Dokumentation Projekt M326

Inhaltsverzeichnis

Inhaltsverzeichnis 1

Einleitung 2

Aufgabenstellung 2

Aufgabenplanung 2

Realisierung 4

Analyse - OOA 4

Design - OOD 4

Implementation - OOP 5

Dokumentation 5

Programmbeschreibung 6

Testkonzept 7

Fazit 8

Was haben wir erreicht? 8

Was konnten wir nicht fertigstellen? 8

Welche Probleme hatten wir? 8

Was muss nächstes Mal besser gelingen? 8

Was haben wir gelernt? 8

Fazit David 8

Fazit Silvan 9

Fazit Simon 9

Fazit Marco 9

Fazit Remo 10

Schlusswort 10

Anhang 10

# Einleitung

Im Modul 326 der Lehre als Applikationsentwickler gilt es eine Projektarbeit nach der objektorientierten Vorgehensweise zu realisieren. Dies beinhaltet die objektorientierte Analyse, das objektorientierte Designen und die objektorientierte Programmierung. Die Projektarbeit dient dazu, dass die Lernenden lernen, wie ein solches Projekt ablaufen soll und wie man aus der realen Welt ein oder mehrere Objekte analysiert und für die objektorientierte Implementierung in Diagrammen abbildet. Anhand dieser Schritte werden IT-Projekte im optimalen Fall durchgeführt.

## Aufgabenstellung

Die Aufgabenstellung ist wie in der Einleitung beschrieben ein Projekt nach dem entsprechenden Vorgehen abzuwickeln. Das Projekt wird allerdings nicht alleine gemacht sondern in Gruppen mit bis zu fünf Personen.

Der wesentliche Teil der Projektarbeit handelt von der Umsetzung einer Modelleisenbahn der realen Welt in ein Programm mit welchem man einen Zug auf einer Strecke im Kreis fahren lassen kann. Das Programm soll folgende Klassen beinhalten:

* Gerade Schiene
* Gebogene Schiene
* Weiche
* Dreiwegweiche
* Kreuzung
* Einfache Kreuzungsweiche
* Trafo
* Lok
* Güterwagen
* Postwagen

Die Abgaben des Projekts beinhalten folgende Elemente:

* Dokumentation des Vorgehen nach OO
* Arbeitsjournal
* JAR-File der Implementation

## Aufgabenplanung

Für das Projekt fallen einige Tätigkeiten an, welche erledigt werden müssen. Folgend sind die Elemente, welche mindestens vorhanden sein müssen aufgelistet:

* Pflichtenheft
* Use-Case-Diagramm
* Use-Case-Diagramm beschrieben
* Paketdiagramm
* Klassendiagramm
* Beschreibung Klassendiagramm
* Assoziations-Identifikation
* Assoziations-Identifikation beschrieben
* Generalisierungs-Identifikation
* Generalisierungs-Identifikation beschrieben
* Attribute vollständig beschrieben
* Sequenzdiagramm
* Implementation gemäss Klassen- und Sequenzdiagramm
* Testfälle JUnit implementiert
* JAVADoc
* Dokumentation des gesamten Projekts
* Reflexion des gesamten Projekts

Für die Projektarbeit sind folgende Daten in der Schule vorgesehen, geplant mit den Aufgabenbereichen die wir bis zum jeweiligen Datum beendet haben möchten.

* 20.11.2014
  + Projektbeginn
* 27.11.2014
  + Objektdiagramm
  + Beginn der Dokumentation
* 04.12.2014
  + Klassendiagramm
  + Use-Case-Diagramm
* 11.12.2014
  + Paketdiagramm
  + Beschreibung Use-Case-Diagramm
  + Klassendiagramm Teilbeschreibung
  + Assoziations-Identifikation
  + Assoziations-Identifikation beschrieben
* 18.12.2014
  + Generalisierungs-Identifikation
  + Generalisierungs-Identifikation beschrieben
  + Attribute vollständig beschrieben
  + Sequenzdiagramm
  + Grosser Teil der Dokumentation
* 08.01.2015
  + Dokumentation fertig
  + Arbeitsjournal komplett
  + Implementation komplett
  + JAVADoc komplett
  + Testfälle implementiert
  + Reflexion jedes Gruppenmitglieds

An jedem dieser Termine gibt es eine Sprintbesprechung bei welcher diskutiert wird, wie weit wir gekommen sind.

