# Java Sound

Jean-Marc Farinone
(Maître de Conférences CNAM)

# Traitement des sons : historique

- Dès la sortie de Java en 1995, on pouvait exécuter des sons (fichiers sons)
- Mais ...
- Seulement dans les applets
- de format `.au`
- Par contre multi-plateforme (PC, macintosh, Solaris)
- Voir à `http://cedric.cnam.fr/~farinone/AFCET/14.html`

# Traitement des sons désormais

- Utilisable dans les applications comme dans les applets.
- Traite les formats AIFF, AU, WAV, MIDI, et RMF en lecture comme en enregistrement.
- Complètement intégré dans Java SE Java 1.3 (i.e. 100% pur Java) => pas de téléchargement supplémentaire en plus du Java SE (!= Java 3D, JMF, Java Speech).
- **Mais** Java Sound fonctionne bien avec J2SE 1.3 et 1.4 pas bien avec 1.5 ! => télécharger J2SE 1.4 à `http://java.sun.com/javase/downloads/previous.jsp`

# Documentation pour Java Sound

- Page de départ de la technologie Java Sound
  `http://java.sun.com/products/java-media/sound/index.html`
- Tutorial à
  `http://java.sun.com/docs/books/tutorial/sound/index.html`
- Guide de programmation Java Sound à
  `http://java.sun.com/j2se/1.3/docs/guide/sound/prog_guide/javasoundTOC.fm.html`

# Tester les sons avec J2SE

- Après installation du SDK 1.3 on trouve, des démos sons dans `%JAVA_HOME %\demo\sound`
- Ce n'est plus le cas avec Java SE 1.5
- Ces démos sont téléchargeables à `http://java.sun.com/products/java-media/sound/samples/JavaSoundDemo/`
- Lancer la démo par `java -jar JavaSoundDemo.jar`

`2demoJDKSonsAppli.bat`

# Bidouille sécurité Java (1/3)

- Par défaut les applets ne peuvent pas écrire sur le disque local
- Depuis Java 1.2, la sécurité peut être "affiné". On peut autoriser certains droits à certaines applets de certaines provenances, etc.
- Voir tutorial sur la sécurité à : `http://java.sun.com/docs/books/tutorial/security/index.html`

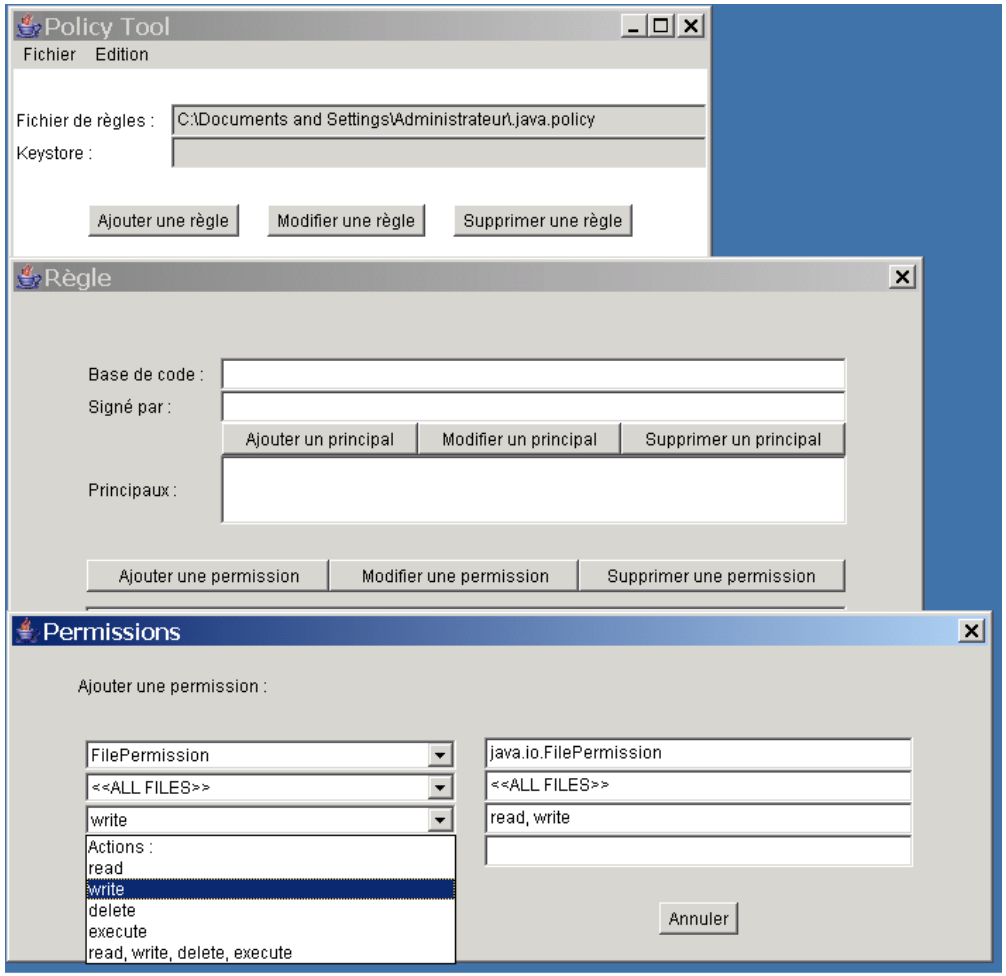
# Bidouille sécurité Java (2/3)

- Pour enregistrer des sons par des applets, le fichier `.java.policy` doit contenir :

```
grant {
  permission java.io.FilePermission "<<ALL FILES>>", "read, write";
  permission javax.sound.sampled.AudioPermission "record";
  permission java.util.PropertyPermission "user.dir", "read";
};
```

