

Digital Input Computer Keyboard

F.Zeller und E.Hubenschmid

4. Juli 2015

Inhaltsverzeichnis

1	Einleitung	3
1.1	Vorstellung des Projektes	3
1.2	Das Projektmanagement	4
2	Technische Dokumentation	6
2.1	Dokumentation der Storys	6
2.2	Klassendiagramme	13
2.3	Sequenzdiagramme	13
2.4	Struktogramme	15
2.5	Beschreibung einiger Methoden	17
2.6	Javadocs	17
3	Benutzerhandbuch	18
3.1	Einleitung	18
3.2	Projekt	18

1 Einleitung

1.1 Vorstellung des Projektes

Unser Projekt das Digital-Input Computer-Keyboard, ist wie der Name schon vermuten lässt ein Digitales Keyboard. Man kann damit mithilfe seines PCs und einer Tastatur Keyboard spielen. Es kann allerdings auch noch mehr. Der Benutzer hat die Möglichkeit verschiedene Samples (Tonspuren die Instrumente darstellen) einzustellen. Somit kann er theoretisch jedes Musikinstrument über seine Tastatur spielen. Außerdem kann der Benutzer gespieltes aufnehmen und abspielen lassen. Somit kann er mit verschiedenen Instrumenten und Tonspuren eigene Lieder zusammensampeln.

Dieses Projekt hat Emanuel und mir sehr gut gefallen, da wir beide ein Musikinstrument spielen und auch allgemein Musik begeistert sind. Ein eigenes kleines Sample-Programm zu gestalten lag also nahe.

1.1.1 Unser Team

Bei unserem IT-Projekt hatten wir die Möglichkeit Teams zu bilden um größere Projekte umzusetzen. Das Team des Projektes Digital Input Computer Keyboard, besteht aus Emanuel Hubenschmid und Fabian Zeller. Diese beiden sind musikalisch aktiv und hatten somit Interesse daran den Computer mit musikalischer Kreativität zu verbinden. Das Arbeitsklima untereinander war durchgehend angenehm, da jeder seine Arbeit hatte, diese erledigt hat und alles dann gemeinsam zusammengeführt wurde. Es kam also nie zu Streit oder Uneinigkeit über den weiteren Verlauf des Projekts.

1.2 Das Projektmanagement

1.2.1 SCRUM

Definition:

Scrum wurde ursprünglich in der Softwaretechnik entwickelt, ist aber auch in anderen Bereichen einsetzbar. Mittlerweile wird Scrum in sehr vielen Firmen eingesetzt. Scrum basiert darauf das Projekt oder auch die Projektplanung kontinuierlich zu verbessern. Es ist also sehr flexibel und passt sich ständig an. Ziel von dieser kontinuierlichen Verbesserung ist die schnelle und kostengünstige Entwicklung hochwertiger Produkte.

Anfangs hat man nur eine Vorstellung seines Projekts die durch den Kunden oder den Entwickler gegeben wird. Man versucht dann diese Vorstellung anhand von einer Liste mit Eigenschaften festzuhalten. Anders als bei anderen Methoden fertigt man kein Lasten- oder Pflichtenheft an, sondern eben jene Liste. Die Liste dieser Eigenschaften oder auch Anforderungen an das Endprodukt nennt man Backlog.

Diese Anforderungen werden dann intervallartig abgearbeitet. Diese Intervallartigen umsetzungen werden Sprints genannt. Nach Abschluss eines Sprints hat man dann also ein fertiges Teilprodukt des ganzen, dass man testen oder dem Kunden präsentieren kann. Hier fallen dann schon eventuelle Änderungen auf, die dann in das Backlog, bzw. in das nächste Sprint einfließen. Somit verändert sich das Backlog ständig und das Endprodukt wird optimiert.

