Medientechnik

Übung 5 - Tutorial

Überblick

- Kurze Wiederholung der Java Sound Konzepte

- Programmieraufgabe: SimpleClipPlayer
  - Clip aus dem Internet laden und abspielen
  - Anzahl der Loops einstellen
  - Liste unterstützter Controls ausgeben
  - Balance/Pan Control

Literaturquellen

- Java `javax.sound.sampled` API
  http://java.sun.com/j2se/1.5.0/docs/api/javax/sound/sampled/package-summary.html
- Java `javax.sound.midi` API
  http://java.sun.com/j2se/1.5.0/docs/api/javax/sound/midi/package-summary.html
- Java Sound API Programmer's Guide
  http://java.sun.com/j2se/1.3/pdf/javasound.pdf
- The Java Tutorials: "Trail: Sound"
  http://java.sun.com/docs/books/tutorial/sound/index.html
- Tutorial "Sound-Programmierung in Java"
  http://web.informatik.uni-bonn.de/IV/strelen/Lehre/Veranstaltungen/prak2000/SoundinJava.doc

Java Sound API

- "Low-level" API
  - zur Steuerung der Ein- und Ausgabe von Tonmedien
  - umfasst Funktionen für digitale Audioinformation und für MIDI-Daten
  - erweiterbare Basis, keine ausgefeilten Editor-Funktionen o.ä.
- Verwandte Java-Technologien:
  - Java Media Framework (JMF)
    - auf höherer Ebene angesiedelt
    - einfachere Lösung für Abspielen von Tonmedien
    - Synchronisation mit anderen Medien (v.a. Video)
- Pakete des Java Sound APIs (in Standard-Java-Installation enthalten):
  - **`javax.sound.sampled`** ← **Fokus dieser Übung**
  - **`javax.sound.midi`**