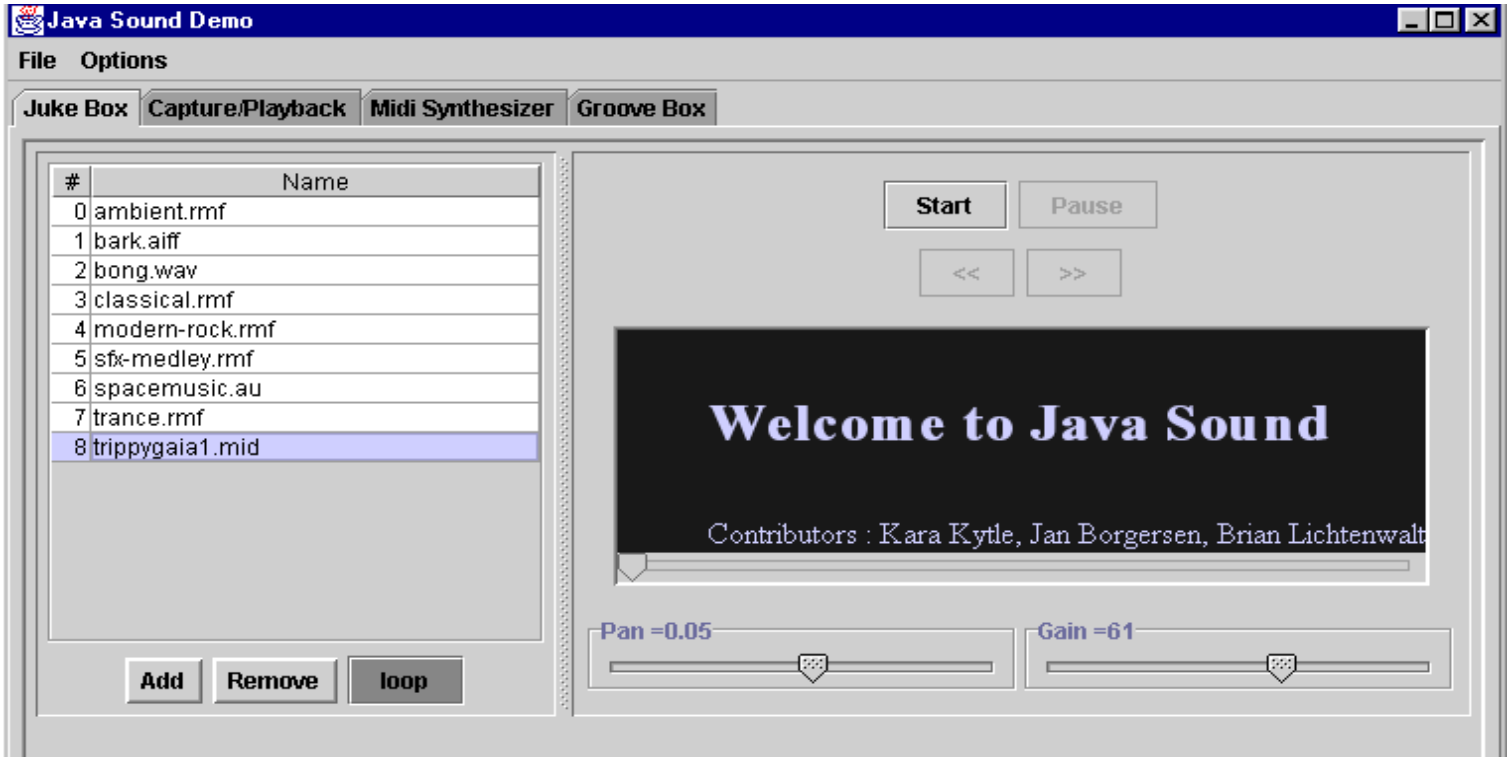
- cf. le `README.txt` dans le répertoire des demos JavaSound
- Utiliser `policytool`