Zusammengefasst basiert Scrum auf 3 Säulen:

- **Transparenz:** Der Fortschritt und die Verzögerungen werden kontinuierlich festgehalten und man hat immer eine Vorstellung davon wie weit das Projekt fortgeschritten ist.
- **Überprüfung:** Durch die Sprints werden regelmäßig Teilprodukte geliefert die man testen, überprüfen und optimieren kann.
- **Anpassung:** Dadurch dass nicht von Anfang das Endprodukt festgelegt ist, sondern kontinuierlich angepasst wird, entstehen aus einem meist komplexen System viele kleinere Teilsysteme die einfacher zu bewältigen sind.

Wen dieses Konzept interessiert kann sich hier weiter über Scrum erkundigen: [Link](#)

Unsere Erfahrung mit Scrum

Wir haben für unser Schulprojekt Scrum verwendet. Wir haben ein Backlog erstellt und wöchentlich Sprints abgeschickt. Scrum hat uns sehr geholfen unser Projekt zu strukturieren, allerdings waren wöchentliche Sprints nicht optimal. Da wir als Schüler nicht Vollzeit an unserem Projekt arbeiten und die Sprints wöchentlich abzuschicken waren, haben sich manche Stories über mehrere Sprints angestaut. In einem Betrieb oder einer Entwicklerfirma in welcher nur an einem Projekt entwickelt wird sind häufige Sprints sinnvoll, bei nur 2 Leuten und eine quasi Nebenbeschäftigung waren diese doch in zu kurzen Intervallen. Ansonsten war Scrum eine große Hilfe.

1.2.2 GitHub

"Build software better, together."(Motto von GitHub)

Für unsere Projektarbeiten verwendeten wir GitHub. GitHub ist ein webbasierter Hosting-Dienst für Softwareprojekte. Damit Emanuel und ich also bequem von zuhause aus zusammen arbeiten konnten haben wir uns ein Repository in GitHub eingerichtet. Zusammen mit dem Eclipse-Plugin EGit konnten wir unsere Arbeit austauschen und vergleichen. Obwohl es Anfangs Probleme mit der Bedienung und den verschiedenen Funktionen GitHubs und EGits gab, hat es uns doch sehr geholfen und vieles vereinfacht. So konnten wir zum Beispiel gleichzeitig an verschiedenen Problemen arbeiten, indem wir verschiedene Branches (Pfade also Ableger des Projekts) erstellt haben und dann an diesen Branches gearbeitet haben. Sobald dann ein Problem behoben war hat man den Pfad wieder dem Hauptprojekt hinzugefügt und konnte besprechen was genau gemacht wurde und was vielleicht noch verbessert werden muss.

GitHub war uns im allgemeinen eine sehr große Hilfe, aber zu Anfang auch eine große Hürde. Bis wir zurechtkamen mit den Branches, Commits ect. hat es eine Weile gedauert. Man kann aber durchaus behaupten das sich der Aufwand gelohnt hat. Ich würde jedem der ein Softwareprojekt entwickelt empfehlen mit GitHub zu arbeiten, auch wenn er alleine daran arbeitet. Denn EGit zwingt einen dazu seine Änderungen zu dokumentieren und zwischenspeichern. Dies ist zwar etwas nervig, aber man kann sein Projekt immer wieder auf einen beliebigen Standpunkt zurücksetzen wenn etwas komplett schiefgelaufen ist. Dies hat mir oft sehr viel Arbeit erspart.

Gerade jetzt für diese Dokumentation verwenden wir auch GitHub for Windows. Ein Programm für Windows welches einem erlaubt jede Art von Datei über GitHub zu veröffentlichen und zu bearbeiten. GitHub hat uns also sehr viel Arbeit erspart und es uns ermöglicht obwohl wir weit auseinander wohnen ein Projekt auch zusammen zu erarbeiten.

