (või klassi) tarvis kirjeldatud muutujat, siis seda probleemi ei teki.

Kordamine

Lõigu lõppu näitab kuularisse saabuv teade tüübiga nr. 47. Kui seepeale palun sekventseril uuesti otsast mängima hakata, siis tundub kasutajale, nagu lugu kordaks end järjepanu.

```
final Sequencer sequencer=MidiSystem.getSequencer();
  sequencer.open();
  sequencer.setSequence(sequence);
  sequencer.start();
  sequencer.addMetaEventListener(new MetaEventListener(){
  public void meta(MetaMessage m){
    //kui lugu läbi, siis alustatakse uuesti
    if(m.getType()==47)sequencer.start();
  }
```

```
});
```

Kui soovida, et sekventsi kordamisel erinevalt mängitaks, siis võib sinna panna mitu rada ning igal korral määrata, milliseid radasid mängitakse, milliseid mitte. Sekventseri käsu `setTrackMute` abil saab määrata, kas rada on tumm või mitte. Kui soovitakse, et ainult üks rada mängiks ning teised oleksid vait, siis tuleb see rada panna soleerima käsu `setTrackSolo` abil.

MIDI faili mahamängimine

... on lihtne, sest vastavad vahendid on küllaltki valmiskujul kättesaadavad. Kui arvutis on MIDI väljund (helikaardi kaudu) kõlaritesse või kõrvaklappidesse saadetud ning muusikafail ilusti kettal olemas, siis peaks all näha oleva nelja käsu abil olema võimalik etteantud nimega failist muusikat kuulata. Klassi MidiSystem käsk getSecuence võimaldab sekventsi lugeda failist (või mõnest muust voost). Kui avatud sekventser määrata vastavat sekventsi mängima, siis võimegi kuulata, mis faili salvestatud on.

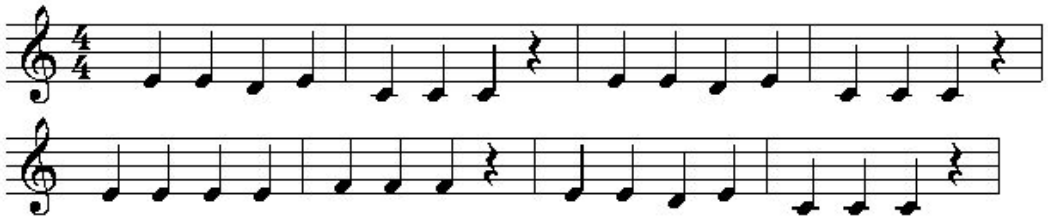
```
import javax.sound.midi.*;
public class Midimangija1 {
  public static void main(String argumendid[]) throws Exception{
    Sequencer sekventser=MidiSystem.getSequencer();
    sekventser.open();
    sekventser.setSequence(MidiSystem.getSequence(new java.io.File("koduke.mid")));
    sekventser.start();
  }
}
```

MIDI failis paiknevate andmete kohta võib muudki teada saada: andmete pikkust mikrosekundites ning tiksudes, radade arvu ning teated ükshaaval radade kaupa. Rajalt saab käsuga get kätte järjekorranumbri järgi MidiEvent'i. Sealt edasi getMessage väljastab teate sisu ning getTick tiksu (ajahetke), millal vastav teade süntesaatorile saadetakse. Tuleb vaadata, millist tüüpi teatega on tegemist ning vastavalt käituda. Helidega seotud teated on tüübist ShortMessage. Iga teate juures on täisarvuna kirjas kanal, käsklus ning kaks teatebaiti. Nende baitide interpretatsioon sõltub sellest, millise käsuga on tegemist.

```
import javax.sound.midi.*;
import java.util.*;
public class Midimangija2 {
  public static void main(String argumendid[]) throws Exception{
    Sequence sekvents=MidiSystem.getSequence(new java.io.File("koduke.mid"));
    System.out.println(sekvents.getDivisionType()+" pikkus: "+sekvents.getMicrosecondLength()/1000000.0+
                  " sekundit, "+sekvents.getTickLength()+" tiksu");
    Track[] rajad=sekvents.getTracks();
    System.out.println(rajad.length+" rada");
    for(int nr=0; nr<rajad.length; nr++){
      System.out.println("Rada "+nr);
      for(int t=0; t<rajad[nr].size(); t++){
        MidiEvent me=rajad[nr].get(t);
        long tiks=me.getTick();
        MidiMessage teade=me.getMessage();
        if(teade instanceof ShortMessage){
          ShortMessage sm=(ShortMessage)teade;
          System.out.println(tiks+" ShortMessage "+
            "kanal: "+sm.getChannel()+
            " teade: "+sm.getCommand()+  //näiteks noteOff
            " bait1: "+sm.getData1()+    //kõrgus
            " bait2: "+sm.getData2()     //valjus
          );
        }
        if(teade instanceof MetaMessage){
          MetaMessage mm=(MetaMessage)teade;
          System.out.print(tiks+" MetaMessage"+
           " tyyp: "+mm.getType());
          byte[] b=mm.getData();
          for(int i=0; i<b.length; i++)System.out.print(" "+b[i]);
          System.out.println();
        }
      }
```

```
    }
  }
}

/*

Väljund:

0.0 pikkus: 15.5 sekundit, 2976 tiksu
2 rada
Rada 0
0 MetaMessage tyyp: 88 4 2 24 8
0 MetaMessage tyyp: 89 0 0
0 MetaMessage tyyp: 81 7 -95 32
2976 MetaMessage tyyp: 47
Rada 1
0 ShortMessage kanal: 0 teade: 192 bait1: 1 bait2: 0
0 ShortMessage kanal: 0 teade: 176 bait1: 7 bait2: 80
0 ShortMessage kanal: 0 teade: 144 bait1: 64 bait2: 80
96 ShortMessage kanal: 0 teade: 144 bait1: 64 bait2: 0
96 ShortMessage kanal: 0 teade: 144 bait1: 64 bait2: 80
192 ShortMessage kanal: 0 teade: 144 bait1: 64 bait2: 0
192 ShortMessage kanal: 0 teade: 144 bait1: 62 bait2: 80
288 ShortMessage kanal: 0 teade: 144 bait1: 62 bait2: 0
288 ShortMessage kanal: 0 teade: 144 bait1: 64 bait2: 80
384 ShortMessage kanal: 0 teade: 144 bait1: 64 bait2: 0
384 ShortMessage kanal: 0 teade: 144 bait1: 60 bait2: 80
480 ShortMessage kanal: 0 teade: 144 bait1: 60 bait2: 0
480 ShortMessage kanal: 0 teade: 144 bait1: 60 bait2: 80
576 ShortMessage kanal: 0 teade: 144 bait1: 60 bait2: 0
576 ShortMessage kanal: 0 teade: 144 bait1: 60 bait2: 80
672 ShortMessage kanal: 0 teade: 144 bait1: 60 bait2: 0
768 ShortMessage kanal: 0 teade: 144 bait1: 64 bait2: 80
864 ShortMessage kanal: 0 teade: 144 bait1: 64 bait2: 0
864 ShortMessage kanal: 0 teade: 144 bait1: 64 bait2: 80
960 ShortMessage kanal: 0 teade: 144 bait1: 64 bait2: 0
960 ShortMessage kanal: 0 teade: 144 bait1: 62 bait2: 80
1056 ShortMessage kanal: 0 teade: 144 bait1: 62 bait2: 0
1056 ShortMessage kanal: 0 teade: 144 bait1: 64 bait2: 80
1152 ShortMessage kanal: 0 teade: 144 bait1: 64 bait2: 0
1152 ShortMessage kanal: 0 teade: 144 bait1: 60 bait2: 80
1248 ShortMessage kanal: 0 teade: 144 bait1: 60 bait2: 0
1248 ShortMessage kanal: 0 teade: 144 bait1: 60 bait2: 80
1344 ShortMessage kanal: 0 teade: 144 bait1: 60 bait2: 0
1344 ShortMessage kanal: 0 teade: 144 bait1: 60 bait2: 80
1440 ShortMessage kanal: 0 teade: 144 bait1: 60 bait2: 0
1536 ShortMessage kanal: 0 teade: 144 bait1: 64 bait2: 80
1632 ShortMessage kanal: 0 teade: 144 bait1: 64 bait2: 0
1632 ShortMessage kanal: 0 teade: 144 bait1: 64 bait2: 80
1728 ShortMessage kanal: 0 teade: 144 bait1: 64 bait2: 0
1728 ShortMessage kanal: 0 teade: 144 bait1: 64 bait2: 80
1824 ShortMessage kanal: 0 teade: 144 bait1: 64 bait2: 0
1824 ShortMessage kanal: 0 teade: 144 bait1: 64 bait2: 80
1920 ShortMessage kanal: 0 teade: 144 bait1: 64 bait2: 0
1920 ShortMessage kanal: 0 teade: 144 bait1: 65 bait2: 80
2016 ShortMessage kanal: 0 teade: 144 bait1: 65 bait2: 0
2016 ShortMessage kanal: 0 teade: 144 bait1: 65 bait2: 80
2112 ShortMessage kanal: 0 teade: 144 bait1: 65 bait2: 0
2112 ShortMessage kanal: 0 teade: 144 bait1: 65 bait2: 80
2208 ShortMessage kanal: 0 teade: 144 bait1: 65 bait2: 0
2304 ShortMessage kanal: 0 teade: 144 bait1: 64 bait2: 80
2400 ShortMessage kanal: 0 teade: 144 bait1: 64 bait2: 0
2400 ShortMessage kanal: 0 teade: 144 bait1: 64 bait2: 80
2496 ShortMessage kanal: 0 teade: 144 bait1: 64 bait2: 0
2496 ShortMessage kanal: 0 teade: 144 bait1: 62 bait2: 80
2592 ShortMessage kanal: 0 teade: 144 bait1: 62 bait2: 0
2592 ShortMessage kanal: 0 teade: 144 bait1: 64 bait2: 80
2688 ShortMessage kanal: 0 teade: 144 bait1: 64 bait2: 0
2688 ShortMessage kanal: 0 teade: 144 bait1: 60 bait2: 80
2784 ShortMessage kanal: 0 teade: 144 bait1: 60 bait2: 0
2784 ShortMessage kanal: 0 teade: 144 bait1: 60 bait2: 80
2880 ShortMessage kanal: 0 teade: 144 bait1: 60 bait2: 0
2880 ShortMessage kanal: 0 teade: 144 bait1: 60 bait2: 80
2976 ShortMessage kanal: 0 teade: 144 bait1: 60 bait2: 0
2976 MetaMessage tyyp: 47

*/
```

