README.md

piano

GitLab: https://dev.hftm.ch:4430/david.burkhart/piano

Projekt aufsetzen:

```
git clone https://dev.hftm.ch:4430/david.burkhart/piano.git
cd piano
mvn clean
mvn compile
mvn package
```

Projekt starten:

```
java -jar target/piano-0.1.jar
```

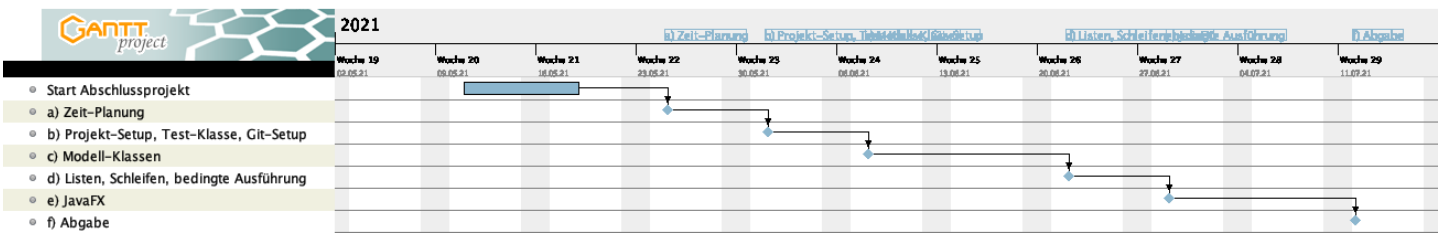
Inhaltsverzeichnis Dokumentation

a) Zeit-Planung

Von der Projekt-Aufgabe her wurden bereits fünf Meilensteine definiert (*a, b, c, d, e*). Daher machte es für mich am meisten Sinn diese bereits definierten Meilensteine mittels einem Gantt-Diagramm auf eine Zeitliche Achse zu verteilen.

Als Start des Projekts nahm ich den Tag, an dem der Dozent das Abschlussprojekt vorstellte und uns den ersten Auftrag erteilte. Dieser Tag war Dienstag, 11. Mai 2021. Als Ende des Projekts nahm ich den Tag der Abgabe des Projekts. Dieser Tag ist Samstag, 10. Juli 2021. Dies konnte ich auf dem Kurs im Moodle entnehmen. Dieser Tag habe ich als Meilenstein mit dem Namen *f) Abgabe* in das Diagramm aufgenommen.

Im Moodle konnte ich auch den Abgabe-Termin für den ersten Meilenstein *a) Zeit-Planung* entnehmen. Dieser ist am Dienstag, 25. Mai 2021. Zwischen dem Abgabetermin des ersten Meilensteins *a)* und dem Meilenstein *f) Abgabe* liegen sechs Wochen. Somit konnte ich die restlichen vier Meilensteine auf diese vier Wochen verteilen. Zuerst setzte ich die Meilensteine in einem Abstand von einer Woche, von Dienstag zu Dienstag auf den Zeitstrahl. Danach verschob ich den Meilenstein *d) Listen, Schleifen, bedingte Ausführung* eine Woche nach hinten. Nach meinem Verständnis wird beim Meilenstein *d)* die Logik der Applikation umgesetzt, was voraussichtlich am meisten Zeit der Implementation in Anspruch nehmen wird. Nun liegen auch zwischen dem Meilenstein *e) JavaFX* und dem letzten Meilenstein *f) Abgabe* zwei Wochen. Diese zwei Wochen werden für den Abschluss des Projektes benötigt und dienen auch als Reserve. Da am Schluss oft noch an Details gearbeitet wird, erachte ich es auch hier als sinnvoll mehr Zeit einzuplanen.



1. Ziele

- Meilensteine auf einen Zeitstrahl legen.

- Zeitliche Übersicht um einen IST-SOLL-Vergleich zu ermöglichen.
- Erste Gedanken machen zu den Aufgaben innerhalb der einzelnen Meilensteine.

Produkte:

- Gantt-Diagram mit Meilensteinen

2. Alternativen / Risikoanalyse

Alternativ zu einem Zeitplan anhand eines Gantt-Diagrams, gäbe es die Auflistung der Meilensteine und den dazu geschätzten Aufwand. Da so aber die einzelnen Schritte kein "Abgabedatum" haben, wäre es weniger gut nachvollziehbar, ob man noch auf dem richtigen Weg ist und/oder es zeitlich schaffen wird.

Eine andere Alternative wäre keine Zeitplanung zu haben sondern ohne zeitliche Aufteilung oder Anhaltspunkte auf den Abgabetermin hin zu arbeiten. Bei kleineren Projekten kann diese Methode unnötige administrative Einschränkungen aushebeln, führt aber dazu, dass nur nach besten Wissen und Gewissen auf das Ziel hin gearbeitet wird.

3. Entwicklung / Test

Die Umsetzung des Gantt-Diagrams wird mit dem Gratis-Program "GanttProject" gemacht. Ich habe mich für dieses Program entschieden, da ich bereits Erfahrungen damit in anderen Projekten sammeln konnte. Ausserdem erlaubt es das Program den Plan als Bild oder als PDF zu exportieren, was sehr hilfreich ist für die Dokumentation.