2 Technische Dokumentation

2.1 Dokumentation der Storys

2.1.1 Der Benutzer möchte ein Fenster

1. Das Fenster ist sichtbar
2. Das Fenster hat eine festgelegte gröÙe
3. Das Fenster kann verschoben werden

Storypoints: 2

Bearbeitet von: Fabian Zeller

Beschreibung:

2.1.2 Der Benutzer möchte 3 Sektionen in dem Fenster

1. Die Sektionen sollen beschriftet sein
2. Sektionen untereinander"

Storypoints: 2

Bearbeitet von: Fabian Zeller

Beschreibung:

2.1.3 Der Benutzer möchte, dass in der unteren Sektion Klaviertasten angezeigt werden

1. Die Tasten haben 2 Farben (schwarz/weiß)
2. Die Tasten können angeklickt werden
3. Die Tasten sind wie auf einem Klavier angeordnet

Storypoints: 5

Bearbeitet von: Fabian Zeller

Beschreibung:

2.1.4 Der Benutzer möchte mittels Tasten (Tastatur) einen Klavierton abspielen

1. Die Tasten spielen jeweils genau einen Ton wenn sie geklickt werden
2. Die Tasten können einen Ton halten

Storypoints: 5

Bearbeitet von: Emanuel Hubenschmid

Beschreibung: Zu Begin versuchten wir dies über eine Synthesizer-Klasse der Javabibliothek. Dies erwies sich allerdings als äußerst umständlich. Deswegen entschlossen wir uns die Samples selber zusammen zu stellen und diese als Clips aufzurufen.

2.1.5 Der Programmierer arbeitet sich in Git-Hub ein

1. Das Git-Hub Repository kann von allen Programmierern benutzt werden

Storypoints: 3

Bearbeitet von: Fabian Zeller, Emanuel Hubenschmid

Beschreibung: Um die Zusammenarbeit zu erleichtern verwendeten wir GitHub zur Verwaltung unseres Projektes (Siehe **1.2.2 GitHub**)

2.1.6 Der Programmierer arbeitet sich in Javadocs ein

1. Die Javadoc-Seite soll alle wichtigen Informationen zu dem Programmquellcode enthalten
2. Die Javadoc-Seite soll in der Dokumentation verlinkt werden

Storypoints: 3

Bearbeitet von: Fabian Zeller, Emanuel Hubenschmid

Beschreibung: Wir entschieden uns unseren Quelltext mittels Javadocs auszukommentieren. Dies bot uns die Möglichkeit den Überblick zu behalten und dem jeweils anderem eine einfache Möglichkeit zu bieten den Quelltext besser zu verstehen.

2.1.7 Der Benutzer möchte, dass die gespielte Taste hervorgehoben wird

1. Wenn eine Taste auf der Tastatur betätigt wird, färbt sich die entsprechende Taste auf dem Bildschirm

Storypoints: 5

Bearbeitet von: Fabian Zeller

Beschreibung:

2.1.8 Der Benutzer möchte voreingestellte Samples auswählen können

1. Die Sample sollen die Tasten mit unterschiedlichen Tönen belegen
2. Die voreingestellten Samples sollen Klaviertöne und Schlagzeugtöne abspielen

Storypoints: 5**Bearbeitet von:** Emanuel Hubenschmid

Beschreibung: Das Programm sollte eine Möglichkeit bieten, direkt etwas spielen zu können. Da die Java Sound API nur .wav Dateien mit 8 oder 16 Bit, mussten wir auch solche Sounddateien als Samples verwenden. Die Samples können über Radiobutton ausgewählt werden.

2.1.9 Der Benutzer möchte mehrere Töne gleichzeitig spielen können

1. Die Töne sollen gleichzeitig abgespielt werden können
2. Die Töne sollen nicht durch andere Töne unterbrochen werden
3. Die Töne, die gleichzeitig angespielt wurden, sollen auch gleichzeitig abgespielt werden
4. Die Anzahl der Töne die gleichzeitig gespielt werden können ist auf 3 beschränkt

Storypoints: 5**Bearbeitet von:** Emanuel Hubenschmid

Beschreibung: Damit Töne gleichzeitig abgespielt werden können müssen diese über einen Thread in einem Vektor erzeugt und gestartet werden. Der Key Buffer kann jedoch nicht mehr als 3 Tasten gleichzeitig erfassen.

