RandomSequencePlayer.java 5(5)

```
private class Changer implements LineListener {

  public void update(LineEvent e) {
    if (e.getType().equals(LineEvent.Type.START))
      System.out.println("New clip started.");
    if (e.getType().equals(LineEvent.Type.STOP))
      startRandomClip();
  }
}
```

Lähdedatalinjan käyttö

- varataan puskuriksi `byte`-taulukko
- puskurin koko
  - lyhyt: nopeampi vaste, katkosten riski
  - pitkä: hitaampi vaste, sietää katkoksia
- `write`-metodin kutsu aloittaa toiston (mm. lähettää kuuntelijalle aloitusviestin)
- `drain`-metodi odottaa, kunnes kaikki kirjoitettu data on toistettu
- `flush`-metodi poistaa kirjoitetun datan

SynthPlayer.java 1(5)

```
import java.io.*;
import java.util.*;
import javax.sound.sampled.*;

public class SynthPlayer {
  public static void main(String[] args) {
    float sampleFreq = 44100.0f;
    int bitsInQuantization = 8;
    int channels = 1;
    boolean signed = true;
    boolean bigEndian = true;
    AudioFormat format = new AudioFormat(
                sampleFreq, bitsInQuantization,
                channels, signed, bigEndian);
```

SynthPlayer.java 2(5)

```
DataLine.Info info = new DataLine.Info(
                  SourceDataLine.class, format);

if (AudioSystem.isLineSupported(info)) {
  try {
    SourceDataLine line =
       (SourceDataLine)AudioSystem.getLine(info);

    int bufferSize = 6 * (int)sampleFrequency;
    byte[] buffer = new byte[bufferSize];

    int twoSecMarker = 2 * (int)sampleFrequency;
    int fourSecMarker = 4 * (int)sampleFrequency;
```

SynthPlayer.java 3(5)

```
Random random = new Random();
for (int i = 0; i < twoSecMarker; i++)
  buffer[i] = (byte)random.nextInt();

int waveLength = (int)sampleFrequency / 440;
for (int i = twoSecMarker;
     i <= (fourSecMarker - waveLength);
     i += waveLength) {
  for (int j = i; j < i + waveLength / 2; j++)
    buffer[j] = Byte.MAX_VALUE;
  for (int j = i + waveLength / 2;
       j < i + waveLength; j++)
    buffer[j] = Byte.MIN_VALUE;
} // for
```

SynthPlayer.java 4(5)

```
for (int i = fourSecMarker;
     i <= (bufferSize - waveLength);
     i += waveLength) {
  for (int j = i; j < i + waveLength; j++)
    buffer[j] = (byte)(127.0 * Math.sin(
         (double)j / waveLength * 2 * Math.PI));
} // for
```

SynthPlayer.java 5(5)

```
// Avataan linja ja aloitetaan toisto.
line.open(format);
line.start();
// Kirjoitetaan puskuri linjalle.
line.write(buffer, 0, bufferSize);
// Odotetaan linjan tyhjentymistä
// ennen kuin lopetetaan.
line.drain();
line.stop();
line.close();
```

Kohdedatalinjan käyttö

- varataan puskuriksi `byte`-taulukko
- puskurin koon vaikutus kuten lähdedatalinjassa
- `read`-metodin kutsu lukee puskuriin dataa ja palauttaa sen määrän tavuina
- `drain`-metodi odottaa, kunnes mikserissä oleva data tulee luetuksi
- `flush`-metodi poistaa lukemista odottavan datan (muuten se jää odottamaan mikseriin)

Karaoke.java 1(2)

```
DataLine.Info infoTarget = new
     DataLine.Info(TargetDataLine.class, format);
DataLine.Info infoSource = new
     DataLine.Info(SourceDataLine.class, format);
if (AudioSystem.isLineSupported(infoTarget)
  & AudioSystem.isLineSupported(infoSource) {
  try {
    TargetDataLine lineTarget = (TargetDataLine)
                AudioSystem.getLine(infoTarget);
    SourceDataLine lineSource = (SourceDataLine)
                AudioSystem.getLine(infoSource);
int bufferSize =  (int)(bufferLength *
  format.getFrameSize() * format.getFrameRate());
byte[] buffer = new byte[bufferSize];
```