Das Resultat sieht befriedigend aus und erfüllt meine Anforderungen an einen einfachen und übersichtlichen Zeitplan mit den Meilensteinen.

4. Planung nächster Meilenstein

Mit dem bevorstehenden Schritt wechselt es auf die technische Ebene. Für den nächsten Schritt müssen gewisse Informationen über das Projekt vorhanden sein. Dies betrifft konkret den Projekt-Namen und die damit verbundene Programm-Idee.

Ich habe mich entschieden ein virtuelles Klavier zu erstellen, welches Töne mittels der MIDI-Schnittstelle Töne ausgeben soll, die Java zur Verfügung stellt.

Aus dieser Idee ergibt sich der Projekt-Name "piano".

b) Projekt-Setup, Test-Klasse, Git-Setup

In diesem Schritt werden die Grundsteine für die Umsetzung der Applikation gesetzt. Diese Sachen werden am Anfang aufgesetzt und sollten während der Umsetzung nicht mehr oder nur noch minimal angepasst werden müssen.

1. Ziele

- JavaFX-Projekt mittels Maven aufsetzen (`pom.xml`).
- Projekt-Repository auf dem von der Schule zur Verfügung gestellten GitLab erstellen.
- Dem Dozenten/Ansprechspartner Zugriff zum Repository gewähren.
- Initiale Java-Projekt-Struktur sowie die Dokumentation dem Repository hinzufügen.
- Eine Test-Klasse erstellen um den Einstieg in das Java-Program sicherzustellen.

2. Alternativen / Risikoanalyse

Porjekt-Setup

Anstelle von Maven könnte man auch keinen Dependency-Manager gebrauchen und die benötigten Resourcen (z.B. JavaFX) manuel einbinden und in Java zur Verfügung stellen.

Dies hat den Nachteil, dass die Abhängigkeiten auf jedem System auf dem das Projekt aufgesetzt wird, auf ein Neues manuel eingerichtet werden müssen. Durch dass diese Abhängigkeiten mittels Maven auf jedem Gerät auf die gleiche Weise verwaltet werden können, finde ich den Gebrauch von Maven die angemessenste Lösung.

Git Setup

Anstelle vom GitLab der HFTM könnte ich das Projekt auf meinem privaten GitHub aufsetzen. Eine weiter Alternative wäre es, das Git-

Repository auf meinem privaten Web-Server zur Verfügung zu stellen.

Dadurch vermischt sich aber das Private und das Studium, was ich an dieser Stelle verhindern will. Darum setze ich das Git-Repository auf dem GitLab der HFTM auf.

Test-Klasse

Die Test-Klasse kann für verschiedene Dinge genutzt werden. Man kann während der Umsetzung des Projekts immer wieder die Test-Klasse erweitern um die neue Funktionailtät des Programs zu testen und vereinfacht zu veranschaulichen.

Da meine Applikation hauptsächlich von den Funktionalität der Benutzerobefläche bestimmt wird, sehe ich das Aufsetzen einer solchen Test-Klasse nicht als zwingend. Es würde viel zusätzlichen Aufwand mit sich bringen um jegliche Funktionen zusätzlich in einer solchen Test-Klasse zum Testen einzubinden.

Ich entscheide mich darum gegen eine solche Test-Klasse und werde diese nur nutzen um am Anfang der Projekt-Initialisierung sicherzustellen, ob ich die Java-Applikation wie gewünscht starten kann.

3. Entwicklung / Test

Das Git-Setup war das Erste, dass ich gemacht habe, als ich mit dem Projekt begann. Der Grund dafür war, dass ich von Anfang an auch die Dokumentation auf dem Git-Repository haben wollte. Somit erstellte ich ein leeres Git-Repository und startete mit dem Aufsetzen der Dokumentation. Für das Projekt-Setup nehme ich ein bereits gemachtes Maven-JavaFX-Projekt von der Schule als Vorlage (`pom.xml` , Ordnerstruktur, ...) als Vorlage. Ich starte die Applikation und schaue ob die JavaFX-Komponenten geladen werden können und angezeigt werden. Nach einem erfolgreichen Start, hat die Test-Klasse keinen weiteren Nutzen mehr.

4. Planung nächster Meilenstein

Im nächsten Schritt werden die verschiedenen Modell-Klassen erstellt, sprich, die verschiedenen Entitäten (Dinge) des Programms mit Java-Klassen und/oder Enums abgebildet.

Da ich die Dokumentation nicht während der Umsetzung nachgeführt habe, endet die Dokumentation hier. Meine Prioritäten bezüglich dem Projekt waren sehr selbstlos. Dadurch leidete mein eigenes Projekt. Ich hoffe trotzdem, dass beim Klavier-Spielen am PC Freude aufkommen kann.

c) Modell-Klassen

1. Ziele

2. Alternativen / Risikoanalyse

3. Entwicklung / Test

4. Planung nächster Meilenstein

d) Listen, Schleifen, bedingte Ausführung

1. Ziele

2. Alternativen / Risikoanalyse

3. Entwicklung / Test

4. Planung nächster Meilenstein

e) JavaFX

1. Ziele

2. Alternativen / Risikoanalyse

3. Entwicklung / Test

4. Planung nächster Meilenstein