2.1.10 Der Benutzer möchte, dass in der oberen Sektion Notenlinien angezeigt werden

1. Die Notenlinien bestehen aus fünf Linien
2. Die Notenlinien beginnen mit einem Notenschlüssel

Storypoints: 5**Bearbeitet von:** Fabian Zeller**Beschreibung:**

2.1.11 Der Benutzer möchte Notenlinien, auf denen Noten angezeigt und abgespeichert werden

1. Die Notenlinien speichern das Sample (pro Notenlinie genau ein Sample)
2. Die Notenlinien sollen, mit den gespielten Noten, gespeichert werden können
3. Noten speichern Informationen

Storypoints: 5**Bearbeitet von:** Fabian Zeller**Beschreibung:****2.1.12 Der Benutzer möchte das gespielte Töne als Note angezeigt werden**

1. Noten laufen die Notenlinie entlang
2. Vorzeichen bei bestimmten Noten

Storypoints: 8**Bearbeitet von:** Fabian Zeller**Beschreibung:****2.1.13 Der Benutzer möchte das gespeicherte Notenlinien in einem externen Fenster aufgerufen werden können**

1. Die Notenlinien werden in einem externen Fenster geöffnet

Storypoints: 5**Bearbeitet von:** Emanuel Hubenschmid

Beschreibung: Wird eine Datei abgespielt öffnet sich ein separates Fenster, in welchem der Player die gerade abgespielte Note abbildet.

2.1.14 Der Benutzer möchte gespeicherte Notenlinien abspielen

1. Die Notenlinien können abgespielt, pausiert und gestoppt werden

Storypoints: 8**Bearbeitet von:** Emanuel Hubenschmid

Beschreibung: Die gespeicherten Noten werden aus einer Textdatei ausgelesen und abgespielt. Ein Button kann diese nun pausieren. Wird das Fenster geschlossen, wird auch die Datei gestoppt.

2.1.15 Der Benutzer möchte Noten, die auf den Notenlinien angezeigt und abgespeichert werden

1. Die Noten, die gespielt wurden, sollen auf den richtigen Notenlinien angezeigt werden
2. Die Noten sollen die Tonhöhe speichern

Storypoints: 8**Bearbeitet von:** Fabian Zeller, Emanuel Hubenschmid**Beschreibung:** Wird während der Aufnahme eine Taste gedrückt wird diese in ein Textdokument geschrieben.**2.1.16 Der Entwickler möchte eine Latex Dokumentation**

1. Deckblatt
2. Inhaltsverzeichnis
3. Link auf Javadocs
4. Quellen

Storypoints: 5**Bearbeitet von:** Fabian Zeller, Emanuel Hubenschmid**Beschreibung:****2.1.17 Der Benutzer möchte ein ToolBar-Menü**

1. Die ToolBar enthält ein Hilfe-Menü
2. Die ToolBar enthält ein Projekt Öffnen-Menü
3. Die ToolBar enthält ein Neues Projekt-Menü

Storypoints: 3**Bearbeitet von:** Fabian Zeller, Emanuel Hubenschmid**Beschreibung:** Wir entschieden uns dazu keine Untermenüs zu verwenden, da bei einer so geringen Anzahl an Optionen, dies viel umständlicher für den Benutzer wäre.

2.1.18 Der Benutzer möchte, dass in der Mittleren Sektion Buttons zur Steuerung angezeigt werden

1. Samples können über Radiobutton ausgewählt werden

Storypoints: 3**Bearbeitet von:** Emanuel Hubenschmid

Beschreibung: Ursprünglich waren die Button zur Aufnahme und Wiedergabe ebenfalls in diesem Bereich geplant. Diese sind jedoch nun Projektgebunden und deswegen in dem Projektfenster. Es finden sich hier nurnoch die Radiobutton zur Auswahl der Sample.