# Bidouille sécurité Java (3/3) : `policytool`



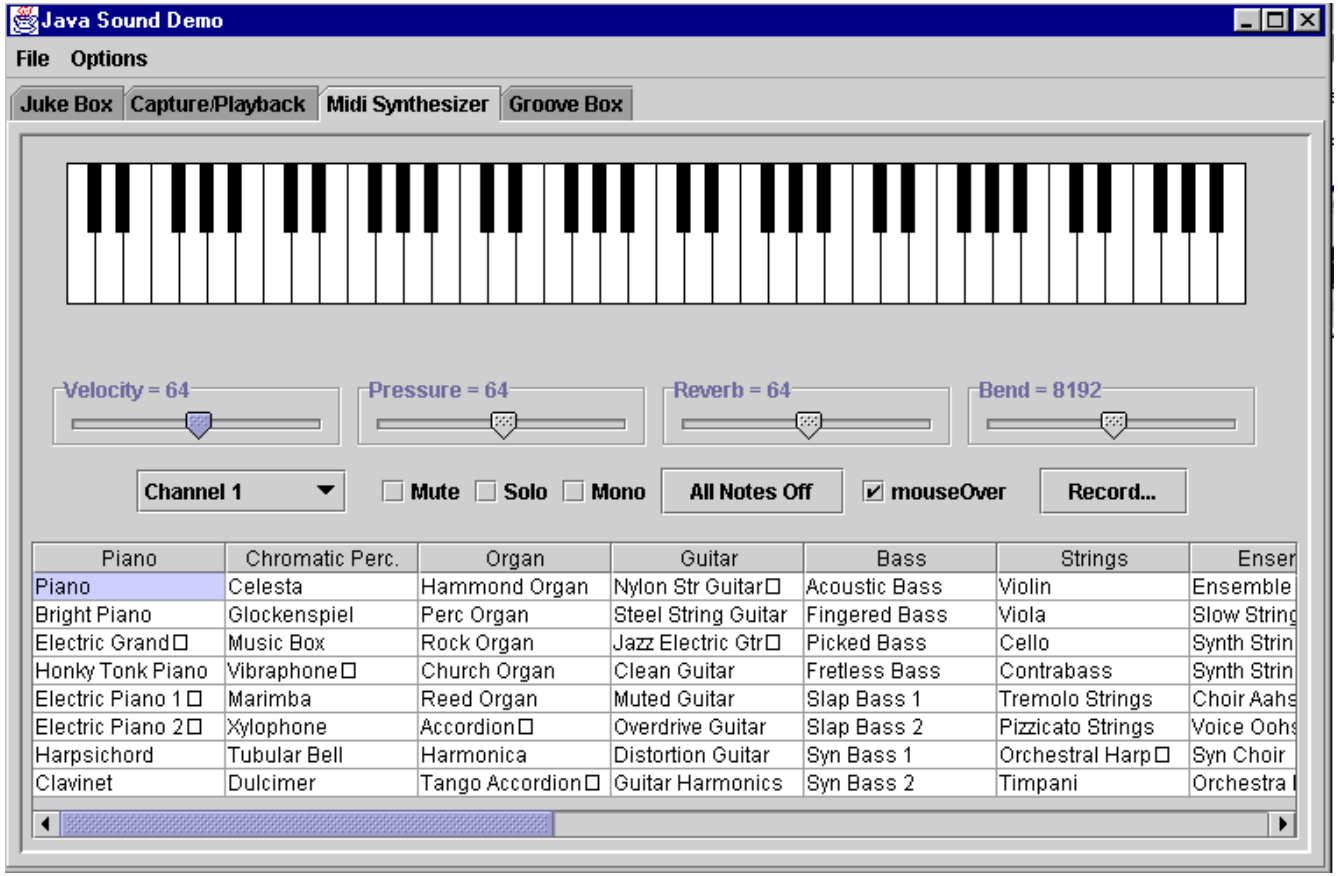
# Tester les sons

- Dans les 2 cas on obtient :



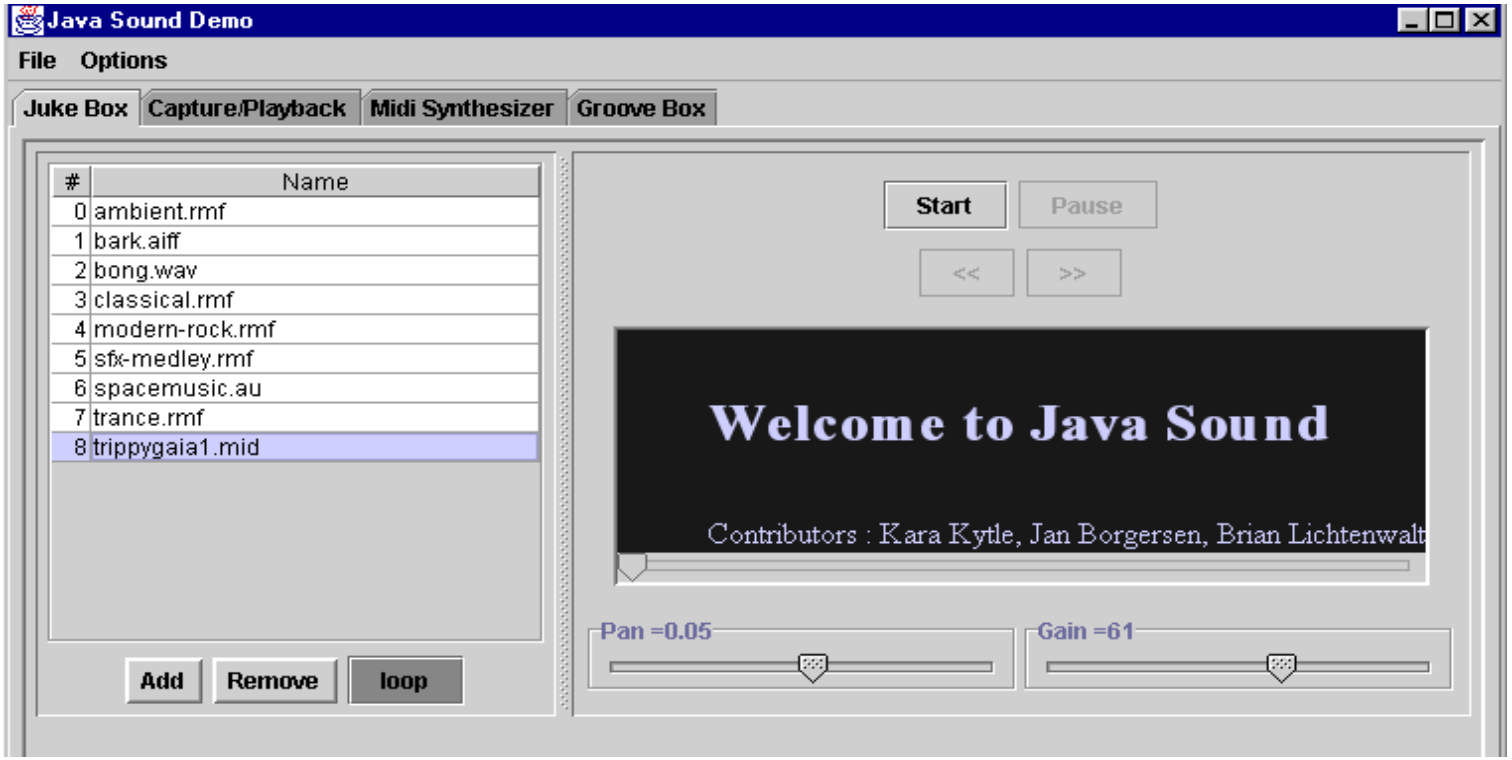
# La demo Java Sound de SUN (1/5)

- La démo propose :
  - lecture des divers formats de fichiers sons (Juke Box)
  - capture audio et sauvegarde en divers formats
  - un synthétiseur MIDI
  - une boite à rythmes (Groove box)



