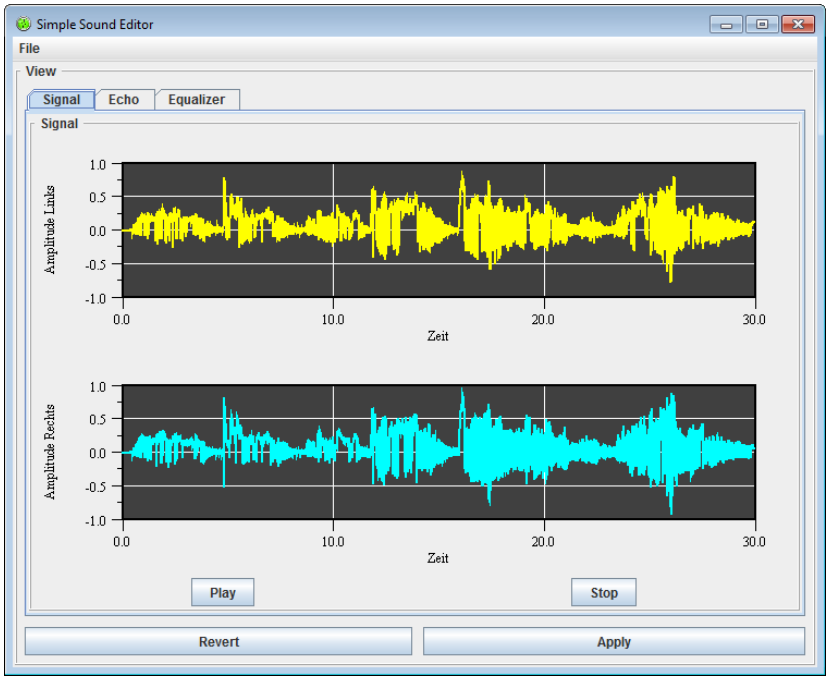
OOP2 MODULSCHLUSSPRÜFUNG FS16

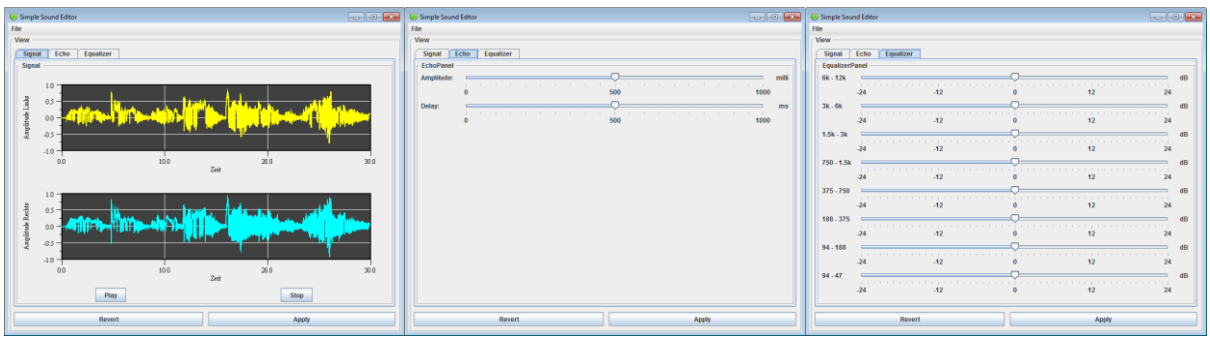
Bedingungen:

- Erlaubte Hilfsmittel: Unterrichtsunterlagen, Java Buch und Übungen.
- **Die Prüfung ist schrittweise, gemäss Aufgabenstellung lokal auf Ihrem Computer zu lösen. Kopieren sie zu diesem Zwecke den gesamten Ordner *MSP_SoundEditor_FS2016_Vorlage* auf Ihre lokale Harddisk und importieren sie das Projekt in Eclipse. Am Ende der Prüfung ist der Ordner *src* umbenannt in *NameVorname* abzugeben.**
- **Setzen sie als erstes Ihren Namen und Vornamen in die Dateien.**
- **Gegenseitiges Abschreiben in irgendeiner Form führt zur Note 1!**
- **Folgend sie bei der Wahl von Variablen exakt den Angaben in der Aufgabenstellung.**
- **Die Beilage mit dem Layout muss unterschrieben zurückgegeben werden!**