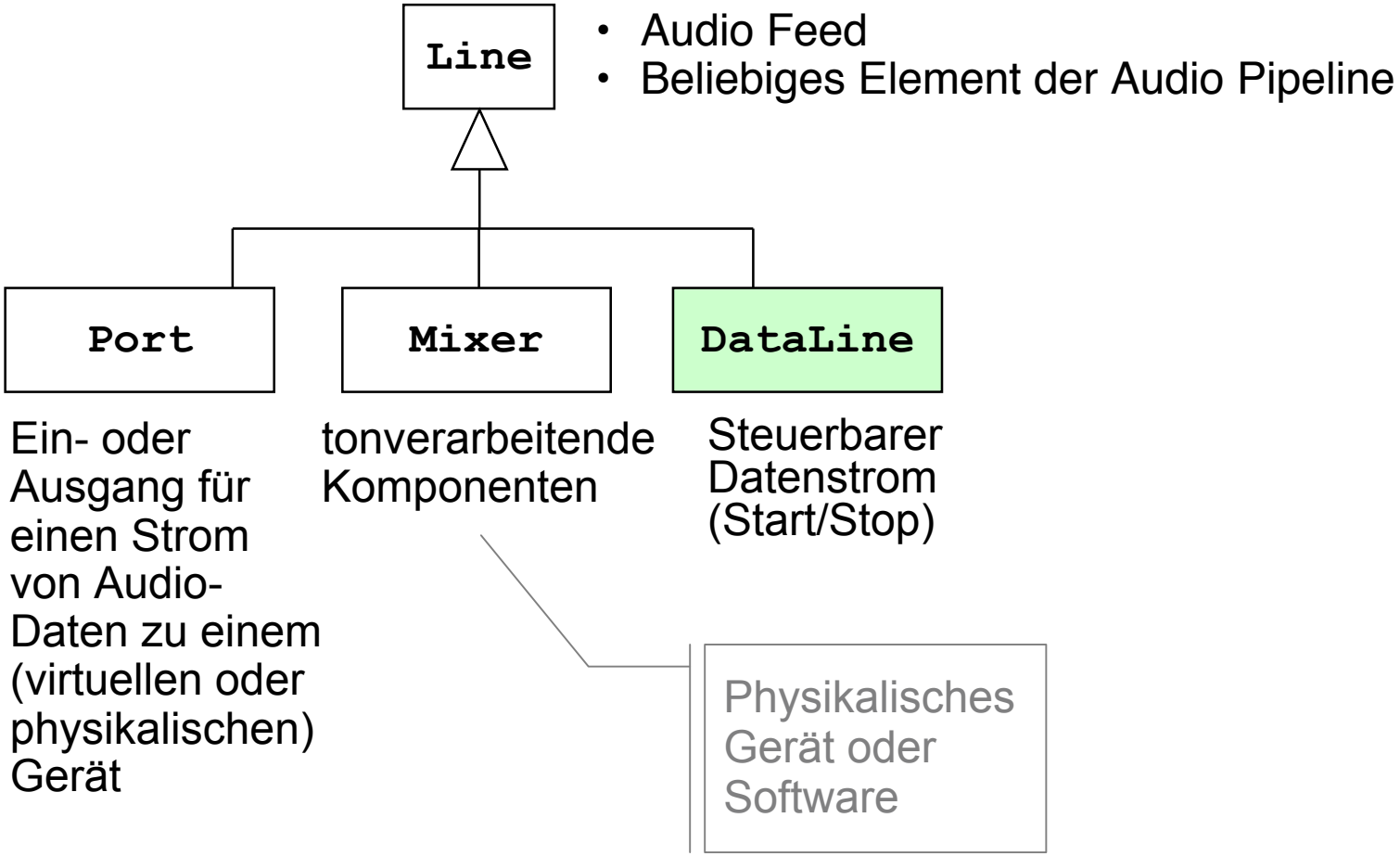
Digitale Audioverarbeitung

- Zeitlicher Verlauf
  - Realzeit-Behandlung:
    - Weitersenden empfangener Information in kurzer Zeit, ungefähr mit der gleichen Rate wie Daten empfangen werden (online)
  - Speicher-basierte Behandlung:
    - Laden der gesamten Information (evtl. stufenweise) in den Computerspeicher
    - Operationen werden erst begonnen, wenn alle Daten bekannt sind (offline)