# La demo Java Sound de SUN (2/5)

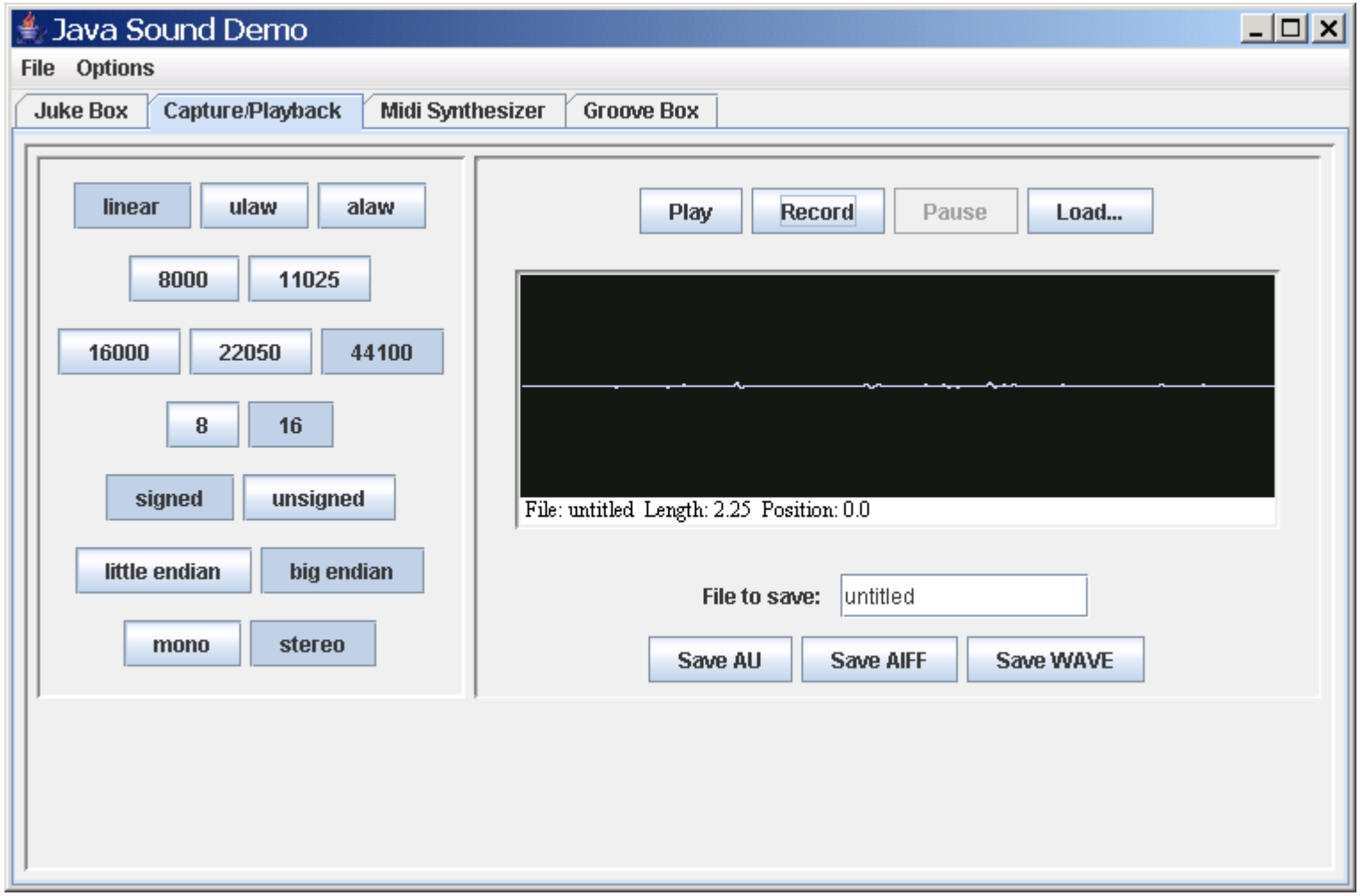
- Le juke box :




- Hum. Pb pour les fichiers `.rmf`, `.mid` avec Java 1.5 ?

