AudioSystemTest.java

```
import javax.sound.sampled.*;

public class AudioSystemTest
  {
  public static void main(String[] args)
    {
    // Haetaan järjestelmään asennetut mikserit.
    Mixer.Info[] mi = AudioSystem.getMixerInfo();

    // Käydään läpi löydetyt mikserit.
    for (int i = 0; i < mi.length; i++)
      {
      System.out.println(mi[i]);
      Mixer m = AudioSystem.getMixer(mi[i]);

      // Haetaan mikserin lähdelinjat.
      Line.Info[] sli = m.getSourceLineInfo();
      for (int j = 0; j < sli.length; j++)
        System.out.println("source: " + sli[j]);

      // Haetaan mikserin kohdelinjat.
      Line.Info[] tli = m.getTargetLineInfo();
      for (int j = 0; j < tli.length; j++)
        System.out.println("target: " + tli[j]);

      System.out.println();
      }
    }
  }
```

SimplePlayer.java

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

public class SimplePlayer
  {
  public static void main(String[] args)
    {
    // Komentoriviargumenttien käsittelyä.
    if (args.length == 0) System.exit(0);
    File file = new File(args[0]);
    int loopCount = 0;
    if (args.length > 1 && args[1].equals("loop"))
      {
      if (args.length > 2)
        {
```

```
    try
      {
      // Silmukoiden lukumäärä.
      loopCount = Integer.parseInt(args[2]) - 1;
      }
    catch (NumberFormatException e)
      {
      System.err.println("Ei kokonaisluku: " + args[2]);
      System.exit(1);
      }
    }
  else
    // Ikisilmukka.
    loopCount = Clip.LOOP_CONTINUOUSLY;
  }

try
  {
  // Haetaan tiedostovirta ja sen ääniformaatti.
  AudioInputStream source =
                            AudioSystem.getAudioInputStream(file);
  AudioFormat format = source.getFormat();

  // Luodaan Clip-tyyppinen linja ääniformaatille.
  DataLine.Info info = new DataLine.Info(Clip.class, format);

  // Onko järjestelmässä pyydetyn laista linjaa?
  if (!AudioSystem.isLineSupported(info))
    {
    System.err.println("Ei löydy sopivaa linjaa.");
    System.exit(1);
    }

  try
    {
    // Pyydetään järjestelmältä linja.
    Clip clip = (Clip)AudioSystem.getLine(info);

    // Luetaan tiedosto.
    clip.open(source);

    // Soitetaanko vain kerran vai silmukoidaanko?
    if (loopCount == 0)
      clip.start();
    else
      clip.loop(loopCount);
    }
  catch (LineUnavailableException e)
    {
    System.err.println("Linjaa ei voi käyttää.");
    }
  } // try
```

```
    catch (IOException e)
      {
      System.err.println("Virhe tiedoston luvussa.");
      }
    catch (UnsupportedAudioFileException e)
      {
      System.err.println("Tuntematon tiedostoformaatti.");
      }
    } // main()
  } // class
```

RandomSequencePlayer.java

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

public class RandomSequencePlayer
  {
  private Random random = new Random();
  private Clip[] clips;

  public RandomSequencePlayer(File[] files)
    {
    try
      {
      // Haetaan tiedostovirrat ja formaatit ja varataan linjat.
      AudioInputStream[] sources =
                              new AudioInputStream[files.length];
      AudioFormat[] formats = new AudioFormat[files.length];
      DataLine.Info[] infos = new DataLine.Info[files.length];
      for (int i = 0; i < sources.length; i++)
        {
        sources[i] = AudioSystem.getAudioInputStream(files[i]);
        formats[i] = sources[i].getFormat();
        infos[i] = new DataLine.Info(Clip.class, formats[i]);
        if (!AudioSystem.isLineSupported(infos[i]))
          {
          System.err.println("Ei löydy sopivaa linjaa.");
          System.exit(1);
          } // if
        } // for

      // Luetaan tiedostot.
      try
        {
        clips = new Clip[sources.length];
        for (int i = 0; i < clips.length; i++)
          {
          clips[i] = (Clip)AudioSystem.getLine(infos[i]);
          // Lisätään linjalle tapahtumia kuunteleva olio.
          clips[i].addLineListener(new Changer());
```

```
        clips[i].open(sources[i]);
        }
      }
    catch (LineUnavailableException e)
      {
      System.err.println("Linjaa ei voi käyttää.");
      }
    } // try
  catch (IOException e)
    {
    System.err.println("Virhe tiedoston luvussa.");
    }
  catch (UnsupportedAudioFileException e)
    {
    System.err.println("Tuntematon tiedostoformaatti.");
    }
  } // constructor


public void startRandomClip()
  {
  // Arvotaan ääni, siirretään osoitin sen alkuun
  // ja soitetaan se.
  int finger = random.nextInt(clips.length);
  clips[finger].setFramePosition(0);
  clips[finger].start();
  }


public static void main(String[] args)
  {
  // Parsitaan komentorivi ja käynnistetään sovellus.
  File[] files = new File[args.length];
  for (int i = 0; i < files.length; i++)
    files[i] = new File(args[i]);
  RandomSequencePlayer rsp = new RandomSequencePlayer(files);
  rsp.startRandomClip();
  }


// Linjan tapahtumien kuuntelija.
private class Changer implements LineListener
  {
  public void update(LineEvent e)
    {
    // Jos soitto alkoi, raportoi.
    if (e.getType().equals(LineEvent.Type.START))
      System.out.println("New clip started.");
    // Jos soitto loppui, käynnistä uusi ääni.
    if (e.getType().equals(LineEvent.Type.STOP))
      startRandomClip();
    }
  }
} // class
```