Beschreibung:

Ziel dieser Prüfung ist es, ein Programm zum Editieren von Audiodateien zu schreiben. Das Programm erlaubt vorerst, Audiosignale darzustellen und abzuspielen, Echo hinzuzufügen, sowie die Klangcharakteristik mittel Oktav-Equalizer zu verändern. Strukturell ist es so aufgebaut, dass sich leicht weiter Effekte hinzufügen lassen.





Die Applikation ist im klassischen Model - View/Controller Entwurfsmuster, gemäss Klassendiagramm in der Beilage, implementiert.

Die *View* beinhaltet ein *JTabbedPane* mit den drei Tabs "Signal", "Echo" und "Equalizer" sowie zwei *JButton*, mit denen die Bearbeitung angewendet, respektive rückgängig gemacht werden kann. Auf dem Tab "Signal" ist das *SignalPanel* eingebettet, das einen *SignalPlot* sowie zwei *JButton* zum Abspielen/Stoppen beinhaltet. Auf dem Tab "Echo" befindet sich das *EchoPanel*, das die entsprechenden *JLabel* sowie zwei *Slider* zur Einstellung von Signalverzögerung und Amplitude des Echos umfasst. Auf dem Tab "Equalizer" befindet sich das *EqualizerPanel*, das neben den entsprechenden *JLabeln* acht *Slider* zur Gewichtung der verschiedenen Frequenzbänder beinhaltet. Die Einstellung erfolgt dabei im logarithmischen Mass Dezibel (dB). Die Klasse *Slider* erbt von der Klasse *JSlider* und hat per Default eine Beschriftung mit *Major*- und *Minor*-Ticks. Die *View* und das *SignalPanel* verfügen über den *Controller*, um die entsprechenden Aktionen auszulösen.

Der *Controller* leitet allfällige Aufgaben an das *Model* weiter. Die allenfalls benötigte Information bezüglich *Slider*-Einstellungen wird vom *Controller* via *View* ausgelesen. Der *Controller* hat einen *WavPlayer*, mit dem der *SoundTrack* abgespielt werden kann.

Das Model hat den *SoundTrack* sowie einen *Vector<SoundTrack>* mit der Historie der Bearbeitung. Der SoundTrack beinhaltet die notwendige Information eines Stereo-Audiosignales. Der *SoundTrack* wird in einem eigenen Thread unter Verwendung der Klasse *SoundEffects* bearbeitet.

Die Aufgabenstellung führt Sie schrittweise zum Ziel. Folgen Sie daher beim Lösen der Prüfung der Aufgabenstellung.

Achten Sie darauf, dass Ihr Code kompilierbar bleibt!

Aufgabe 1: Aspekt View: Klassen des User - Interfaces (~ 67 Pte.)

View:

Die *View* ist als *GridBagLayout* organisiert und enthält die erwähnten Komponenten. Das benötigte *JTabbedPane* hat die Tabs default-mässig oben und ist wie ein *JPanel* Komponente und Container zugleich. Komponenten werden mittels `jTabbedPane.add(" Tab-Text ", panel);` hinzugefügt. Die *View* verfügt weiter über die beiden *JButton* "Revert" und "Apply".

Das *SignalPanel* ist ebenfalls als *GridBagLayout* organisiert und enthält den *SignalPlot* sowie die beiden *JButton*.

Das *EchoPanle* ist wiederum als *GridBagLayout* organisiert und enthält die beiden *Slider* sowie anonyme *JLabel*.