2.1.19 Der Benutzer möchte mit der Leertaste alle geöffneten Aufnahmen abspielen

1. Die Leertaste startet oder Stoppt alle geöffneten Aufnahmen

Storypoints: 8**Bearbeitet von:** Emanuel Hubenschmid

Beschreibung: Damit alle Aufnahmen gleichzeitig abgespielt werden können und als “ganzes Lied” gehört werden können. Ermöglicht es die Leertaste alle geöffneten Aufnahmen gleichzeitig abzuspielen.

2.1.20 Der Benutzer möchte eingene Samples benutzen können

1. Die Samples sollen in eine Bibliothek geladen werden können
2. Die Samples sollen vom Benutzer ausgewählt und abgespielt werden können

Storypoints: 5**Bearbeitet von:** Emanuel Hubenschmid

Beschreibung: Am einfachsten und übersichtlichsten erwies es sich, eigene Samples nur mit einem Projekt verwenden zu können. Diese sind dabei immer in dem Projekt-Ortner unter Samples in einem Ortner abzuspeichern.

2.1.21 Der Benutzer möchte einen Projektordner erzeugen können

1. Der Projektordner enthält eine Sample- und einen Aufnahme-Ordner

Storypoints: 8**Bearbeitet von:** Emanuel Hubenschmid

Beschreibung: Wird ein neues Projekt erstellt speichert dies nur den Ordnerpfad in dem sich die Samples und Aufnahmen befinden.

2.1.22 Der Benutzer möchte das die Note einer Taste auf der Taste angezeigt wird

1. Sichtbare Note auf den Tasten

Storypoints: 5

Bearbeitet von: Fabian Zeller

Beschreibung:

2.1.23 Der Benutzer möchte die Samples wechseln

1. Die Samples sollen mittels RadioButton ausgewählt werden können,
2. Die RadioButton sollen für jede Taste das gleiche Sample auswählen
3. Die Samples sollen mit Tasten gewechselt werden können"

Storypoints: 5

Bearbeitet von: Emanuel Hubenschmid

Beschreibung: Ist ein Radiobutton ausgewählt wird der Ordnerpfad des jeweiligen Samples benutzt um den Clip zu starten.