Nagu näha, seda MIDI faili loonud programm on kasutanud noodi kõlamise alustamiseks ning lõpetamiseks sama käsku (NOTE_ON, 144), vaid mängimise lõpetamise puhul on noodile määratud valjus 0. Kanalile 0 pole muusika kõlamisega seotud noote pandud.



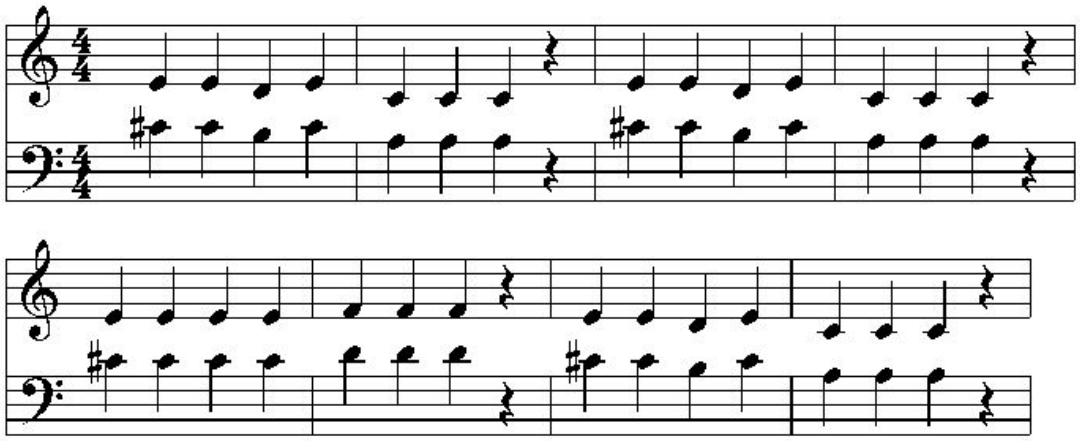
MIDI faili loomine

Sekventsi võib faili kirjutada ühe käsuga, andes ette MIDI formaadi (0 või 1) ning voo, kuhu andmed saata.

```
MidiSystem.write(sekvents, 1, new FileOutputStream("koduke2.mid"));
```

Nii võib rahumeeli lasta programmil loo välja mõtelda või kasutaja käest küsida ning siis kettale salvestada, et seda edaspidi kuulata või uuesti muuta. All näites on võetud ette lugu (koduke.mid), sellele lisatud rada ning algse viimase (siin ainukese) raja nootide järgi pandud uuele rajale samad noodid väikese tertsi (3 pooltooni) jagu allapoole. ShortMessage'd on ümber arvutatud, muud muutumatuna üle kantud.

```
import javax.sound.midi.*;
import java.io.*;
public class Midimangija3 {
  public static void main(String argumendid[]) throws Exception{
    Sequence sekvents=MidiSystem.getSequence(new File("koduke.mid"));
    Track algrada=sekvents.getTracks()[sekvents.getTracks().length-1]; //viimane rada
    Track uusrada=sekvents.createTrack();
    int nihe=-3;
    for(int nr=0; nr<algrada.size(); nr++){
      MidiEvent me=algrada.get(nr);
      if(me.getMessage() instanceof ShortMessage){
        ShortMessage sm=(ShortMessage)me.getMessage();
        ShortMessage sm2=new ShortMessage();
        if(sm.getCommand()==ShortMessage.NOTE_ON ||
             sm.getCommand()==ShortMessage.NOTE_OFF){
          sm2.setMessage(
            sm.getCommand(), sm.getChannel(), sm.getData1()+nihe, sm.getData2()
          );
        } else {
          sm2=(ShortMessage)sm.clone();
        }
        uusrada.add(new MidiEvent(sm2, me.getTick()));
      } else {
        uusrada.add(me);
      }
    }
    MidiSystem.write(sekvents, 1, new FileOutputStream("koduke2.mid"));
  }
}
```



Saateautomaat

Küllap olete näinud ja kuulnud süntesaatorit lugusid saatmas. Mõni koputab lihtsalt rütmi taha, teisele tuleb ette anda helistik ning selle peale mängitakse saateakordi. Leidub ka selliseid saateprogramme, mis vastavalt etteantud helistikule ise saatenoote juurde mõtlevad või lisaks sellele juba mängitava viisi pealt harmoonia ennustavad. Viimase võimaluse jätame siin välja, kuid juurde kuuluva pikema näite ja seletuse läbi töötanud lugeja peaks mõningase pusimise järel esimese kolme osaga küll hakkama saama.

Näites mängitakse saadet Juhansonide Unenäo laulule. Kasutaja saab määrata helistiku ning kuulata bassi, akorde ning taustaks mängitavat rahvalikku viiulipartiid, samuti määrata tempot. Koodi lühiduse ja kompaktsuse huvides pole viisi lisatud, kuid olles seletustest aru saanud, peaks see täiesti jõukohane olema.

Üheaegselt mängitavad rajad luuakse valmis ja pannakse kokku üheks sekventsiks. Rajad, mille heli parajasti kuulda ei soovita, pannakse vaikima. Kuna ühes helistikus mängimisel piirdub lugu kolme duuriga, siis arvutatakse nii bassi kui akordi tarvis välja rajad kõigi kolme duuri jaoks, kuid kõlada lastakse vaid siis, kui kasutaja vastavat instrumenti soovib kõlamas kuulda ning parajasti mängitav duur ja rada kokku langevad. Nii pääseb pidevast nootide arvutamisest ning uued rajad tuleb luua vaid helistiku vahetamisel. Vaid viiulipartii luuakse iga takti algul uuesti, et see tunduks värske ja kordumatuna.