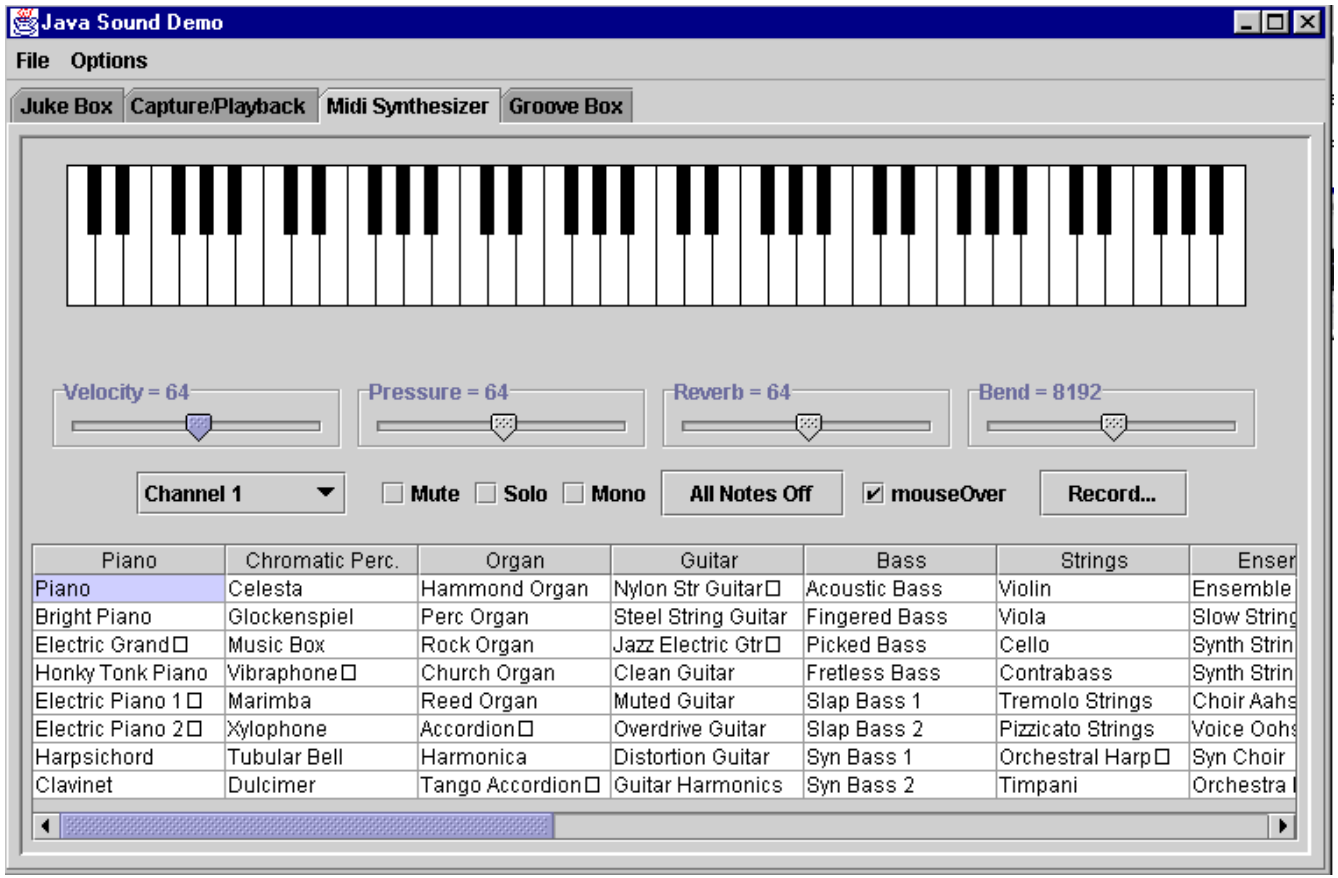
# La demo Java Sound de SUN (3/5)

- La capture, sauvegarde et l'exécution de fichier sons



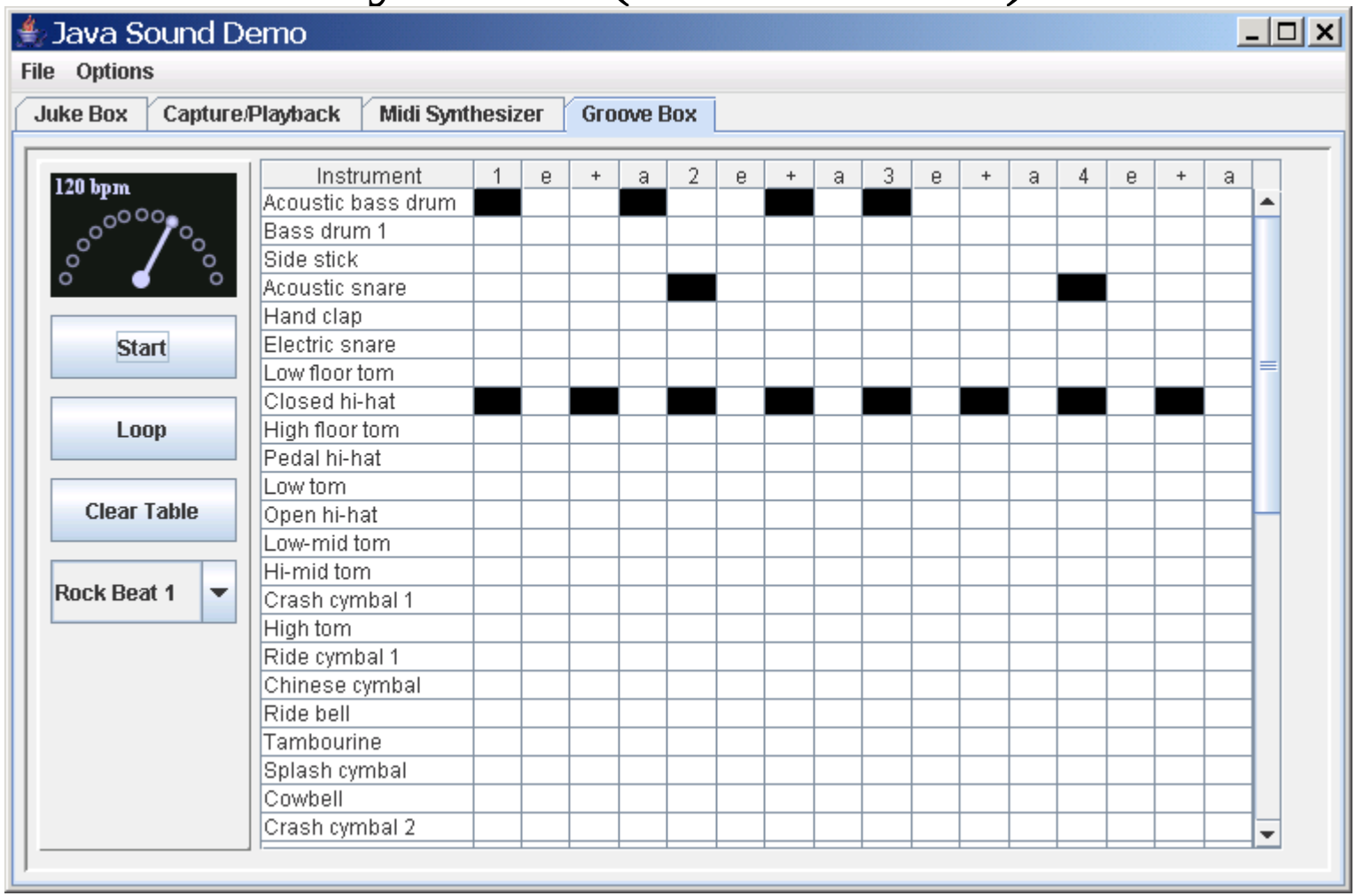
# La demo Java Sound de SUN (4/5)

- Le synthétiseur MIDI



# La demo Java Sound de SUN (5/5)

- La boite à rythmes (Groove Box)




- Appuyer sur Start si on a sélectionné Loop

# Java Sound

- Amène une solution logicielle pour le traitement des sons sur ordinateur et évite des cartes sons adéquates. Seul besoin un digital-to-analog converter (DAC) fourni par les cartes sons.
- Permet de lire (jouer) et écrire (construire de la musique) des fichiers sons.