Für die Zuständigkeiten und weitere Informationen sehen sie bitte das Arbeitsjournal ein.

# Realisierung

Bei der Realisierung sind wir mit dem vorgegebenen Schema vorgegangen. Und zwar war dies zuerst die Analyse, dann das Design und schlussendlich die Implementation zu machen. Wir begonnen also mit der Analyse.

## Analyse - OOA

Im Analyseprozess ging es vorerst darum, sich mit dem Projektauftrag auseinander zu setzen um sich eine Vorstellung zu machen, wie der Analyseprozess durchgeführt werden soll. Wir nahmen also die Modelleisenbahn und analysierten die einzelnen Bestandteile, welche diese beinhalten soll. Diese sind im Kapitel Aufgabenstellung beschrieben.

Im ersten Schritt erstellten wir nur mal ein Objektdiagramm, welches noch keine Attribute beinhaltete. Diese sollten dann anschliessend folgen.

Der zweite Schritt war den Objekten ihre Eigenschaften, die sie aufweisen müssen, zu zuweisen. Dieser Prozess beinhaltete schon mehr Gedankenarbeit, denn welche Eigenschaften benötigen die Objekte wirklich für die endgültige Implementation? Der Punkt bei diesem Schritt ist, dass man noch nicht soweit vorausschauen soll. Ob schlussendlich mehr oder weniger Eigenschaften benötigt werden kann man künftig entscheiden und verbessern.

Wir haben ein Dokument erstellt, in welchem alle Objekte mit ihren Eigenschaften und wie sie funktionieren beschrieben sind. Ausserdem hat jedes Objekt auch ein Bild dazu bekommen. Der Analyseprozess war somit weitgehend abgeschlossen.

Die Dateien sind im Anhang zu finden.

## Design - OOD

In einem ersten Designschritt galt es das Use-Case-Diagramm zu erstellen. Dieses und das Objektdiagramm sollten dazu dienen, die Grundlagen für das Klassendiagramm herzugeben. Für das Klassendiagramm mussten wir uns erstmals überlegen, welche Klassen in das Klassendiagramm gehören. Schlussendlich mussten wir mehr Klassen implementieren, als es in der Realität gibt.

Ein erster Entwurf des Klassendiagramms sollte zuerst mal erstellt werden. Dies funktioniert eigentlich recht gut und wir besprachen meist in Zweiergruppen, wie das Klassendiagramm zu verbessern ist. Der erste Entwurf hatte noch keine Assoziationen und Methoden. Die Assoziationen sollten in einem zweiten Schritt erstellt werden. Was wir uns überlegen mussten, war welche Klassen welche Beziehungen zu welchen Klassen hatten.

Nachdem wir die zweite Version des Klassendiagramms erstellt hatten, war der nächste Schritt, beziehungsweise dank der grossen Anzahl an Personen in unserer Gruppe während der Erstellungszeit des Klassendiagramms, das Sequenzdiagramm zu erstellen. Das Sequenzdiagramm sollte ein Beispiel darstellen, welche Methoden unsere Objekte benutzen und aufzeigen, wie schlussendlich ein Teil unserer Main-Methode aussehen könnte. Das Sequenzdiagramm sollte später dann bei der Implementierung als Unterstützung dienen. Es dient nämlich der Implementation der Methoden ins Klassendiagramm. So konnte das Klassendiagramm weitestgehend zu Ende gebracht werden.

Nebenbei erstellten wir auch noch das Paketdiagramm, welches auch notwendig für die Vollständigkeit unseres Projekts und unseres objektorientierten Designs war.