Bassi partii on kõige lihtsam, sestap alustan selle loomise seletamisest. Takti jooksul on bassipartiis vaid üks noot, mängitava akordi esimene aste, mis kõlab kogu takti jooksul. Takt koosneb kolmest löögist, iga löök neljast tiksust (see määrati sekventsi loomisel). Nii peab bass alustama kõlamist tiksust 0 ning lõpetama selle viimase tiksu järel ehk kaheteistkümnenda tiksu alguses. Selle ühe noodi kõlamiseks on vaja rajale kolme teadet: tuleb määrata kanalil kõlama hakkav pill, kõlamise algus ja lõpp. Bass pannakse mängima etteantud põhinoodist 2 oktaavi ehk 24 pooltooni allpoolt.

```
void looBass(Track t, int toonika){
  try{
```

Kõigepealt tehakse tühjad teated valmis

```
   ShortMessage m[]=new ShortMessage[3];
   for(int i=0; i<m.length; i++){
     m[i]=new ShortMessage();
   }
```

seejärel antakse neile väärtused

```
   m[0].setMessage(ShortMessage.PROGRAM_CHANGE, 1, 39, 0);
   m[1].setMessage(ShortMessage.NOTE_ON, 1, toonika-24, 60);
   m[2].setMessage(ShortMessage.NOTE_OFF, 1, toonika-24, 60);
```

ning lõpuks pannakse rajale, lisades juurde, mitmenda tiksu juures nad aktiivseks peavad muutuma.

```
   t.add(new MidiEvent(m[0], 0));
   t.add(new MidiEvent(m[1], 0));
   t.add(new MidiEvent(m[2], 12));
  }catch(Exception e){e.printStackTrace();}
 }
```

Kolmkõla loomine toimub tehniliselt sarnaselt bassiga võrrelduna. Kui bassi partii koosnes takti jooksul vaid ühest noodist, siis kolmkõla tarvis kõlavad takti jooksul kolm nooti. Esimesel löögil alustatakse põhinoodiga, teisel kolmanda ning kolmandal viienda astmega. Kolmanda löögi lõpus lõpetatakse kõikide nootide kõlamine. Endise kolme teate asemel on nüüd vaja seitset: instrumendi määramiseks ning kõigi kolme noodi kõlamise alustamiseks ja lõpetamiseks.

Et võiks korraga välja arvutada ühe helistiku juures vaja minevad kolmkõla- ning bassirajad, selleks sai loodud meetod, millele antakse ette toonika ning mis väljastab sekventsi kummagi esituse radadega esimese, neljanda ning viienda astme tarvis. Kolmkõlade juures võetakse nii neljas kui viies aste (vastavalt viis ja seitse pooltooni) toonikast üles, bassi puhul alla. Tegemist on täiesti sisetundega, milliselt poolt võtta, kuid siin tundus, et helilõigu keskele sobib sügav mahlakas bass paremini ning lõpus kõrgemas toonikas justkui saabub rahu ja tasakaal. Põhjalikumate saadete puhul tuleb leidmise algoritm tõenäoliselt keerulisemaks teha, et arvestataks ilusti kõlavaid absoluutkõrgusi ning miks mitte ka viisi kulgemist.

```
Sequence pohikolmkolad(int toonika) throws InvalidMidiDataException{
   Sequence s1=new Sequence(Sequence.PPQ, 4);
   looKolmkola(s1.createTrack(), toonika);
   looKolmkola(s1.createTrack(), toonika+5); //kvart
   looKolmkola(s1.createTrack(), toonika+7); //kvint
   looBass(s1.createTrack(), toonika);
   looBass(s1.createTrack(), toonika-7); //IV ehk kvint alla
   looBass(s1.createTrack(), toonika-5);
   return s1;
 }
```

Et parasjagu valitud helistiku põhikolmkõladeks saatehääled kätte saada, ka selleks on eraldi — kuigi lühike — alamprogramm.

```
 Sequence looSekvents() throws InvalidMidiDataException{
   return pohikolmkolad(helikorgused[helistik.getSelectedIndex()]);
 }
```

Nime all helistik peitub valik ning helistik.getSelectedIndex() annab järjekorranumbri, mitmenda kasutaja oli loetelust valinud. Massiivis helistikud on tähtedega määratud, mitmendal kohal mingi helistik asetseb ning massiiv helikorgused näitab vastavate helistike toonikate asukohad MIDI skaalal. Sõnes jooksevHelistik hoitakse meeles viimati valitud helistiku nimi, et edaspidi oleks võrdlemisel teada, millal kasutaja on helistikku vahetanud, et oleks põhjust saatesekvents uuesti arvutada.

```
 JComboBox helistik=new JComboBox();
 String[] helistikud={"Bb", "F", "C", "G", "D", "A", "E"};
  int[] helikorgused={58, 53, 60, 55, 50, 57, 52};
  String jooksevHelistik="";
```

Mängitav sekvents ning mängiv sektventser,

```
  Sequence sequence;
  Sequencer sequencer;
```

graafilised vahendid töö juhtimiseks,

```
  JButton nupp=new JButton("Mängi");
  JCheckBox ruut=new JCheckBox("Korda");
  JCheckBox bass=new JCheckBox("Bass");
  JCheckBox akord=new JCheckBox("Akord");
  JCheckBox taust=new JCheckBox("Taust");
  JRadioButton[] raadionupud=new JRadioButton[3];
  String[] raadionupustring={"I", "IV", "V"};
  JScrollBar tempo=new JScrollBar(JScrollBar.HORIZONTAL, 190, 5, 40, 320);
```

rada viiulipartii paigutamiseks.

```
  Track muutuvRada;
```
Diatoonilise duuri astmetele vastavate pooltoonide arv, et hiljem oleks võimalik määrata, mitmendale astmele liikuda ning see arvutile pooltoonides selgeks teha.
```
static final int[] noodivahed={-1, 0, 2, 4, 5, 7, 9, 11, 12};
  //toonika kohal 1 väärtus 0
```
Konkreetse loo duuride järjestus.
```
  int[] duurid={0, 1, 2, 0, 1, 2, 0, 0};  // Juhansonide unenäo laul
                                          // 0- toonika, 1-IV, 2-V
  int duurinr=0; //mängitava takti järjekorranumber
```

Taustaks mängib viiul lihtsa algoritmi järgi: alustatakse kõlava duuri esimeselt astmelt ning liigutakse iga noodiga sellelt ühe võrra kas üles või alla juhuslikult võetuna. Sellisena satub rõhuline noot kokku kõlava akordiga ning taustaks kõlab ühtlane meloodiline saagimine. Veidi parandades ning ebaloogilisi järske üleminekuid vältides annaks loomulikkust suurendada, kuid ka sellisel kujul ei riiva hullusti kõrva ning viiulimängu algusest alustades tuleks tükk harjutada, et selliselegi tulemusele jõuda.

Et algoritm arvestab diatooniliste astmetega, MIDI aga loeb kõik pooltoonid samaväärseteks siis on loodud ümberarvutamiseks eraldi alamprogramm, millele antakse ette algne aste ja soovitav nihe ning mis väljastab vahe pooltoonides, lubades ka ühe oktaavi piirest välja nihkuda.

```
static int pooltoonid(int aste, int muutus){
   int loppaste=aste+muutus;
   int okt=loppaste/7;
   if(loppaste<0)okt--;
   loppaste=loppaste-7*okt;
   int vahe=noodivahed[loppaste]-noodivahed[aste];
   return vahe+12*okt;
}
```

Meetodile enesele antakse ette rada, kuhu teated panna, toonikanoodinumber ning aste, millelt mängimist alustada.

```
void looTaustaviiul(Track t, int toonika, int aste){
 try{
```