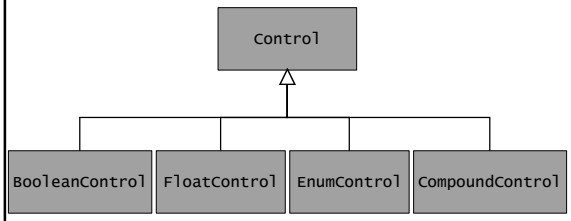
Karaoke.java 2(2)

```
lineTarget.open(format, bufferSize);
lineSource.open(format);
lineTarget.start();
lineSource.start();

while (true) {
  int dataSize =
          lineTarget.read(buffer, 0, bufferSize);

  // Tässä kohtaa puskurissa olevalle äänelle
  // voisi tehdä jotain jäynää...

  lineSource.write(buffer, 0, dataSize);
}
```

Linjan säätimet (*controls*)

- `Control`-luokasta periytyy neljä säädintyyppiä:
  - `BooleanControl`: katkaisin
    - esim. mykistys (*mute*)
  - `FloatControl`: säätökytkin
    - esim. vahvistus, panorointi
  - `EnumControl`: valintakytkin
    - esim. kaiunnan esivalinnat
  - `CompoundControl`: säädinkokoelma
    - esim. taajuuskorjain voi olla kokoelma `FloatControl`-tyyppisiä säätökytkimiä

Säädinten luokkahierarkia



Linjan säädinten haku

- `getControls`-metodi palauttaa taulukon linjan tarjoamista säätimistä
- `isControlSupported`-metodi palauttaa onko halutun tyyppistä säädintä tarjolla
- `getControl`-metodi palauttaa pyydetyn tyyppisen säätimen

Säädintyyppejä

- `BooleanControl.Type.MUTE`
- `EnumControl.Type.REVERB`
- `FloatControl.Type.MASTER_GAIN`
- `FloatControl.Type.PAN`
- `FloatControl.Type.SAMPLE_RATE`

`FloatControl`-luokan metodeja

- `float getValue()`
- `void setValue(float newValue)`
- `float getMaximum()`
- `float getMinimum()`
- `float getPrecision()`
- `String getMaxLabel()`
- `String getMidLabel()`
- `String getMinLabel()`
- `String getUnits()`

`ControlPlayer.java` 1(3)

```
boolean gain = false, pan = false,
        rate = false, mute = false;
if (args.length >= 1) {
  gain = args[1].equals("gain");
  pan = args[1].equals("pan");
  rate = args[1].equals("rate");
  mute = args[1].equals("mute");
}
float parameter = 0.0f;
if (args.length == 3) {
  try {
    parameter = Float.parseFloat(args[2]);
  } catch (NumberFormatException e) { ... }
}
```

`ControlPlayer.java` 2(3)

```
clip.open(source);
clip.start();

if (gain && clip.isControlSupported(
       FloatControl.Type.MASTER_GAIN)) {
  FloatControl gainCtrl =
     (FloatControl)clip.getControl(
         FloatControl.Type.MASTER_GAIN);
  gainCtrl.setValue(parameter);
}
```

`ControlPlayer.java` 3(3)

```
if (mute && clip.isControlSupported(
                   BooleanControl.Type.MUTE)) {
  BooleanControl muteCtrl =
       (BooleanControl)clip.getControl(
                    BooleanControl.Type.MUTE);
  muteCtrl.setValue(true);
} // if

Control[] ctrl = clip.getControls();
for (int i = 0; i < ctrl.length; i++)
  System.out.println(ctrl[i]);
```

Ääniominaisuuksien käyttöoikeudet

- määritelty `AudioPermission`-luokassa:
  - toisto
  - äänitys
- appletti: saa toistaa muttei äänittää
- sovellus: saa toistaa ja äänittää
- ohjelmien oikeuksia voidaan muuttaa Policy Tool -ohjelmalla