Das *EqualizerPanel* ist erneut als *GridBagLayout* organisiert und enthält die acht *Slider* sowie anonyme *JLabel*.

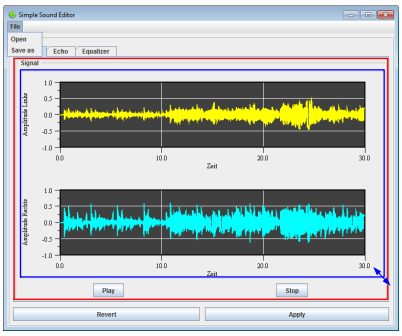


Abbildung 1: *SignalPanel*

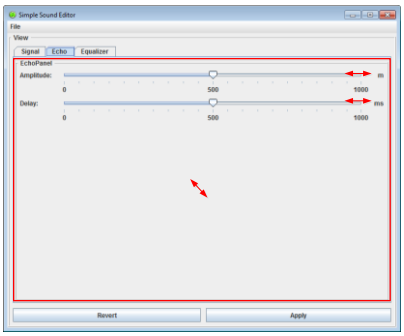


Abbildung 2: *EchoPanel*

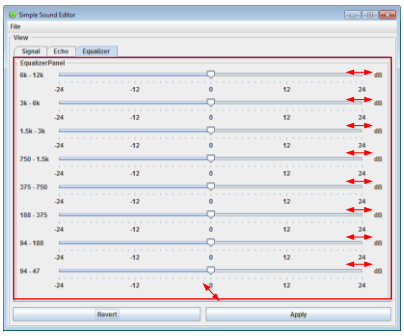
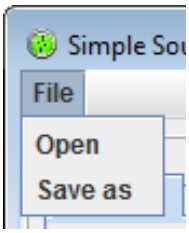


Abbildung 3: *EqualizerPanel*

a) Implementieren Sie die Klassen *View*, *SignalPanel*, *EchoPanel* und *EqualizerPanel* gemäss Dokumentation und Beilage.

MenuBar:

Der Klasse *MenuBar* kommt die Aufgabe zu, folgende Menüs zu realisieren:



b) Implementieren Sie die Klasse *MenuBar* gemäss Beschreibung und Pseudo-Code.

Aufgabe 2: Aspekt Controller: *Klasse zu Steuerung* (~ 16 Pte.)

Controller: Der *Controller* leitet allfällige Aufgaben an das *Model* weiter. Die allenfalls benötigte Information wird vom *Controller* via *View* ausgelesen. Der *Controller* hat den *WavPlayer*, mit dem der *SoundTrack* abgespielt werden kann.

a) Implementieren Sie die Klasse *Controller* gemäss Dokumentation und Pseudo-Code.

Aufgabe 3: Aspekt Model: Klassen zur Berechnung (~ 52 Pte.)

Model: Das Model hat, wie eingangs erwähnt, den *SoundTrack* sowie einen *Vector<SoundTrack>* mit der Historie der Bearbeitung: Der momentane *SoundTrack* befindet sich als letzter in der Historie im *Vector<SoundTrack>*. Beim Rückgängigmachen wird das letzte Element in der Historie entfernt und das neuerdings Letzte als *SoundTrack* gesetzt. Der *SoundTrack* wird in einem eigenen *Thread* unter Verwendung der Klasse *SoundEffects* bearbeitet. Zu diesem Zwecke wird in den entsprechenden Methoden ein *Thread* mit einem anonymen *Runnable*-Objekt erzeugt. Im zugehörigen *run*() wird die statisch Effekt-Methode der Klasse *SoundEffects* aufgerufen.

a) Implementieren Sie die Klasse *Model* gemäss Beschreibung und Pseudo-Code.

SoundEffects:

Die Klasse *SoundEffects* beinhaltet statische Methoden, sowie einen *Equalizer*. Zur Erzeugung der Echos wird der Zirkuläre Puffer *Delay* verwendet. Die Beschreibung der Algorithmen ist auf der nachfolgenden Seite zu finden.

b) Implementieren Sie die Klasse *SoundEffects* gemäss Beschreibung und Pseudo-Code.