Kõigepealt luuakse mäluruum arvutustulemuste paigutamiseks soovitud arvu leitud nootide tarvis,
```
   int nootidearv=6;
   int[] samm=new int[nootidearv];
   int i=0;
   samm[i++]=0;
```
siis arvutatakse eelpool kirjeldatud algoritmi järgi väärtused nootidele, kusjuures iga järgmine asub eelmise kõrval
```
   while(i<nootidearv){
     if(Math.random()<0.5){
       samm[i]=samm[i-1]-1;
     } else {
       samm[i]=samm[i-1]+1;
     }
     i++;
   }
```
ning lõpuks asendatakse leitud astmed MIDI teadetega. Ette nagu ikka pilli määramine, sedakorda kanalile number 2.
```
   ShortMessage m=new ShortMessage();
   m.setMessage(ShortMessage.PROGRAM_CHANGE, 2, 41, 0);
   t.add(new MidiEvent(m, 0));
   for(i=0; i<nootidearv; i++){
     m=new ShortMessage();
```
Iga helikõrguse arvutamiseks liidetakse kokku helistiku põhikõrgus (toonika), pooltoonide arv kõlava duuri põhitoonini jõudmiseks (noodivahed[aste]) ning takti jooksul sellest nihkutud astmete arv pooltoonidena (pooltoonid(aste, samm[i])).
```
     m.setMessage(ShortMessage.NOTE_ON, 2,
        toonika+noodivahed[aste]+pooltoonid(aste, samm[i]), 60);
     t.add(new MidiEvent(m, i*2));
     m=new ShortMessage();
     m.setMessage(ShortMessage.NOTE_OFF, 2,
        toonika+noodivahed[aste]+pooltoonid(aste, samm[i]), 60);
     t.add(new MidiEvent(m, (i+1)*2));
   }
  }catch(Exception e){e.printStackTrace();}
 }
```

Hoolitsemaks, et mängitaks vaid nendel radadel, mille kuulamist kasutaja ootab, tuleb ülejäänud vaikima määrata. Sobivat duuri näitab raadionupp. Siin näites määrab selle valitu programm, kuid kergesti võib ka inimesel lasta määrata, mitmenda astme kolmkõla ta soovib saateks kuulata.

```
void paneSoolo(){
   for(int i=0; i<raadionupud.length; i++){
     sequencer.setTrackMute(i, !raadionupud[i].isSelected() || !akord.isSelected());
     sequencer.setTrackMute(3+i, !raadionupud[i].isSelected() || !bass.isSelected());
   }
```

Taustaviiul jäetakse kõlama vaid siis, kui vastav ruut on märgitud.

```
   sequencer.setTrackMute(6, !taust.isSelected());
 }
```

Iga uue takti mängimisel tuleb teha mõned ettevalmistused.

```
 void alusta(){
  try{
```

Kui kasutaja on helistikku vahetanud, tuleb kogu sekvents uuesti arvutada.

```
  if(!jooksevHelistik.equals(helistik.getSelectedItem())){
   sequence=looSekvents();
   jooksevHelistik=helistik.getSelectedItem().toString();
  }
```

Kui tegemist pole loo algusega, siis tuleb kustutada eelmises taktis mängitud viiulipartii

```
   if(muutuvRada!=null)sequence.deleteTrack(muutuvRada);
```

ning luua uus rada uue partii paigutamiseks.

```
   muutuvRada=sequence.createTrack();
   int pohitoon=helikorgused[helistik.getSelectedIndex()];
   int aste=1;
   if (duurid[duurinr]==1)aste=4;
   if (duurid[duurinr]==2)aste=5;
```

ja arvutada sinna noodid.

```
   looTaustaviiul(muutuvRada, pohitoon, aste);
   sequencer.setSequence(sequence);
```

Mängitava duuri raadionupp märgistada,

```
   raadionupud[duurid[duurinr]].setSelected(true);
```

leida järgmise korra jaoks järgneva takti number või suunata lugu algusse tagasi

```
   duurinr=(duurinr+1)%duurid.length;
```

ning lõpuks väljundisse lugu mängima hakata.

```
   mangi();
  }catch(Exception e){e.printStackTrace(); System.out.println(e);}
 }
```

Mängimise tarvis tuleb

```
void mangi(){
```

kõigepealt sekventser tööle panna

```
   sequencer.start();
```

alles seejärel saab määrata, millised rajad vaikivad

```
   paneSoolo();
```

ning kui kiiresti muusika liigub.

```
   sequencer.setTempoInBPM(tempo.getValue());
 }
```

Siia meetodisse saadetakse teade sekventsi lõppemise kohta. Kui kasutaja soovib jätkamist, hakatakse uuesti otsast peale, muul juhul lõpetatakse mängimine ning vabastatakse ressursid teiste programmide tarvis.

```
  public void meta(MetaMessage m){
    if(m.getType()==47 && ruut.isSelected()){
      alusta();
    } else {
      sequencer.close();
    }
  }
```

Töölesaamise käivitusprogramm on ikka standardne. Tuleb vaid hoolitseda, et init-meetod käima pandaks. Rakendina käivitamisel jääb siinne staatiline meetod sootuks täitmata. Sõltuvalt masina

jõudlusest ja konfiguratsioonist võib mõnikord kohalikult kettalt iseseisva programmi või rakendina siinse saateprogrammi käivitamine sujuvama tulemuse ande kui üle võrgu tööle panek, sest sel juhul kipub vähemalt aeglasemate masinate korral õiguste kontrolliks kuuldavalt aega kuluma.

```
  public static void main(String argumendid[]){
    JFrame f=new JFrame("Saade");
    Noodirakend9a ap=new Noodirakend9a();
    ap.init();
    f.getContentPane().add(ap);
    f.pack();
    f.setVisible(true);
  }
```

