3. Javan ääniohjelmointi

1. `java.applet`
2. `javax.sound.sampled`

3.1. `java.applet`

- tarjoaa yleisen, yksinkertaisen ja laitteistoriippumattoman äänipalvelun
- käytettävissä:
  - appleteissa (JDK 1.0)
  - sovelluksissa (JDK 1.2)
- tukee ääniformaatteja AIFF, AU, WAV, MIDI, RMF

`Applet`-luokan äänimetodeja

- `void play(URL url)`
- `void play(URL url, String name)`
- `AudioClip getAudioClip(URL url)`
- `AudioClip getAudioClip(URL url, String name)`
- `static AudioClip newAudioClip(URL url)`

Muita esille tulevia `Applet`-metodeja

- `void init()`
- `void start()`
- `void stop()`
- `URL getCodeBase()`
- `void showStatus(String msg)`
- `String getParameter(String name)`

`AudioClip`-rajapinta

- metodit yksittäistoistoon, silmukointiin ja pysäyttämiseen

```
interface AudioClip {
    public void play();
    public void loop();
    public void stop();
}
```

`URL`-luokka

- sijaitsee pakkauksessa `java.net`
- `getCodeBase`-metodilla saadaan appletin perusosoite, johon lisätään äänitiedoston nimi
- osoite luodaan `URL`-luokan konstruktorilla
  `public URL(String spec) throws MalformedURLException`
- poikkeus on käsiteltävä

SimpleAudioApplet.java 1(3)

```
import java.applet.*;
import java.net.*;

public class SimpleAudioApplet
                  extends Applet {

  private AudioClip sound = null;
```

SimpleAudioApplet.java 2(3)

```
  public void init() {
    try {
      URL sf = new URL(getCodeBase()
                       + "sound.wav");
      sound = getAudioClip(sf);
    } catch (MalformedURLException e) {
      showStatus("Cannot load the "
                 + "audio file.");
    } // try
  } // init()
```

SimpleAudioApplet.java 3(3)

```
  public void start() {
    if (sound != null)
      sound.play();
  } // start()
} // class
```

SimpleAudioApplet.html

```
...
<applet
code="SimpleAudioApplet.class"
width=300 height=300>
Your browser doesn't
support applets.
</applet>
...
```

Appletin elinkaari

- applettiolio luodaan
- appletti alustetaan kutsumalla `init`-metodia
- appletin suoritus aloitetaan kutsumalla `start`-metodia
- jos appletista poistutaan (esim. vaihdetaan sivua, minimoidaan selain), kutsutaan `stop`-metodia
- kun applettiin palataan takaisin, kutsutaan `start`-metodia
- appettioliota poistettaessa kutsutaan `finalize`-metodia

BackgroundMusicApplet.java 1(2)

```
public void init() {
  String name = getParameter("BG_MUSIC");
  try {
    String base = getCodeBase() + "snd/";
    sound = getAudioClip(new
                         URL(base + name));
  } catch (MalformedURLException e) {
    showStatus("Cannot load audio file ”
               + name + ".");
  } // try
} // init()
```

BackgroundMusicApplet.java 2(2)

```
public void start() {
  if (sound != null)
    sound.loop();
} // start()

public void stop() {
  if (sound != null)
    sound.stop();
} // stop()
```

BackgroundMusicApplet.html

```
...
<applet
code="BackgroundMusicApplet.class"
width=300 height=300>
<param name="BG_MUSIC"
value="muzak.wav">
Your browser doesn't
support applets.
</applet>
...
```

Säikeistä

- jos äänitiedostot ovat pitkiä, niiden lataaminen kannattaa siirtää taustalle omaan säikeeseen

→ appletti voi aloittaa suorituksensa heti, eikä sen tarvitse odottaa äänitiedostojen latautumista

- säieolio periytyy joko `Thread`-luokasta tai se toteuttaa `Runnable`-rajapinnan

AudioApplet.java 1(6)

```
base = getCodeBase() + "sounds/";
for (int i = 0; i < sounds.length; i++) {
  String fileName = getParameter("SOUND"
                                  + i);
  if (fileName != null) {
    AudioLoader audioLoader =
           new AudioLoader(fileName, i);
    audioLoader.start();
  } // if
```