# Applet Java Sound : syntaxe

- C'est le code de 1995 !!
- Repérer le clip audio par `public AudioClip getAudioClip(URL url)` lancée sur l'objet applet courante en indiquant l'URL du fichier son. Le chargement (synchrone pas de streaming) est fait à ce moment-ci != images.
- Lancer `play()`, `loop()` ou `stop()` sur l'`AudioClip` retourné.
- Remarque : c'est le même code quel que soit le type de fichier sons.

# Applet Java Sound : syntaxe

- On peut aussi utiliser la méthode `public AudioClip getAudioClip(URL url, String nomRessource)`

```
AudioClip leClipAudio;
...
leClipAudio = getAudioClip(getCodeBase(),
    nomAudioClip);

...
leClipAudio.play();
...
leClipAudio.stop();
...
leClipAudio.loop();
```

# Applet exécute son : exemple complet

`appletviewer audios.html`

donne :




`3demoDiversFormatsSons.bat`

- Remarques
  - voir taille des divers fichiers sons

# Applet exécute son : exemple complet

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import java.applet.AudioClip;

public class AudioLoop extends JApplet implements ActionListener {
  AudioClip sound;
  JButton btJoue, btBoucle, btArret;
  JComboBox btDiversFormatsSons;
  String[] tabFichiersSons = {"bottle-open.wav", "flute+hrn+mrmba.aif",
        "spacemusic.au", "trippygaia1.mid"};
  Container contentPane = getContentPane();
```

# Applet exécute son : exemple complet (suite)

```
public void init() {
    btDiversFormatsSons = new JComboBox(tabFichiersSons);
    btJoue = new JButton("joue");
    btBoucle = new JButton("en boucle");
    btArret = new JButton("arrêt");

    contentPane.setLayout(new FlowLayout());
    contentPane.add(btDiversFormatsSons);
    contentPane.add(btJoue);
    contentPane.add(btBoucle);
    contentPane.add(btArret);
    btJoue.addActionListener(this);
    btArret.addActionListener(this);
    btBoucle.addActionListener(this);
  }
```

# Applet exécute son : exemple complet (fin)

```
  public void actionPerformed(ActionEvent evt) {

    if (evt.getSource() == btJoue) {
        sound = getAudioClip(getCodeBase(),
            (String)btDiversFormatsSons.getSelectedItem());
sound.play();
    } else if (evt.getSource() == btBoucle) {
        sound = getAudioClip(getCodeBase(),
            (String)btDiversFormatsSons.getSelectedItem());
sound.loop();
    } else if (evt.getSource() == btArret) {
        sound.stop();
    }
  }
}
```

# Application indépendante Java Sound : syntaxe

- Repérer le clip audio par la méthode statique `public static final AudioClip newAudioClip(URL url)` de la classe `Applet` (!!) en indiquant l'URL du fichier son.
- Lancer `play()`, `loop()` ou `stop()` sur l'`AudioClip` retourné.
- Remarque : c'est le même code quel que soit le type de fichier sons.

# Application indépendante Java Sound : syntaxe

```
AudioClip leClipAudio;
…
URL urlDelAudioClip = new URL("file:" +
    emplacementAudioClip);

leClipAudio = Applet.newAudioClip(urlDelAudioClip);
...
leClipAudio.play();
...
leClipAudio.stop();
...
leClipAudio.loop();
```

- Ce code peut être utilisé pour une applet.

# Exercice

- Construire un seul programme qui soit à la fois une applet et une application indépendante Java Sound
- Construire le fichier HTML associé.

# Java Sound : architecture

- 4 Paquetages :
  - deux concernant MIDI (`javax.sound.midi` et `javax.sound.midi.spi`)
  - 2 concernant les sons échantillonés (`javax.sound.sampled` et `javax.sound.sampled.spi`).

# Java Sound vs. JMF

- Java Sound : exécuter et d’enregistrer des sons échantillonnés (sampled) ou MIDI
- Compression, décompression, synchronisation, transport par anticipation (streaming), transport réseau voir JMF (données temporelles)
- JMF s'appuie sur Java Sound pour exécuter les sons.

# Enregistrer avec Java Sound

# Enregistrer des sons...

- ...échantillonnés
- Traiter par le paquetage `javax.sound.sampled`
- deux notions :
  - format de données sonores (data audio format) : PCM, ALAW, ULAW
  - format de fichier son (file audio format) : WAV, AIFF, AU

# Formats de données sonores (1/3)

- Son = pression du médium ambiant
- Indique comment interpréter la suite des octets représentant un son
- exemple : PCM (Pulse Code Modulation), ALAW, ULAW
- Représentation linéaire de la pression en PCM
- Représentation logarithmique de la pression en ALAW, ULAW => meilleure discrétisation pour les sons de faible puissance

# Formats de données sonores (2/3)

- Objet de la classe `AudioFormat` indiquant
  - le type d'encodage (linéaire ou pas),
  - le nombre de canaux (1 pour monophonique, 2 pour stéréophonique),
  - la vitesse d'échantillonnage, c'est à dire le nombre d'échantillon par seconde et par canal,
  - le nombre de bits pour coder un échantillon,
  - la manière d'ordonner les octets (octets de poids fort en premier ou le contraire),
  - le nombre et la taille des trames (une trame contenant les données sonores de tous les canaux à un instant donné).

# Formats de données sonores (3/3)