Siit lehekülgedelt aga peaks olema võimalik leida piisavalt ideid ja soovitusi süntesaatori programmeerimiseks, et mõningane maitse suhu saada ning muusikateadmisi ja keerulisematel juhtudel ka raamatuid appi võttes saaks midagi täiesti kasutatavat kokku panna.

```
import javax.sound.midi.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;


public class Noodirakend9a extends JApplet implements ActionListener, MetaEventListener{
  Sequence sequence;
  Sequencer sequencer;
  JButton nupp=new JButton("Mängi");
  JCheckBox ruut=new JCheckBox("Korda");
  JCheckBox bass=new JCheckBox("Bass");
  JCheckBox akord=new JCheckBox("Akord");
  JCheckBox taust=new JCheckBox("Taust");
  JComboBox helistik=new JComboBox();
  JRadioButton[] raadionupud=new JRadioButton[3];
  String[] raadionupustring={"I", "IV", "V"};
  JScrollBar tempo=new JScrollBar(JScrollBar.HORIZONTAL, 190, 5, 40, 320);
  Track muutuvRada;
  String[] helistikud={"Bb", "F", "C", "G", "D", "A", "E"};
  int[] helikorgused={58, 53, 60, 55, 50, 57, 52};
  String jooksevHelistik="";
  static final int[] noodivahed={-1, 0, 2, 4, 5, 7, 9, 11, 12};  //toonika kohal 1 väärtus 0
  int[] duurid={0, 1, 2, 0, 1, 2, 0, 0};  // Juhansonide unenäo laul
                                          // 0- toonika, 1-IV, 2-V
  int duurinr=0; //takti järjekorranumber

  void looKolmkola(Track t, int toonika){
  try{
   ShortMessage m[]=new ShortMessage[7];
   for(int i=0; i<m.length; i++){
     m[i]=new ShortMessage();           // Tühjad teated valmis
   }
   m[0].setMessage(ShortMessage.PROGRAM_CHANGE, 0, 25, 0);  //pill nr 25 rajale 0
   m[1].setMessage(ShortMessage.NOTE_ON, 0, toonika, 60);
   m[2].setMessage(ShortMessage.NOTE_ON, 0, toonika+4, 60); //terts rajale 0 valjusega 60
   m[3].setMessage(ShortMessage.NOTE_ON, 0, toonika+7, 60); //kvint
   m[4].setMessage(ShortMessage.NOTE_OFF, 0, toonika, 60);
   m[5].setMessage(ShortMessage.NOTE_OFF, 0, toonika+4, 60);
   m[6].setMessage(ShortMessage.NOTE_OFF, 0, toonika+7, 60);
    t.add(new MidiEvent(m[0], 0));
   for(int i=1; i<4; i++){
     t.add(new MidiEvent(m[i], (i-1)*4));  // nelja tiksu (ühe löögi) tagant
   }                                       // noodid sisse
   for(int i=4; i<7; i++){
     t.add(new MidiEvent(m[i], 12));       // 12nda tiksu juures vaikus
   }
  }catch(Exception e){
    e.printStackTrace();
  }
 }


 void looBass(Track t, int toonika){
  try{
   ShortMessage m[]=new ShortMessage[3];
   for(int i=0; i<m.length; i++){
     m[i]=new ShortMessage();
   }
   m[0].setMessage(ShortMessage.PROGRAM_CHANGE, 1, 39, 0);
   m[1].setMessage(ShortMessage.NOTE_ON, 1, toonika-24, 60);
   m[2].setMessage(ShortMessage.NOTE_OFF, 1, toonika-24, 60);
```

```
  t.add(new MidiEvent(m[0], 0));
  t.add(new MidiEvent(m[1], 0));
  t.add(new MidiEvent(m[2], 12));
 }catch(Exception e){e.printStackTrace();}
}

static int pooltoonid(int aste, int muutus){
   int loppaste=aste+muutus;
   int okt=loppaste/7;
   if(loppaste<0)okt--;
   loppaste=loppaste-7*okt;
   int vahe=noodivahed[loppaste]-noodivahed[aste];
   return vahe+12*okt;
}


void looTaustaviiul(Track t, int toonika, int aste){
 try{
  int nootidearv=6;
  int[] samm=new int[nootidearv];
  int i=0;
  samm[i++]=0;
  while(i<nootidearv){
    if(Math.random()<0.5){
      samm[i]=samm[i-1]-1;
    } else {
      samm[i]=samm[i-1]+1;
    }
    i++;
  }
  ShortMessage m=new ShortMessage();
  m.setMessage(ShortMessage.PROGRAM_CHANGE, 2, 41, 0);
  t.add(new MidiEvent(m, 0));
  for(i=0; i<nootidearv; i++){
    m=new ShortMessage();
    m.setMessage(ShortMessage.NOTE_ON, 2, toonika+noodivahed[aste]+pooltoonid(aste, samm[i]), 60);
    t.add(new MidiEvent(m, i*2));
    m=new ShortMessage();
    m.setMessage(ShortMessage.NOTE_OFF, 2, toonika+noodivahed[aste]+pooltoonid(aste, samm[i]), 60);
    t.add(new MidiEvent(m, (i+1)*2));
  }
 }catch(Exception e){e.printStackTrace();}
}


Sequence pohikolmkolad(int toonika) throws InvalidMidiDataException{
  Sequence s1=new Sequence(Sequence.PPQ, 4);
  looKolmkola(s1.createTrack(), toonika);
  looKolmkola(s1.createTrack(), toonika+5); //kvart
  looKolmkola(s1.createTrack(), toonika+7); //kvint
  looBass(s1.createTrack(), toonika);
  looBass(s1.createTrack(), toonika-7); //IV ehk kvint alla
  looBass(s1.createTrack(), toonika-5);
  return s1;
}

Sequence looSekvents() throws InvalidMidiDataException{
  return pohikolmkolad(helikorgused[helistik.getSelectedIndex()]);
}

void paneSoolo(){
  for(int i=0; i<raadionupud.length; i++){
    sequencer.setTrackMute(i, !raadionupud[i].isSelected() || !akord.isSelected());
    sequencer.setTrackMute(3+i, !raadionupud[i].isSelected() || !bass.isSelected());
  }
  sequencer.setTrackMute(6, !taust.isSelected());
}

void mangi(){
  sequencer.start();
  paneSoolo();
  sequencer.setTempoInBPM(tempo.getValue());
}

void alusta(){
 try{
 if(!jooksevHelistik.equals(helistik.getSelectedItem())){
  sequence=looSekvents();
  jooksevHelistik=helistik.getSelectedItem().toString();
 }
  if(muutuvRada!=null)sequence.deleteTrack(muutuvRada);
  muutuvRada=sequence.createTrack();
  int pohitoon=helikorgused[helistik.getSelectedIndex()];
  int aste=1;
  if (duurid[duurinr]==1)aste=4;
  if (duurid[duurinr]==2)aste=5;
```

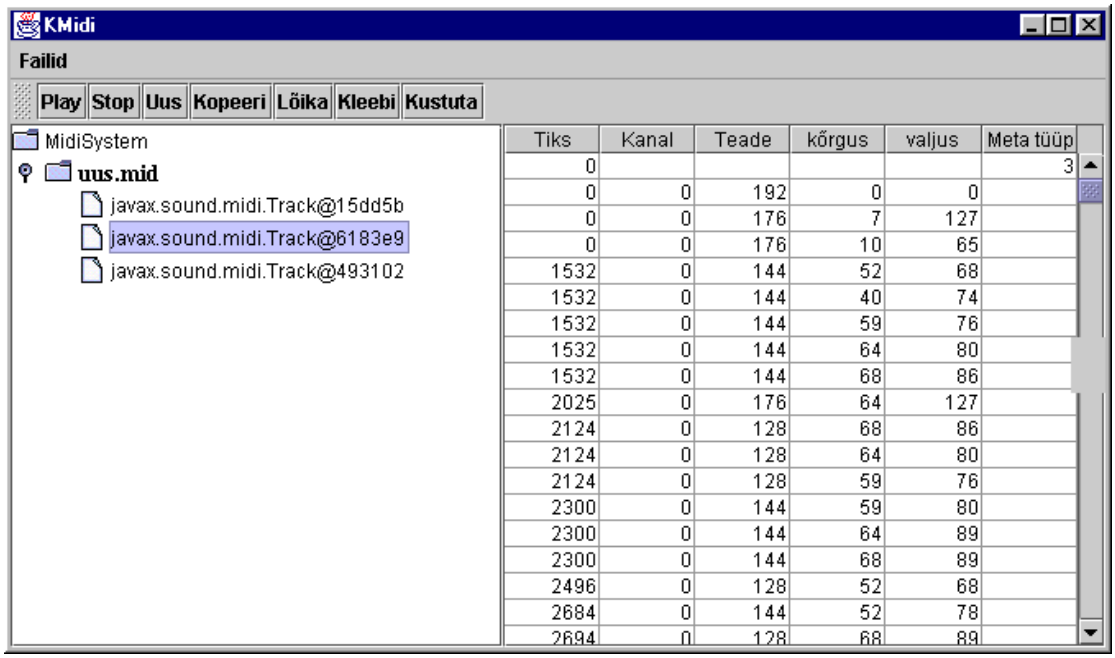
```
   looTaustaviiul(muutuvRada, pohitoon, aste);
   sequencer.setSequence(sequence);
   raadionupud[duurid[duurinr]].setSelected(true);
   duurinr=(duurinr+1)%duurid.length;
   mangi();
  }catch(Exception e){e.printStackTrace(); System.out.println(e);}
 }

 public void init(){
      Panel nupupaneel=new Panel(new GridLayout(1, 4));
      helistik=new JComboBox(helistikud);
      helistik.setSelectedIndex(3);
      nupupaneel.add(helistik);
      ButtonGroup nupugrupp=new ButtonGroup();
      for(int i=0; i<raadionupud.length; i++){
        raadionupud[i]=new JRadioButton(raadionupustring[i]);
        nupugrupp.add(raadionupud[i]);
        nupupaneel.add(raadionupud[i]);
      }
      raadionupud[0].setSelected(true);
      ruut.setSelected(true);
      Panel mangupaneel=new Panel(new GridLayout(1, 2));
      mangupaneel.add(nupp);
      mangupaneel.add(ruut);
      JPanel tempopaneel=new JPanel(new BorderLayout());
      tempopaneel.add(new JLabel("Tempo"), BorderLayout.WEST);
      tempopaneel.add(tempo, BorderLayout.CENTER);
      JPanel valikupaneel=new JPanel(new GridLayout(1, 3));
      valikupaneel.add(bass);
      valikupaneel.add(akord);
      valikupaneel.add(taust);
      akord.setSelected(true);
      JPanel alumine=new JPanel(new GridLayout(3, 1));
      alumine.add(tempopaneel);
      alumine.add(mangupaneel);
      alumine.add(valikupaneel);
      getContentPane().add(alumine, BorderLayout.SOUTH);
      getContentPane().add(nupupaneel, BorderLayout.NORTH);
      nupp.addActionListener(this);
 }


  public void actionPerformed(ActionEvent e){
   try{
         sequencer=MidiSystem.getSequencer();
         sequencer.open();
         sequencer.addMetaEventListener(this);
         jooksevHelistik="";
         alusta();
   }catch(Exception ex){
     ex.printStackTrace();
     System.out.println(ex);
   }
  }

  public void meta(MetaMessage m){
    if(m.getType()==47 && ruut.isSelected()){
      alusta();
    } else {
      sequencer.close();
    }
  }
  public static void main(String argumendid[]){
    JFrame f=new JFrame("Saade");
    Noodirakend9a ap=new Noodirakend9a();
    ap.init();
    f.getContentPane().add(ap);
    f.pack();
    f.setVisible(true);
  }


}
```