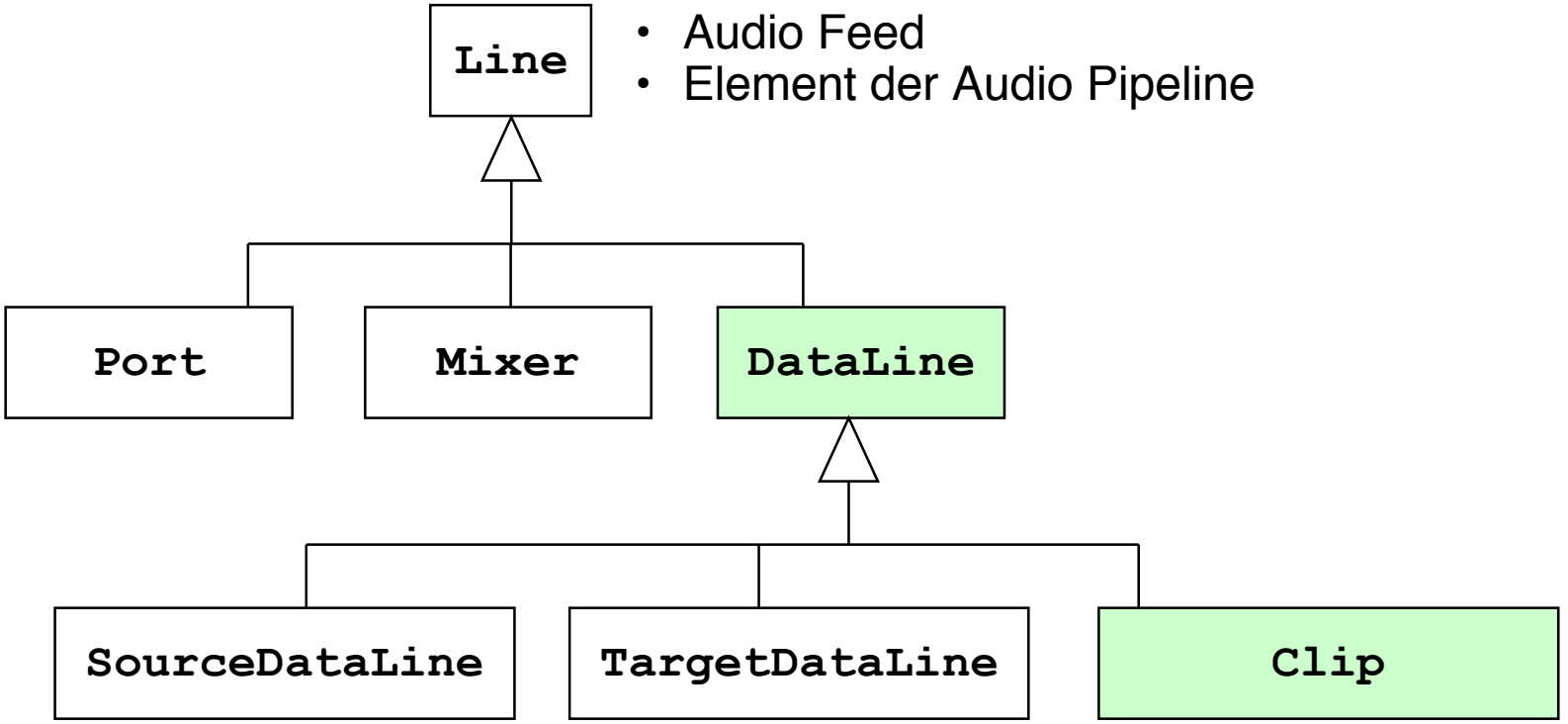
AudioInputStream

- **`javax.sound.sampled.AudioInputStream`**
  - Byteweise lesbare Datenströme
  - **`read(), skip(), close()`**
  - Springen zu markierter Position:
    `markSupported(), mark(), reset()`
- Methoden zum Öffnen einer Audiodatei: javax.sound.sampled.AudioSystem
  - **`AudioInputStream getAudioInputStream(File file)`**
  - **`AudioInputStream getAudioInputStream(URL url)`**
- Ein AudioInputStream hat immer ein festgelegtes AudioFormat
  - Das Format wird z.B. beim Öffnen einer Audio-Datei festgelegt und im Streamobjekt gespeichert.
  - **`AudioFormat getFormat()`**