```
float sampleRate = 8000;
int sampleSizeInBits = 8;
int channels = 1;
boolean signed = true;
boolean bigEndian = true;
AudioFormat formatDonneesAudio = new
  AudioFormat(sampleRate,
    sampleSizeInBits, channels, signed, bigEndian);
```

- La classe `AudioFormat` permet aussi de connaître les caractéristiques de données audio récupérées

# Formats de fichiers son

- Indique la structure d'un fichier son (données + en-tête, taille, nombre d'échantillons, etc.)
- exemple : WAV, AIFF, AU

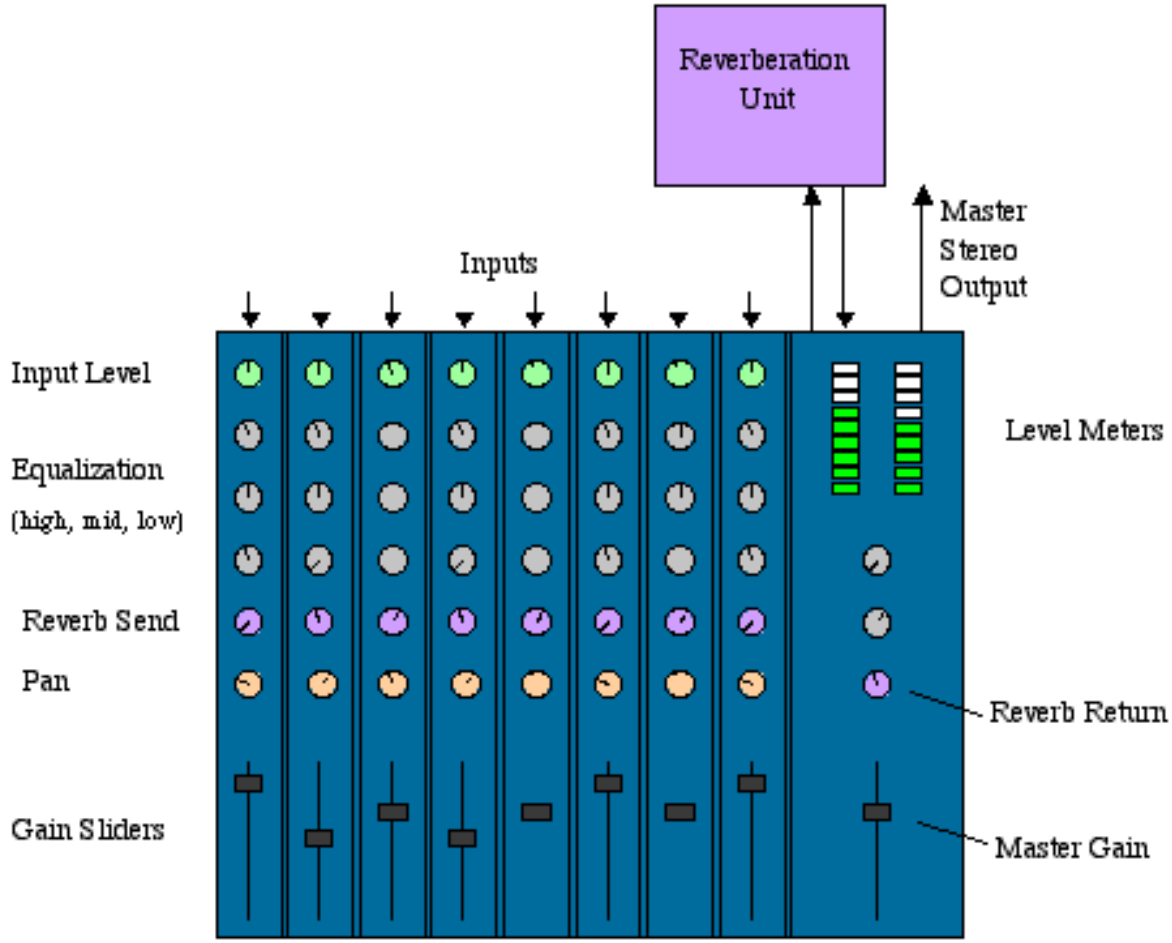
# Formats de fichiers son

- Objet de la classe `AudioFileFormat` indiquant
  - le type de fichier (WAV, AIFF, etc.)
  - la taille du fichier
  - le nombre de trames (= les données de tous les canaux pour un intervalle de temps donné)
  - Les données sonores qu'il contient. Ces données sont modélisées par un objet de la classe `AudioFormat`.

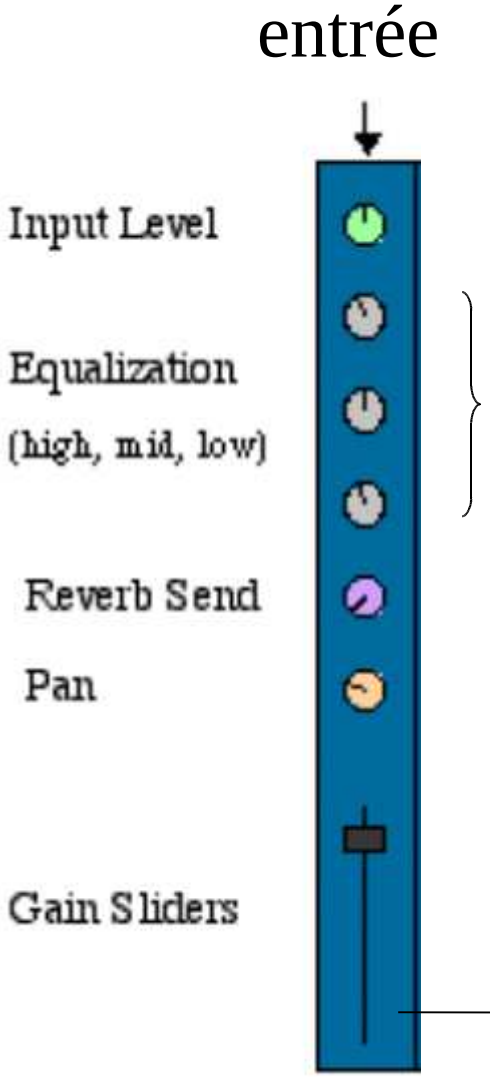
# AudioSystem

- = la classe modélisant la gestion de l'audio sur une machine
- ne contient que des méthodes statiques
- ~ la "fabrique abstraite" pour l'audio.

# Une table (console) de mixage



# Le pipeline de traitement du son




## Les éléments du pipeline

Équaliseur (grave, médium, aigu)

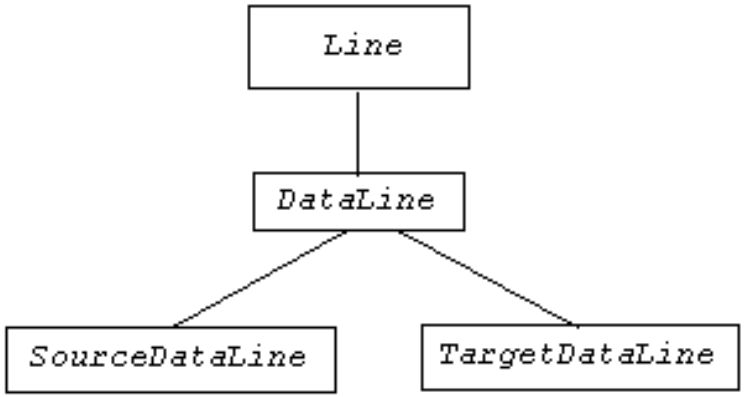
Réverbération.

Panoramique (= balance)

Gain

# Elements du pipeline = `Line`

- Deux exemples de `Line` :
  - `SourceDataLine` = ligne (`Line`) où des données audio peuvent être écrite : HP
  - `TargetDataLine` = ligne (`Line`) d'où des données audio peuvent être lues : microphone
- L'arborescence d'interface :



# Informations pour trouver une `DataLine` : `DataLine.Info`

- On utilise alors