AudioApplet.java 2(6)

```
for (int i = 0; i < sounds.length; i++) {
  Button button =
              new Button("Sound " + i);
  button.addActionListener(
                    new ButtonPress(i));
  add(button);
} // for
```

AudioApplet.java 3(6)

```
public void stop() {
  for (int i = 0; i < sounds.length; i++)
    if (sounds[i] != null)
      sounds[i].stop();
} // stop()
```

AudioApplet.java 4(6)

```
private class AudioLoader
                    extends Thread {
  private String fileName;
  private int finger;

  public AudioLoader(String n, int f) {
    setDaemon(true);
    fileName = n;
    finger = f;
  } // constructor
```

AudioApplet.java 5(6)

```
  public void run() {
    try {
      sounds[finger] = getAudioClip(
             new URL(base + fileName));
    } catch (MalformedURLException e) {
      showStatus("Cannot load file "
                 + fileName + ".");
    } // try
  } // run()
} // class
```

AudioApplet.java 6(6)

```
private class ButtonPress
                  implements ActionListener {
  private int soundNumber;

  public ButtonPress(int s) {
    soundNumber = s;
  } // constructor

  public void actionPerformed(ActionEvent e){
   AudioClip sound = sounds[soundNumber];
   if (sound != null) sound.play();
  } // actionPerformed()
 } // class
```

AudioApplet.html

```
...
<applet code="AudioApplet.class"
width=500 height=100>
<param name="SOUND0" value="music0.wav">
<param name="SOUND1" value="music1.wav">
<param name="SOUND2" value="music2.wav">
...
<param name="SOUND5" value="music5.wav">
Your browser doesn't support applets.
</applet>
...
```

Jar-tiedostot

- useista luokkatiedostoista koostuvat appletit on syytä koota yhdeksi jar-tiedostoksi

→ vältetään useiden pienten tiedostojen lataaminen

- esim.

```
jar cvf AudioApplet.jar *.class
```

JarredAudioApplet.html

```
...
<applet code="AudioApplet.class"
archive="AudioApplet.jar"
width=500 height=100>
<param name="SOUND0" value="music0.wav">
...
<param name="SOUND5" value="music5.wav">
Your browser doesn't support applets.
</applet>
...
```

`Applet`-metodien käyttö sovelluksissa

- staattinen metodi `newAudioClip` palauttaa `AudioClip`-olion
  - huom. parametriksi annetaan `URL`-olio eikä esim. tiedostokahva
  - vinkki: sovelluksen oletushakemiston polun saa metodikutsulla `System.getProperty("user.dir")`
- saatua `AudioClip`-oliota voidaan käyttää normaalisti

`AudioApplication.java` 1(3)

```
public class AudioApplication {
  public static void
                    main(String[] args) {
  AudioClip[] sounds =
              new AudioClip[args.length];
  String base = "file:" +
    System.getProperty("user.dir") + "/";
```

`AudioApplication.java` 2(3)

```
for (int i = 0; i < args.length; i++) {
  try {
    sounds[i] = Applet.newAudioClip(
                new URL(base + args[i]));
  } catch (MalformedURLException e) {
    throw new RuntimeException(
              "Cannot load audio file "
              + args[i] + ".");
  } // try
} // for
```

`AudioApplication.java` 3(3)

```
    for (int i = 0;
         i < sounds.length; i++) {
      sounds[i].loop();
    } // for
  } // main()
} // class
```

3.2. `javax.sound.sampled`

- mukana JDK-versiosta 1.3 alkaen
- tarjoaa matalan tason liittymän alustan äänilaitteistoon (myös havainnointi)
- pyrkii silti olemaan alustariippumaton ja yleistettävissä oleva rakennelma
- mahdollistaa äänisignaalin
  - vastaanottamisen (esim. äänitys)
  - käsittelyn (esim. vahvistus tai kaiunta)
  - toistamisen

Pakkaukset

- `javax.sound.sampled`
  - rajapintoja ja luokkia samplatun äänisignaalin tallennukseen, muokkaamiseen ja toistoon
- `javax.sound.midi`
  - rajapintoja ja luokkia MIDI-käyttöön
- `javax.sound.sampled.spi`
- `javax.sound.midi.spi`
  - ulkopuolisille palveluntarjoajille (*service providers*) tarkoitettuja apuluokkia