Line, Port und Mixer



Schnittstellen-Hierarchie für Audio-Pipelines



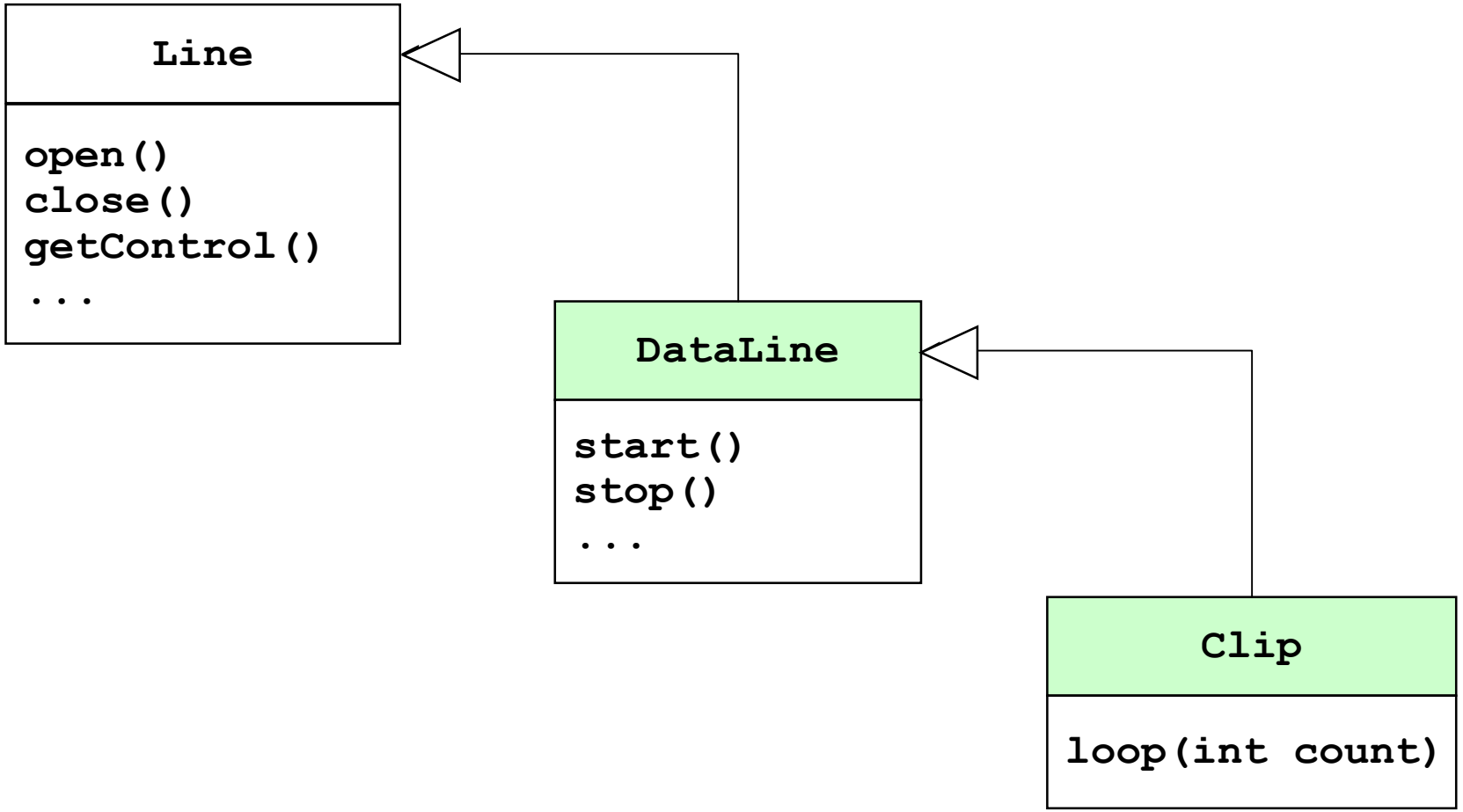
- Audio Feed
- Element der Audio Pipeline

Gepufferte Wiedergabe einer Audioquelle

Gepufferte Aufnahme einer Audioquelle

Wiedergabe einer Audioquelle, die komplett in den Speicher vorgeladen wird

Wichtige Funktionen für heute



Erzeugung eines Line-Objektes

- Typischerweise ist Audioverarbeitung an Systemressourcen gebunden
  - Installierte Geräte
  - Pufferspeicher in Systembereichen
- Erzeugung eines Line-Objekts ausschließlich über zentrale Verwaltung: **`javax.sound.sampled.AudioSystem`**

```
1. AudioInputStream audioInputStream =
   AudioSystem.getAudioInputStream(clipFile/URL);
2. AudioFormat audioFormat =
     audioInputStream.getFormat();
3. DataLine.Info info = new DataLine.Info
     (Clip.getClass(), audioFormat)
4. Clip clip = (Clip)AudioSystem.getLine(info);
```

Nutzung eines Line-Objekts

- Bisher nur die Verfügbarkeit des gewünschten Typs von Line geprüft.
- Nächste Schritte:
  - Reservierung der Ressource mit **`open()`**
    - Reservieren des benötigten Pufferbereichs
    - kann bei gleichzeitigen Anforderungen anderer Programme an Ressourcenmangel scheitern
    - bei **`SourceDataLine`** Angabe eines Audioformats möglich
  - Bei DataLine (d.h. steuerbarem Strom):
    - verschiedene Zustände:
      running = true/false (Übertragung ein/aus)
    - Bei Erzeugung: running = false
    - Starten durch **`start()`**

Threads

Wofür?