```
public DataLine.Info(Class lineClass, AudioFormat format)
```

- Par exemple :

```
AudioFormat format = new AudioFormat (…);
DataLine.Info info = new DataLine.Info(TargetDataLine.class,
format);
TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
```

# Pour obtenir les données audio du microphone

```
AudioFormat format = new AudioFormat (…);
DataLine.Info info = new
  DataLine.Info(TargetDataLine.class, format);
TargetDataLine line = (TargetDataLine)
  AudioSystem.getLine(info);
line.open(format);
line.start();
int bufferSize = (int)format.getSampleRate() *
  format.getFrameSize();
byte buffer[] = new byte[bufferSize];
line.read(buffer, 0, buffer.length);
```

# Pour envoyer les données audio vers les HP

```
AudioFormat format = new AudioFormat (…);
DataLine.Info info = new
  DataLine.Info(SourceDataLine.class, format);
SourceDataLine line = (SourceDataLine)
  AudioSystem.getLine(info);
line.open(format);
line.start();
int bufferSize = (int)format.getSampleRate() *
  format.getFrameSize();
byte buffer[] = new byte[bufferSize];
line.write(buffer, 0, count);
```

# Capturer des données audio, les écrire dans un fichier

- On capture les échantillons nécessaires à une seconde d'écoute. On écrit les échantillons capturés dans un fichier.
- On vérifie que l'utilisateur n'a pas demandé l'arrêt de la capture. Si non on continue la capture.
- Si oui on arrête la capture par :

```
line.flush();
line.stop();
```

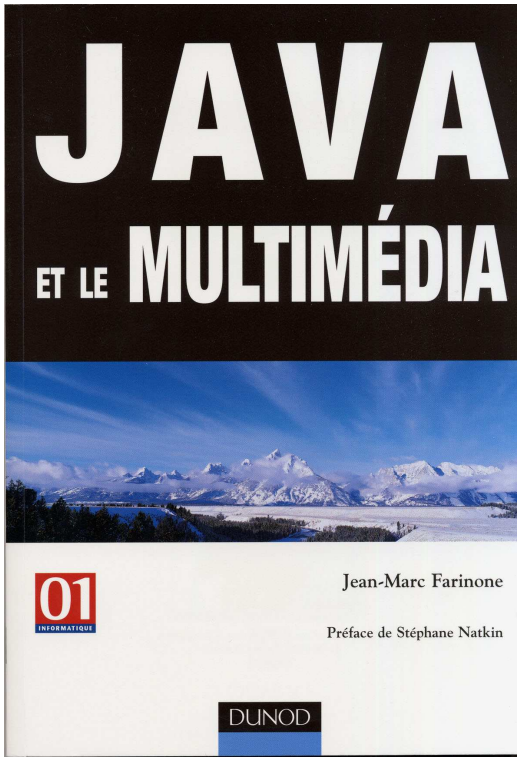
# Ecouter des données audio

- On ne peut pas utiliser les programmes de lecture du début de chapitre. Pourquoi ?
- On lit du fichier construit précédemment, les échantillons nécessaires à une seconde d'écoute. On écrit les échantillons capturés dans la `SourceDataLine` associé au HP.
- Tant qu'on n'a pas atteint la fin du fichier on recommence cette lecture. Après avoir lu tout le fichier on referme la `SourceDataLine` par :

```
line.drain();
line.close();
```

# Capturer et écouter des données audio

- Code complet et détaillé sur :




- démonstration
  `5captureAudioDansFichier.bat`
  `6litFichierAudio.bat`

# Bibliographie

- `http://java.sun.com/docs/books/tutorial/sound/index.html` : le tutorial Java Sound.
- `http://java.sun.com/j2se/1.3/docs/guide/sound/prog_guide/javasoundTOC.fm.html` : le guide de programmation Java Sound

# Fin Java Sound