MIDI redaktor

Järgnevalt näide, kus ühendatud Java MIDI-vahendid failide lugemiseks, mängimiseks, muutmiseks ja salvestamiseks ning Swingi võimalused kasutajaliidese loomisel. Näite on koostanud TPÜ

üliõpilane Kaur Männiko graafika ja muusika programmeerimise kursuse kodutööna. Näide võimaldab avada ja luua MIDI sekventse. Vaadata radu ning neid kopeerida nii sekventsi sees kui sekventside vahel. Raja sees saab vaadata ja muuta kõiki teateid. Olles näite korralkult läbi mõelnud, peaks olema API spetsifikatsiooni järgi võimalik luua enamikke MIDI-rakendusi, mille jaoks ideid ja vajadusi on.

Graafilisest liidesest võtavad suurema osa enese alla Swingi puu ning tabel. Esimene avatud failide/sekventside ning nende sees olevate radade loeteluks. Tabelis on näha märgistatud rajal asuvad teated toimumise järjekorras koos oma tähtsamate andmetega.



Tiks	Kanal	Teade	kõrgus	valjus	Meta tüüp
0					3
0	0	192	0	0	
0	0	176	7	127	
0	0	176	10	65	
1532	0	144	52	68	
1532	0	144	40	74	
1532	0	144	59	76	
1532	0	144	64	80	
1532	0	144	68	86	
2025	0	176	64	127	
2124	0	128	68	86	
2124	0	128	64	80	
2124	0	128	59	76	
2300	0	144	59	80	
2300	0	144	64	89	
2300	0	144	68	89	
2496	0	128	52	68	
2684	0	144	52	78	
2694	0	128	68	89	

Järgnevalt koodis leiduvate tähelepanu nõudvate lõikude kirjeldus järjemööda.

Märgitud piirkonna mahamängimiseks on loodud eraldi alamprogramm. Puu käest küsitakse kasutaja märgitud komponent ning asutakse käituma vastavalt sellele, mida kasutaja märkinud oli. Mängimiseks sobivad sekvents ning rada. Esimesel puhul paikneb puu viimase lehe ehk node sees NodeObjectHolder, milles omakorda sekvents. Sekventsi saab omaette tervikuna mängima panna.

```
    public void play() {
        try {
            Object src = tree.getLastSelectedPathComponent();
...
            DefaultMutableTreeNode node = (DefaultMutableTreeNode) src;
            if ( node.getUserObject() instanceof NodeObjectHolder) {
                sekvents = (Sequence)
                  ((NodeObjectHolder)node.getUserObject()).userObject;

            } else if( node.getUserObject() instanceof Track) {
```

Raja mängitamiseks tuleb luua uus sekvents selle sekventsi parameetritega, milles mängitav rada on ning siis rajas olevad teated uue sekventsi loodavale rajale üle kanda.

```
    Track algrada = (Track)node.getUserObject();
    sekvents = (Sequence) ((NodeObjectHolder)
       ((DefaultMutableTreeNode)node.getParent()).getUserObject() ).userObject;
    sekvents = new Sequence(sekvents.getDivisionType(), sekvents.getResolution());
    Track uusrada = sekvents.createTrack();

    for (int nr = 0; nr < algrada.size(); nr++)
      uusrada.add(algrada.get(nr));
```

Sekventsi käivitamiseks piisab kahest käsust.

```
sekventser.setSequence(sekvents);
sekventser.start();
```

Klassil küljes olev TreeSelectionListener näitab, et KMidi eksemplar peab oskama ümber käia puu küljest tulevate teadetega. Kui puus valitakse uus koht, saabub selle kohta teade valueChanged käsu kaudu. Edasi kontrollitakse, et juhul kui valituks osutus rada ehk Track-tüüpi leht, siis luuakse uus TableModel ning selle kaudu palutakse paremal pool asuval tabelil näidata just valitud raja sees olevaid teateid.

Nii veerunimede kui raja jaoks loodud muutujad on piiritlejaga final, et sisemise klassi loomisel võiks sealne kood kindel olla, et muutuja külge seotud objekt ikka sinna püsima jääb ning keegi seda ära vahetada ei saa. Abstraktsest tabelimudelist tabeli jaoks tarviliku objekti kokkupanekul tuleb üle katta peotäis käsklusi, mis annavad teavet tabeli sisalduse kohta või lubavad ka seda muuta. Niisuguse mudeli kaudu ei pruugi sugugi alati kõik näidatavad väärtused kahemõõtmelises massiivis asuma, vaid võidakse võtta kusagilt mujalt või sootuks käigu peal valmis arvutada. Siin näites hoitakse andmeid raja sees ning tabel on vaid sealsete väärtuste peegliks.

Tabeli mudel on loodud KMidi sees sisemise anonüümse klassina. Paistab new AbstractTableModel(){} ning nende loogeliste sulgude vahele on paigutatud kogu ülekaetavate käskude loetelu koos sisuga. Nagu käskude nimedestki näha getColumnName annab veeru järjekorranumbrile vastava pealkirja. getColumnCount ja getRowCount teatavad, palju peaks sellele mudelile vastavas tabelise ridasid ja veerge olema. Põhjalikumad käsklused on getValueAt ning setValueAt.
Nende abil saab küsida või määrata tabeli konkreetses lahtris asuvat väärtust.

Sõltumata küsitavast veerust küsitakse rajast kõigepealt reanumbrile vastav sündmus ning asutakse selle sisu uurima. Tiksu väärtus leidub sõltumata sündmuse tüübist ning see küsitakse eraldi muutujasse. Edasi toimetatakse vastavalt sündmuse tüübile. Üldjuhul muusikalise teate ehk ShortMessage puhul leitakse kanali ja käskluse koodid ning mõlemad andmebaidid. MetaMessage puhul antakse vaid tiks või tüübi kood (nt. raja lõpp 47).

Andmete seadmisel setValueAt puhul antakse ette uus väärtus ning rida ja veerg, kuhu soovitakse see väärtus paigutada. Rea järgi otsitakse üles vastav MIDI tiks, veeru järgi parameeter selle sees. Luuakse uus teade ning asendatakse muudetav väärtus, muud kopeeritakse. Edasi pannakse uus teade vana asemele.

getColumnClass ning isCellEditable toimivad nii nagu nimigi määrab. Esimene teatab, millise Java klassile vastavate andmetega tegemist on. Teine vastab, kas vastava veeru andmeid tabelis muuta lubatakse.

Edasi järgnevad mõned käsklused radade haldamiseks puus. copyTrack paigutab märgistatud raja osuti ajutisse muutujasse. getCurrentSequence leiab märgistatud kohale vastava sekventsi. Kui märgistatud on sekvents, väljastatakse osuti sellele enesele. Kui aga rada, siis leitakse sekvents, millesse vastav rada kuulub. newSequence loob uue sekventsi, leiab selle nime ning palub sekventsi puusse paigutada. pasteTrack loob märgistatud sekventsi sisse uue raja ning lisab sellesse eelnevalt meelde jäetud rajal paiknevad teated. Uuele rajale kantakse üle vaid teadete osutid, teadetest endist koopiaid ei tehta. Alles siis, kui kasutaja soovib tabeli kaudu muuta teates paiknevaid parameetreid, luuakse endise asemele uus teate eksemplar.

Sekventsi puuse lisamise eest hoolitseb addSequenceToTree. Sekventsi (faili) nimi tehakse rasvaseks HTML-i käskude abil, mida on Swingi komponentide juures lubatud kujunduseks kasutada. Kõikide sekventsis asuvate radade tarbeks luuakse puus esitamise jaoks tarvilikud DefaultMutableTreeNode tüüpi komponendid.