SynthPlayer.java

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

public class SynthPlayer
  {
  public static void main(String[] args)
    {
    float sampleFrequency = 44100.0f;
    int bitsInQuantization = 8;
    int numberOfChannels = 1;
    boolean signedValues = true;
    boolean bigEndianValues = true;

    // Luodaan ääniformaatti.
    AudioFormat format =
      new AudioFormat(sampleFrequency, bitsInQuantization,
                  numberOfChannels, signedValues, bigEndianValues);

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

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

        // Kuuden sekunnin mittainen puskuri.
        int bufferSize = 6 * (int)sampleFrequency;
        byte[] buffer = new byte[bufferSize];

        int twoSecondMarker = 2 * (int)sampleFrequency;
        int fourSecondMarker = 4 * (int)sampleFrequency;

        // Kaksi sekuntia kohinaa.
        Random random = new Random();
        for (int i = 0; i < twoSecondMarker; i++)
          buffer[i] = (byte)random.nextInt();

        // Kaksi sekuntia 440 Hz:n kanttiaaltoa.
        int waveLength = (int)sampleFrequency / 440;

        for (int i = twoSecondMarker;
             i <= (fourSecondMarker - 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;
```

```
          }

        // Kaksi sekuntia 440 Hz:n siniaaltoa.
        for (int i = fourSecondMarker;
             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));
          }

        // 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();
        }
      catch (LineUnavailableException e)
        {
        System.err.println("Linjaa ei voi käyttää.");
        }
      }
    else System.err.println("Ei löydy sopivaa linjaa.");
    }
  }
```

Karaoke.java

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

public class Karaoke
  {
  // Puskurin pituus sekunneissa.
  private static final float bufferLength = 0.5f;

  public static void main(String[] args)
    {
    AudioFormat format = new AudioFormat(44100.0f, 16, 1,
                                         true, true);

    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
      {
      // Pyydetään linjat.
      TargetDataLine lineTarget =
          (TargetDataLine)AudioSystem.getLine(infoTarget);
      SourceDataLine lineSource =
          (SourceDataLine)AudioSystem.getLine(infoSource);

      // Varataan tilaa puskurille.
      int bufferSize =  (int)(bufferLength
            * format.getFrameSize() * format.getFrameRate());
      byte[] buffer = new byte[bufferSize];

       // Linjat auki ja käyntiin.
      lineTarget.open(format, bufferSize);
      lineSource.open(format);
      lineTarget.start();
      lineSource.start();

      // Ikisilmukka (rumaa...)
      while (true)
        {
        // Luetaan dataa puskuriin.
        int dataSize = lineTarget.read(buffer, 0, bufferSize);

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

        // Kirjoitetaan puskurissa oleva data.
        lineSource.write(buffer, 0, dataSize);
        }
      }
    catch (LineUnavailableException e)
      {
      System.err.println("Linja ei ole käytettävissä.");
      }
    }
  else System.err.println("Pyydettyä linjaa ei löydy.");
  }
}
```

ControlPlayer.java

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

public class ControlPlayer
  {
  public static void main(String[] args)
    {
    // Komentoriviargumenttien käsittelyä.
    if (args.length == 0) System.exit(0);
    File file = new File(args[0]);

    boolean gain = false;
    boolean pan = false;
    boolean rate = false;
    boolean 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)
        {
        System.err.println("Ei liukuluku: " + args[2]);
        System.exit(1);
        }
      }

    try
      {
      // Haetaan tiedostovirta ja sen ääniformaatti.
      AudioInputStream source =
                           AudioSystem.getAudioInputStream(file);
      AudioFormat format = source.getFormat();

      // Luodaan Clip-tyyppinen linja ääniformaatille.
      DataLine.Info info = new DataLine.Info(Clip.class, format);

      // Onko järjestelmässä pyydetyn laista linjaa?
      if (!AudioSystem.isLineSupported(info))
        {
        System.err.println("Ei löydy sopivaa linjaa.");
        System.exit(1);
        }
```

```
try
  {
  // Pyydetään järjestelmältä linja.
  Clip clip = (Clip)AudioSystem.getLine(info);

  // Luetaan tiedosto ja soitetaan se.
  clip.open(source);
  clip.start();

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

  // Panorointi.
  if (pan &&
      clip.isControlSupported(FloatControl.Type.PAN))
    {
    FloatControl panCtrl = (FloatControl)clip.getControl(
                            FloatControl.Type.PAN);
    panCtrl.setValue(parameter);
    }

  // Taajuuden muutos.
  if (rate &&
      clip.isControlSupported(FloatControl.Type.SAMPLE_RATE))
    {
    FloatControl rateCtrl = (FloatControl)clip.getControl(
                             FloatControl.Type.SAMPLE_RATE);
    rateCtrl.setValue(parameter);
    }

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

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

  }
catch (LineUnavailableException e)
  {
```

```
      System.err.println("Linjaa ei voi käyttää.");
      }
    } // try
  catch (IOException e)
    {
    System.err.println("Virhe tiedoston luvussa.");
    }
  catch (UnsupportedAudioFileException e)
    {
    System.err.println("Tuntematon tiedostoformaatti.");
    }
  } // main()
} // class
```