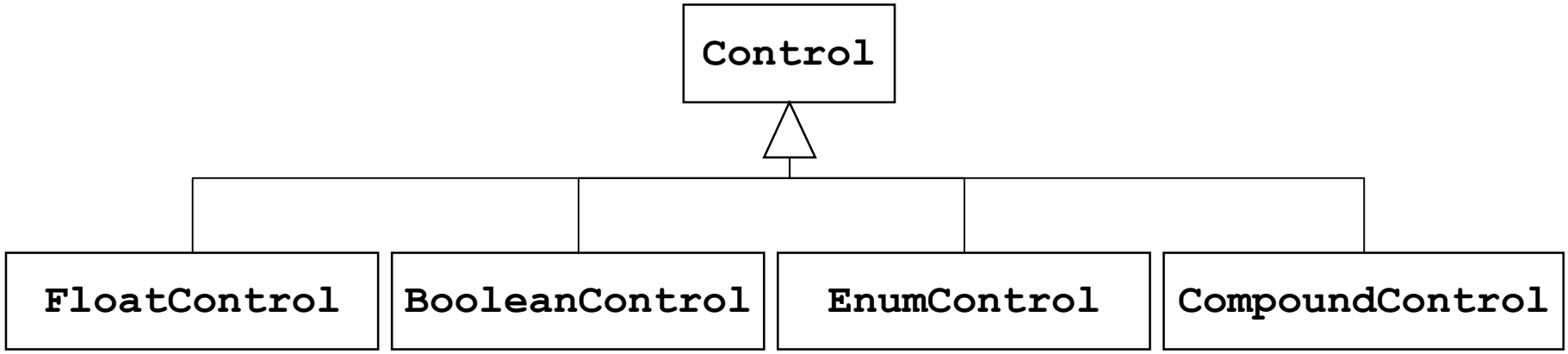
- Nebenläufige Programmierung
  - gleichzeitiges Abspielen von Sound
  - Verändern der Clip-Datei
  - Benutzereingabe
  - …
- Wichtige Methoden
  - **`run()`**
    - Methode in der Thread-Klasse
    - Wird aufgerufen, wenn der Thread gestartet wird durch:
  - **`start()`** startet den Thread von außen

Java Sound Controls

Beschaffung von Control-Objekten:

- Liste aller unterstützen Controls:
  `Line.getControls()`
- Abfrage, ob Control unterstützt wird:
  `Line.isControlSupported(someControlType)`
- Control-Objekt beschaffen:
  `Line.getControl(someControlType)`

Unterklassen von Control



- Manipulieren der Line mit Hilfe des Controls über bestimmte Methoden, z.B.

```
BooleanControl.setValue(boolean newValue)
FloatControl.setValue(float newValue)
```

Java Sound BooleanControl

- BooleanControl: Setzen von boolean Werten
- **`BooleanControl.Type`**

FIELD	Represents a control …
APPLY_REVERB	... for whether reverberation is applied to a line.
MUTE	... for the mute status of a line.

- Abfrage, ob Control unterstützt wird:

  `Line.isControlSupported(BooleanControl.Type.MUTE)`
- Control-Objekt beschaffen:

  `Line.getControl(BooleanControl.Type.MUTE)`

http://java.sun.com/j2se/1.5.0/docs/api/javax/sound/sampled/FloatControl.Type.html

Java Sound FloatControl

- FloatControl: Setzen von Gleitkommawerten
- **`FloatControl.Type`**

FIELD	Represents a control …
AUX_RETURN	... for the auxiliary return gain on a line.
AUX_SEND	... for the auxiliary send gain on a line.
BALANCE	... for the relative balance of a stereo signal between two stereo speakers. [for stereo]
MASTER_GAIN	... for the overall gain on a line.
PAN	... for the relative pan (left-right positioning) of the signal. [for mono]
REVERB_RETURN	... for the post-reverb gain on a line.
REVERB_SEND	... for the pre-reverb gain on a line.
SAMPLE_RATE	... that changes the sample rate of audio playback.
VOLUME	... for the volume on a line.

http://java.sun.com/j2se/1.5.0/docs/api/javax/sound/sampled/FloatControl.Type.html

Programmieraufgabe

- SimpleClipPlayer
  - Clip aus dem Internet laden und abspielen
  - Anzahl der Loops einstellen
  - Liste unterstützter Controls ausgeben
  - Balance/Pan Control