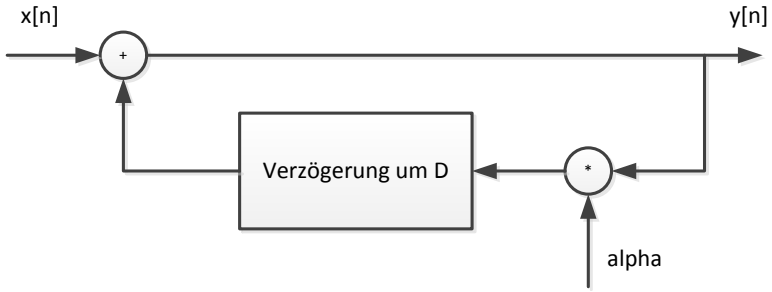
Implementation des Echos:

Ein repetitives Echo entsteht, indem man das Signal gewichtet und verzögert zurückführt. Die Länge der Verzögerung D in Anzahl Abtastwerten ist dabei gegeben durch:

$$D = d \cdot f_s$$

wobei d der Verzögerung in Sekunden und f_s der Abtastrate (*SampleRate*) des Soundtracks entspricht.

Der Berechnung des Echos liegt somit folgendes Blockdiagramm zu Grunde:



Die zugehörige Gleichung wird geschrieben als:

$$y[n] = x[n] + \alpha \cdot y[n - D]$$

Das heisst, es wird zuerst aufgrund von x[n] und dem verzögerten Wert, der Wert y[n] berechnet und anschliessend der mit *alpha* gewichtete Wert y[n] in die Verzögerungsleitung gesteckt.

In unserem Falle setzen wir den Wert $x[n]$ gerade gleich wieder dem Wert von y$[n]$, da wir ja das ursprüngliche Signal durch das echobehaftete Signal ersetzen wollen.

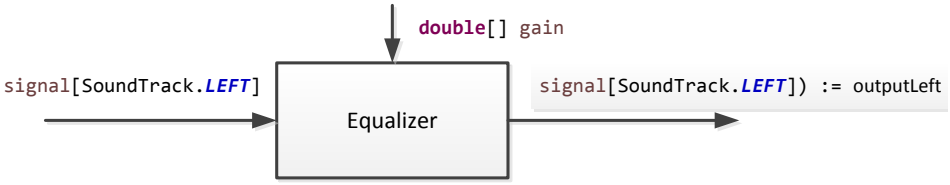
Die Berechnung ist für den linken wie auch für den rechten Kanal zu machen.

Implementation des Equalizers:

Das Equalizer-Panel erlaubt die Verstärkung in den verschiedenen Frequenzbändern in dB einzustellen. Der Zusammenhang zwischen linearem und logarithmischem Mass ist gegeben durch:

$$g_{lin} = 10^{\frac{g_{log}}{20}}$$

Der Equalizer braucht in der Folge das lineare Mass. Der Equalizer ist bereits implementiert und muss nur noch auf die entsprechenden Signale angewendet werden:



Die Verstärkung in den verschiedenen Bändern ist dabei durch den Array *gain* gegeben. Die Berechnung erfolgt sinngemäss auch für den rechten Kanal.

Implementation der Normalisierung:

Die Methode *normalize*() normiert den maximalen Betrag des Signales des linken wie auch des rechten Kanals den Wert *value*. Dies erfolgt in zwei Schritten:

- Maximalen Betrag über alle Werte beider Signale bestimmen.
- Signalwerte beider Kanäle durch das Maximum teilen und mit *value* skalieren.

Aufgabe 4: Clean Up (~ ?)

Gelegentlich geht etwas vergessen:

a) Ergänzen Sie den Code um noch allfällig fehlenden Code.