2.2 Klassendiagramme

2.3 Sequenzdiagramme

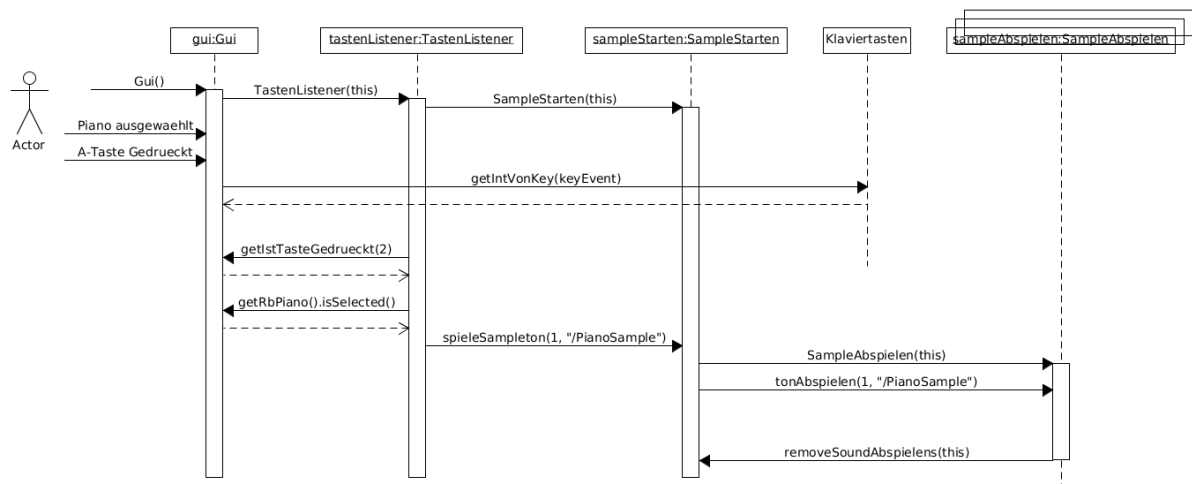


Abbildung 1: Sequenzdiagramm für das Abspielen eines Tones

Beschreibung:

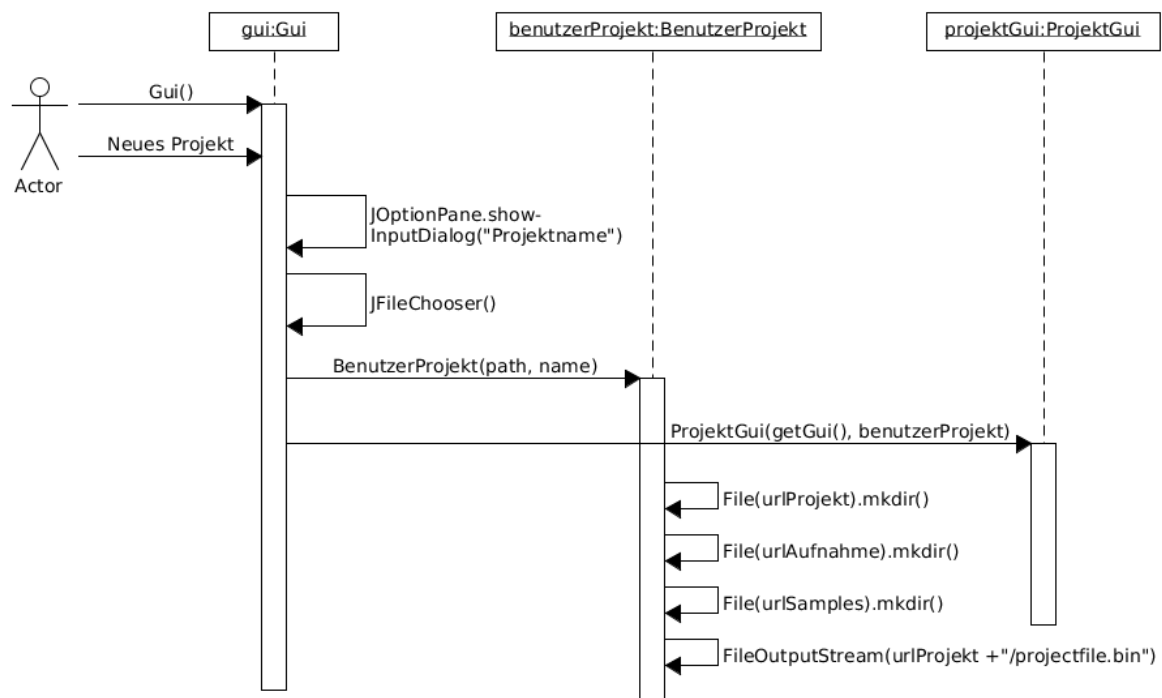


Abbildung 2: Sequenzdiagramm für das Abspielen eines Tones

Beschreibung:

2.4 Struktogramme

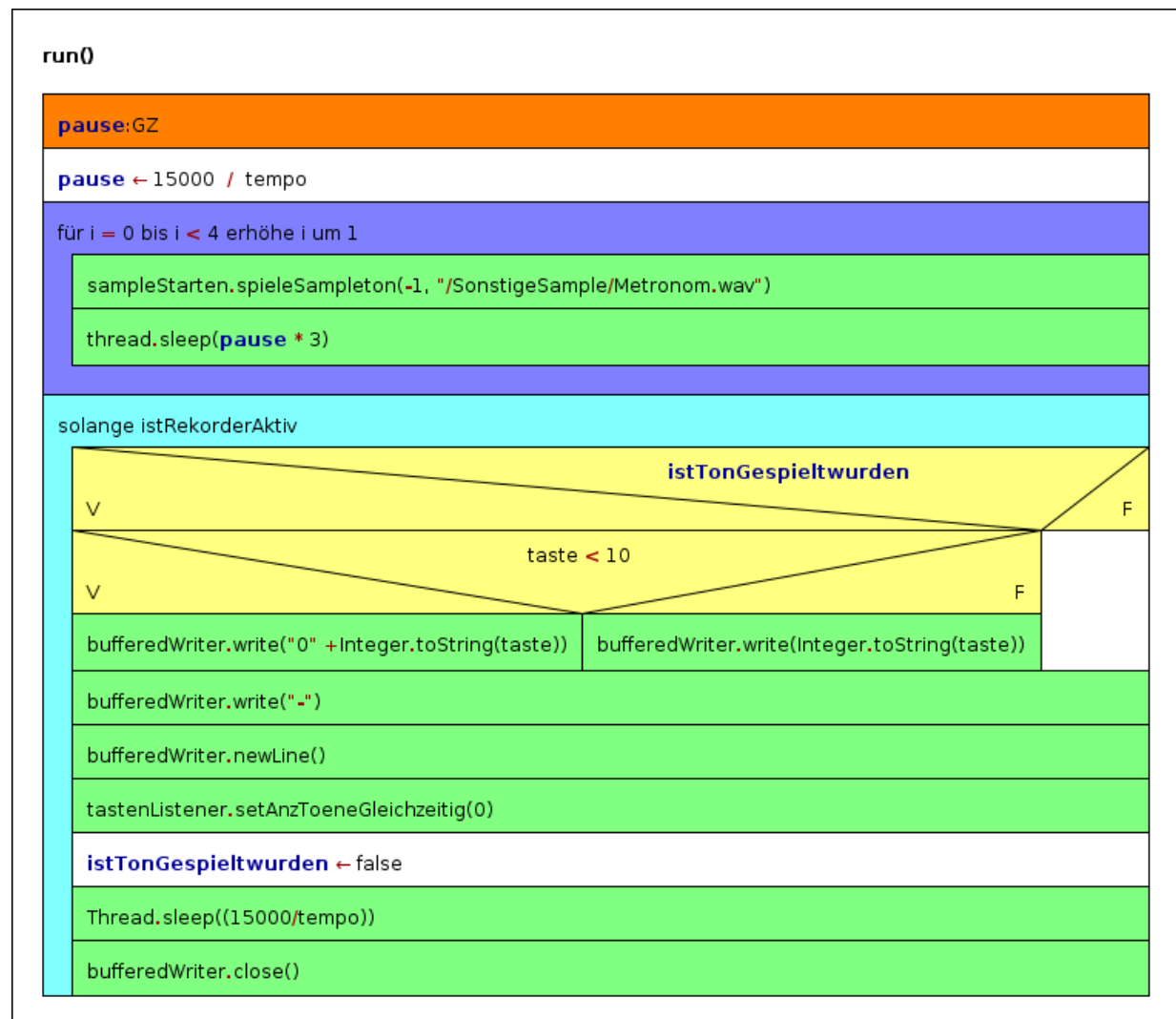


Abbildung 3: Sequenzdiagramm für das Abspielen eines Tones

Beschreibung:

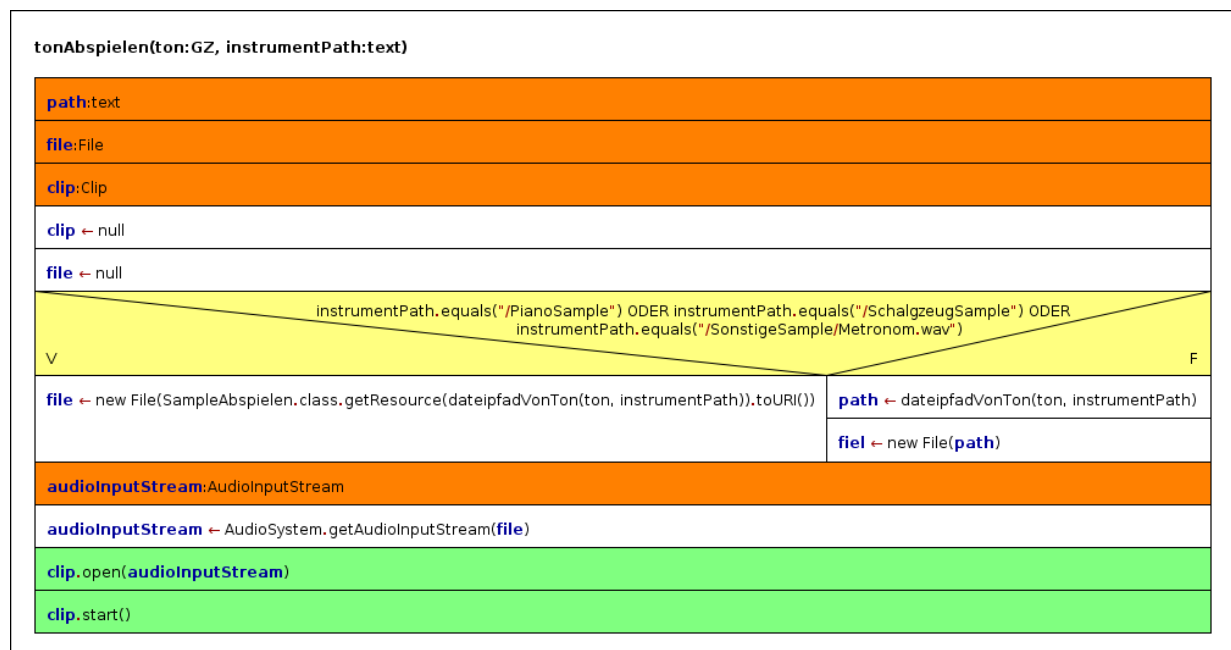


Abbildung 4: Sequenzdiagramm für das Abspielen eines Tones

Beschreibung:

2.5 Beschreibung einiger Methoden

Wir haben während des Projekts alle Methoden mithilfe von Javadocs dokumentiert. Unser komplettes Project ist also hier einsehbar:

Einige größere Methoden werden wir in dieser Dokumentation allerdings etwas genauer beschreiben.

2.6 Javadocs

Der komplette Quelltext wurde mittels Javadocs auskommentiert. Hier kann nachgelesen werden, welche funktion die einzelnen Methoden erfüllen, sowie ihre Übergabe- und Rückgabeparameter, bzw. Exceptions eingesehen werden.

Die komplette Javadocs Dokumentation kann hier geöffnet werden:

[./JavaDocs/index.html](#)

3 Benutzerhandbuch

3.1 Einleitung

Das Digital Input Computer Keyboard ist eine Art elektronisches Keyboard, dass verschiedene Samples abspielen und aufnehmen kann. Man kann mithilfe der gegebenen Samples (Piano und Schlagzeug) spielen und Tonspuren aufnehmen. Aufgenommene Tonspuren können dann auch parallel abgespielt werden. Es können aber auch eigene Samples in das Programm geladen werden. Mit diesen kann man ebenso verfahren wie mit den gegebenen.

3.2 Projekt

3.2.1 Erstellen

3.2.2 Öffnen

3.2.3 Aufnehmen

3.2.4 Apsielen

3.2.5 Eigene Samples