Piirteitä

- pähkinänkuoressa: äänidataa sisältävien tavujen lukua, kirjoitusta ja operointia
- liittymät syöttö- (esim. mikrofoni tai tiedosto) ja tuloslaitteisiin (esim. kaiutin tai tiedosto)
- äänidatan puskurointi (esim. reaaliaikainen äänivirta)
- äänisignaaleiden yhdistäminen
- käyttäjän komennot: aloita, pysäytä, jatka, lopeta

Äänidatan käsittelytavat

- puskuroitu (*buffered*)
  - virta (*streaming*): reaaliaikaisen äänidatan käsittelyä
  - operoitava (esim. äänitettävä tai käsiteltävä) tavuja likimain samassa tahdissa kuin missä niitä lähetetään
- puskuroimaton (*unbuffered*)
  - äänidata sijaitsee (kokonaisuudessaan) muistissa
  - monipuolisempi toisto: silmukointi, aloituspaikan valinta

Äänidatan formaatit 1(2)

- dataformaatti
  - kertoo kuinka sarja tavuja eli ”raaka” samplattu äänidata pitää tulkita
  - `AudioFormat`-luokka
- tiedostoformaatti
  - määrittelee äänitiedoston rakenteen
  - `AudioFileFormat`-luokka

Äänidatan formaatit 2(2)

- vaikka tarjolla on metodeja
  - erilaisten ääniformaattien muuttamiseen
  - yleisten tiedostoformaattien lukemiseen ja tallentamiseen
- kyse ei silti ole kaikenkattavasta äänityökalusta
  - palveluntarjoajilta tukea ja täydennystä valikoimaan

`AudioFormat`-luokka

- koodaustekniikka (esim. PCM, a-law tai μ-law)
- kanavien määrä (1 = mono, 2 = stereo jne.)
- samplaustaajuus
- kvantisointitaso (so. käytettyjen bittien määrä)
- kehystaajuus (*frame rate*)
  - kehys (*frame*) = kaikki tiettyyn hetkeen kuuluva data; (esim. kanavien nykyiset näytearvot)
- kehyksen koko tavuina
- tavujärjestys: *big-endian* tai *little-endian*

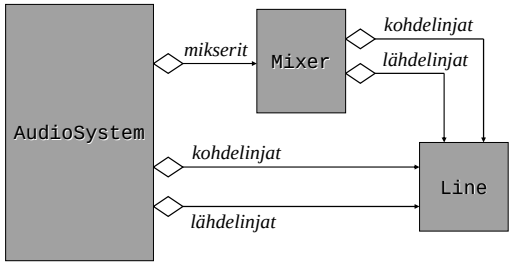
`AudioFileFormat`-luokka

- tiedostotyyppi (esim. WAVE, AIFF)
- tiedoston pituus tavuina
- äänidatan pituus kehyksinä
- `AudioFormat`-olio, joka määrittelee tiedoston sisältämän äänidatan muodon

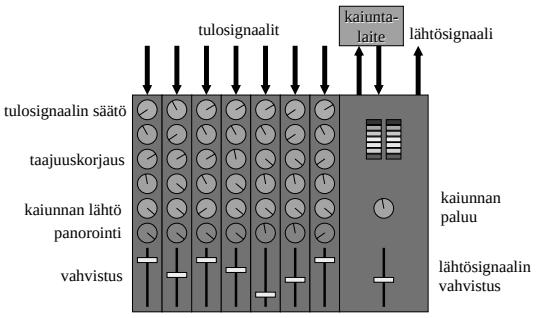
Java Sound -perusosat

- äänijärjestelmä: `AudioSystem`
- mikseri: `Mixer`
- linja: `Line`
- portti: `Port`

Perusrakenne



Analogia: miksauspöytä



Äänijärjestelmä (*audio system*)

- kokoaa yhteen kaikki laitteiston ja ohjelmiston tarjoamat äänipalvelut:
  - mikserit
  - linjat
  - portit
  - äänivirrat
  - tiedostoformaatit
  - ääniformaatit