Järgnev kood on juba küllalt tavapärane pea iga Swingi rakenduses puhul. actionPerformed sündmuste haldamiseks. Faili lugemisel paigutatakse tekstifailis olevad andmed sekventsi,

salvestamisel aga talletatakse sekventsi andmed MIDI-faili.

Menüü juures on tarvis vähemasti kolm osa. JMenuBar tähistab kogu menüüriba, JMenu ühte menüüpaneeli ning sinna külge JMenuItem, mille kaudu juba inimene käsklusi anda saab. Ning edasi juba head katsetamist ning jõudu näite põhjal omale tarviliku rakenduse koostamisel.

```
import javax.swing.*;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.io.IOException;
import java.io.File;
import java.util.*;


import java.awt.*;
import java.awt.event.*;
import java.io.*;


import javax.sound.midi.*;
import javax.swing.tree.*;
import javax.swing.table.*;
import javax.swing.event.*; //TreeSelectionListener, -event


/**
* Midi-redaktor. Koostanud TPÜ Informaatika üliõpilane Kaur Männiko
* graafika ja muusika programmeerimise kursuse kodutööna.
*/
public class KMidi extends JPanel implements ActionListener, TreeSelectionListener {
    Sequence sekvents;
    Sequencer sekventser;
    JButton btnPlay, btnStop, btnNew, btnCopy, btnCut, btnPaste, btnDelete;
    JScrollPane treeView;
    final JTree tree;
    final JTable table;
    File file;
    DefaultMutableTreeNode rootNode;
    DefaultTreeModel treeModel;


    public void play() {
        try {
            Object src = tree.getLastSelectedPathComponent();
            if (!(src instanceof DefaultMutableTreeNode))
                return;
            DefaultMutableTreeNode node = (DefaultMutableTreeNode) src;
            if (node.getUserObject() == null)
                return;

            if ( node.getUserObject() instanceof NodeObjectHolder) {
                sekvents = (Sequence) ((NodeObjectHolder)node.getUserObject()).userObject;

            } else if( node.getUserObject() instanceof Track) {
                Track algrada = (Track)node.getUserObject();
                sekvents = (Sequence) ((NodeObjectHolder) ((DefaultMutableTreeNode)node.getParent
()).getUserObject() ).userObject;
                sekvents = new Sequence(sekvents.getDivisionType(), sekvents.getResolution());
                Track uusrada = sekvents.createTrack();

                for (int nr = 0; nr < algrada.size(); nr++)
                    uusrada.add(algrada.get(nr));


            } else
                return; //root

            sekventser.setSequence(sekvents);
            sekventser.start();

        } catch (InvalidMidiDataException e) {e.printStackTrace();};
    }

    public void stop() {
            sekventser.stop();
    }

    public KMidi() {
        super(new BorderLayout());


        JToolBar toolbar = new JToolBar();


        btnPlay = new JButton("Play");
        btnPlay.addActionListener(this);
```

```
        toolbar.add(btnPlay);
        btnStop = new JButton("Stop");
        btnStop.addActionListener(this);
        toolbar.add(btnStop);
        btnNew = new JButton("Uus");
        btnNew.addActionListener(this);
        toolbar.add(btnNew);
        btnCopy = new JButton("Kopeeri");
        btnCopy.addActionListener(this);
        toolbar.add(btnCopy);
        btnCut = new JButton("Lõika");
        btnCut.addActionListener(this);
        toolbar.add(btnCut);
        btnPaste = new JButton("Kleebi");
        btnPaste.addActionListener(this);
        toolbar.add(btnPaste);
        btnDelete = new JButton("Kustuta");
        btnDelete.addActionListener(this);
        toolbar.add(btnDelete);


        toolbar.setOrientation(JToolBar.HORIZONTAL);
        add(toolbar, BorderLayout.NORTH);


        rootNode = new DefaultMutableTreeNode("MidiSystem");
        treeModel = new DefaultTreeModel(rootNode);


        tree = new JTree(treeModel);
        tree.addTreeSelectionListener(this);

        treeView = new JScrollPane(tree);
        treeView.setPreferredSize(new Dimension(300,200));


        add(treeView, BorderLayout.WEST);

        table = new JTable();
        JScrollPane scrollpane = new JScrollPane(table);

        add(scrollpane, BorderLayout.CENTER);


        try {
            sekventser = MidiSystem.getSequencer();
            if (!sekventser.isOpen()) {
                System.out.print("avatakse sekventser...");
                sekventser.open();
                System.out.println(" ok");
            }
        } catch (MidiUnavailableException e) {e.printStackTrace();};

    }


    public void valueChanged(TreeSelectionEvent e) {
        Object src = e.getPath().getLastPathComponent();
        if (!(src instanceof DefaultMutableTreeNode))
           return;
        DefaultMutableTreeNode node = (DefaultMutableTreeNode) src;
        if (node.getUserObject() == null)
            return;
        if(!( node.getUserObject() instanceof Track))
            return;

        final Track track = (Track) node.getUserObject();
        final String[] columnNames = {"Tiks",
                                      "Kanal",
                                      "Teade",
                                      "kõrgus",
                                      "valjus",
                                      "Meta tüüp"};

        TableModel dataModel = new AbstractTableModel() {
            public String getColumnName(int column) { return columnNames[column];}
            public int getColumnCount() { return columnNames.length; }
            public int getRowCount() { return track.size();}

            public Object getValueAt(int row, int col) {
                MidiEvent me = track.get(row);
                Long tick = new Long(me.getTick());
                MidiMessage msg = me.getMessage();

                if (msg instanceof ShortMessage){
                    ShortMessage sm = (ShortMessage)msg;
                    switch (col) {
                        case 0: return tick;
```

```
                case 1: return new Integer(sm.getChannel());
                case 2: return new Integer(sm.getCommand());
                case 3: return new Integer(sm.getData1());
                case 4: return new Integer(sm.getData2());
                default: return "";
            }
        }
        else if (msg instanceof MetaMessage){
            MetaMessage mm = (MetaMessage)msg;
            switch (col) {
                case 0: return tick;
                case 5: return new Integer(mm.getType());
                default: return "";
            }
        }
        else if (col == 0)
                return tick;
            else
                return "";
    }

    public Class getColumnClass(int c) {
        //return getValueAt(0, c).getClass();
        //return Class.forName("Integer");
        return (new Long(0)).getClass();
    }

    public void setValueAt(Object aValue, int row, int col) {
        System.out.println("set value "+row+","+ col);


        MidiEvent me = track.get(row);
        Long tick = new Long(me.getTick());
        MidiMessage msg = me.getMessage();

        if (col == 0) {
            me.setTick(((Long)aValue).longValue());
            return;
        }
        int val = ((Long)aValue).intValue();

        if (msg instanceof ShortMessage){
            ShortMessage sm = (ShortMessage)msg;
            int command = sm.getCommand();
            int channel = sm.getChannel();
            int data1 = sm.getData1();
            int data2 = sm.getData2();

            int status = sm.getStatus();

            System.out.println("enne muutmist: "+command+","+channel+","+data1+","+ data2);
            switch (col) {
                //case 0: me.setTick(val);
                case 1: channel = val; break;
                case 2: command = val; break;
                case 3: data1 = val; break;
                case 4: data2 = val; break;
            }
            System.out.println("muudetakse: "+command+","+channel+","+data1+","+ data2);

            try {
                ShortMessage sm2 = new ShortMessage();
                sm2.setMessage(command, channel, data1, data2);
                MidiEvent me2 = new MidiEvent(sm2, me.getTick());
                track.remove(me);
                track.add(me2);

            } catch (InvalidMidiDataException ex) {
                ex.printStackTrace();
            }


        }
    }

    public boolean isCellEditable(int row, int col) {
        if (col == 5)
            return false;
        else
            return true;
    }

};

table.setModel(dataModel);
```

```
    }


    Track tempTrack = null;

    public void copyTrack() {
        if (tree.getLastSelectedPathComponent() == null) return;
        DefaultMutableTreeNode node = (DefaultMutableTreeNode) tree.getLastSelectedPathComponent();
        if( node.getUserObject() instanceof Track) {
            tempTrack = (Track)node.getUserObject();

        } else
            return; //root
    }

    private Sequence getCurrentSequence() {
        Sequence seq = null;
        if (tree.getLastSelectedPathComponent() == null) return null;
        DefaultMutableTreeNode node = (DefaultMutableTreeNode) tree.getLastSelectedPathComponent();
        if ( node.getUserObject() instanceof NodeObjectHolder) {
            seq = (Sequence) ((NodeObjectHolder)node.getUserObject()).userObject;

        } else if( node.getUserObject() instanceof Track) {
            Track algrada = (Track)node.getUserObject();
            seq = (Sequence) ((NodeObjectHolder) ((DefaultMutableTreeNode)node.getParent()).
getUserObject() ).userObject;
        }
        return seq;
    }


    private int jnr = 0;

    public void newSequence() {
        sekvents = getCurrentSequence();
        if (sekvents == null) return;
        try{
            sekvents = new Sequence(sekvents.getDivisionType(), sekvents.getResolution());
            //addSeqenceToTree(sekvents, "uus-"+(new Integer(jnr++)).toString()+".mid");
            addSeqenceToTree(sekvents, "uus-"+(jnr++)+".mid");
        }catch (InvalidMidiDataException e) {e.printStackTrace();}
    }

    public void cutTrack() {
        copyTrack();
        deleteTrack();
    }

    public void pasteTrack() {
        if (tree.getLastSelectedPathComponent() == null) return;
        DefaultMutableTreeNode node = (DefaultMutableTreeNode) tree.getLastSelectedPathComponent();
        if ( node.getUserObject() instanceof Track)
            node = (DefaultMutableTreeNode)node.getParent();
        if (!( node.getUserObject() instanceof NodeObjectHolder))
            return;


        sekvents = (Sequence) ((NodeObjectHolder) node.getUserObject() ).userObject;
        Track uusrada = sekvents.createTrack();

        for (int nr = 0; nr < tempTrack.size(); nr++)
            uusrada.add(tempTrack.get(nr));  //NB! ei tee uusi eksemplare


        DefaultMutableTreeNode trackNode = new DefaultMutableTreeNode(uusrada);
        node.add(trackNode);
        treeModel.insertNodeInto(trackNode, node, 0);
    }

    public void deleteTrack() {
        if (tree.getLastSelectedPathComponent() == null) return;
        DefaultMutableTreeNode node = (DefaultMutableTreeNode) tree.getLastSelectedPathComponent();
        if( node.getUserObject() instanceof Track) {
               Sequence sekvents = (Sequence) ((NodeObjectHolder) ((DefaultMutableTreeNode)
node.getParent()).getUserObject() ).userObject;
               Track track = (Track)node.getUserObject();
               sekvents.deleteTrack(track);
               treeModel.removeNodeFromParent(node);
        } else
            return; //root
    }

    public void addSeqenceToTree(Sequence seq, String name) {
        DefaultMutableTreeNode sequenceNode = null;
        DefaultMutableTreeNode trackNode = null;
        NodeObjectHolder noh = new NodeObjectHolder("<html><b>" + name + "</b></html>", seq);
        sequenceNode = new DefaultMutableTreeNode(noh);
```

```
    Track[] rajad = seq.getTracks();
    System.out.println(rajad.length+" rada");
    for (int nr = 0; nr < rajad.length; nr++){
        trackNode = new DefaultMutableTreeNode(rajad[nr]);
        sequenceNode.add(trackNode);
    }


    treeModel.insertNodeInto(sequenceNode, rootNode, rootNode.getChildCount());
}

public void actionPerformed(ActionEvent e) {
    if (e.getSource() == btnPlay) {
        play();
    } else if (e.getSource() == btnStop) {
        stop();
    } else if (e.getSource() == btnCopy) {
        copyTrack();
    } else if (e.getSource() == btnNew) {
        newSequence();
    } else if (e.getSource() == btnCut) {
        cutTrack();
    } else if (e.getSource() == btnPaste) {
        pasteTrack();
    } else if (e.getSource() == btnDelete) {
        deleteTrack();
    } else if (e.getSource() == mnLoad) {
        loadFile();
    } else if (e.getSource() == mnSave) {
        saveFile();
    }

}

JFileChooser fc = new JFileChooser();

public void loadFile() {
    int returnVal = fc.showOpenDialog(this);

    if (returnVal == JFileChooser.APPROVE_OPTION) {
        file = fc.getSelectedFile();
        try {
            System.out.println("Opening: " + file.getCanonicalPath());


            sekvents = MidiSystem.getSequence(file);
            addSeqenceToTree(sekvents, file.getName());
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    } else {
        System.out.println("Open command cancelled by user.");
    }
}

public void saveFile() {
    sekvents = getCurrentSequence(); //millist faili salvestada
    if (sekvents == null) return;

    int returnVal = fc.showSaveDialog(this);
    if (returnVal == JFileChooser.APPROVE_OPTION) {
        file = fc.getSelectedFile();



        try {
            MidiSystem.write(sekvents, 1, new FileOutputStream(file.getCanonicalPath()));

            System.out.println("Saved: " + file.getCanonicalPath());
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    } else {
        System.out.println("Save command cancelled by user.");
    }
}

JMenuItem mnUus, mnLoad, mnSave;

public JMenuBar createMenuBar() {
```

```
        JMenuBar menuBar = new JMenuBar();
        JMenu menu = null;
        JMenuItem menuItem = null;


        menu = new JMenu("Failid");


        menuItem = new JMenuItem("Lae..");
        menuItem.addActionListener(this);
        menu.add(menuItem);
        mnLoad = menuItem;


        menuItem = new JMenuItem("Salvesta..");
        menuItem.addActionListener(this);
        menu.add(menuItem);
        mnSave = menuItem;


        menuBar.add(menu);


        return menuBar;
    }


    public static void main(String[] args) {
        // JFrame.setDefaultLookAndFeelDecorated(true);
        // Käsk kasutatav alates JDK versioonist 1.4


        JFrame frame = new JFrame("KMidi");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


        KMidi kodContentPane = new KMidi();
        kodContentPane.setOpaque(true);
        frame.setJMenuBar(kodContentPane.createMenuBar());
        frame.setContentPane(kodContentPane);


        frame.pack();
        frame.setSize(600, 400);
        frame.setVisible(true);
    }

}



/**
* Vahend, mille abil hoida objekti koos tema juurde kuuluva sildiga
* Swingi puu tarvis esitataval kujul. Kasutatakse näiteks sekventsi
* hoidmiseks.
*/
class NodeObjectHolder {
    public String label = "no name";
    public Object userObject = null;

    public NodeObjectHolder() {
    }

    public NodeObjectHolder(String label, Object userObject) {
        this.label = label;
        this.userObject = userObject;
    }

    public String toString() {
        return label;
    }
}
```

Ülesandeid

Mandoliin

- Joonista ekraanile mandoliin.
- Kasutaja saab määrata joonistatavate krihvide arvu ning kaela laiust.
- Vajutades keele ja krihvi ristumiskohale, kõlab sellele vastav heli.

Kitarri mudel

- Joonista 6-keelse kitarri kaela mudel
- Mängi kitarriakord (E, H, G, D, A, E) (64, 59, 55, 50, 45, 40)
- Peenemal keelel saab määrata, milline krihv on alla vajutatud. Hääl kõlab vastavalt vajutatule
- Krihve saab valida ka teiste keelte puhul ning kuulata tulemust
- Lisaks võib valida täisakorde.

Akordion

- Joonista klaviatuuri ja 36 bassiga akordion.
- Basside ridade ja veergude arvu saab kasutaja määrata. Samuti hiire abil lõõtsa lahti ja kokku vedada.
- Lisaks eelmisele kõlab bassinuppudele vajutades neile vastav heli.

Vilepill

- Joonista vilepill. Pillil on avatud tekstiväljas määratud hulk auke.
- Lisaks eelmisele teeb pillile vajutamisel viimane häält.
- Augule vajutamisel on augud lõpust kuni sinnani avatud. Kõlab sellisele sõrmede paigutusele vastav hääl.

MIDI fail

- Mängi MIDI fail
- Väljasta radade arv
- Salvesta fail tooni jagu kõrgemalt.
- Kopeeri faili muusika iseendale sappa
- Lisa rada, kus viisi mängitakse nihkega (kaanon)