Die Dateien und Diagramme, welche hier beschrieben wurden, sind im Anhang zu finden.

## Implementation - OOP

Bei der Implementation ging es vorerst darum, alle Klassen, die wir im Klassendiagramm beschrieben hatten, in Code umzusetzen. Dieser Vorgang war der einfachste der Implementation. Es bestand somit noch keine wirkliche Logik. Diese sollte später folgen.

Die Methoden, die wir im Klassendiagramm beschrieben hatten, waren der zweite Schritt der Implementation. Diese galt es nun in Code umzusetzen. Das heisst beispielsweise es mussten Vorgänge wie die Möglichkeit, Schienen aneinanderzuhängen, Weichen umzustellen, Wagons aneinanderzuhängen und viel mehr möglich sein. Die Logik konnte vorerst nur durch die Implementation noch nicht getestet werden. Als nächstes galt es nämlich zuerst die Main-Methode zu erstellen. Das Programm kann nun durch unsere Main-Methode den Zug über die Schienen fünf Runden fahren lassen und schlussendlich wieder beim Bahnhof anhalten. Er kann die innere Runde nehmen, aber auch die äussere. Dass dies sichtbar ist, wurde dies durch Textausgaben sichergestellt.

Nebst der JAVADOC-Dokumentation, welche wir schlussendlich noch tätigen mussten, galt es auch noch für die Methoden der Klassen JUnit-Tests zu schreiben. Diese Tests sind vorhanden und testen, ob die Methoden den richtigen Rückgabewert haben und somit funktionieren oder eben nicht funktionieren. Dies entwickelt sich für uns zu einem starken Vorteil, denn wenn etwas nicht so läuft wie es sollte bemerken wir dies schnell und können so die Verbesserungen an unseren Methoden schnell durchführen.

## Dokumentation

Die Dokumentation ist einer der wichtigsten Bestandteile eines Programmes. Sie dient der Wartbarkeit und der Nachvollziehbarkeit einer anderen Person des gesamten Projekts und des Programms. Diese Dokumentation wurde von uns während des gesamten Projekts aktualisiert. Den grössten Teil allerdings konnten wir erst gegen den Schluss des Projekts erledigen, was selbstverständlich ist, denn das Programm ist ja auch erst eher gegen Ende des Projekts fertig.

Wir haben uns in während dieser Dokumentation an die Richtlininen von JAVADoc gehalten und auch in ein JAVADoc exportiert. Den Ordner dazu finden sie hier. ***TODO***.

# Programmbeschreibung

Unser Programm wurde in drei Pakete gegliedert und zwar in **modelrailway**, **vehicles** und **railsystem**. Es gab dann im Laufe der Implementation noch ein zusätzliches Paket **helper** welches Klassen beinhaltet die während der Implementation noch dazu gekommen sind.

Hier werden nun die Pakete kurz erklärt.

## Modelrailway

Das ist das Hauptpaket, es beinhaltet auch die anderen drei Pakete. Neben diesen existieren zwei Klassen in diesem Paket und zwar **BaseObejct** und die **Main**.

## Vehicles

In diesem Paket finden sich alle Klassen, welche mit einem Zug in Beziehung stehen, sprich **Locomotive**, **Train**, **Wagon**.

## Railsystem

In diesem Paket befinden sich alle Klassen welche ein Schienennetz beschreiben, also Kurven, Geraden, Weichen und diverse Spezialschiene.

# Testkonzept

Unsere Tests haben wir in JUNIT geschrieben. Dabei werden alle Funktionen von allen Klassen geprüft insofern diese keine getter oder setter sind. Wir haben sowiso fast keine Setter implementiert da wir sagten, wenn man eine Schiene erstellt hat, dann ist sie erstellt und kann nicht mehr verändert werden. Das bedeutet die meisten Attribute werden bereits im Konstruktor gesetzt und können danach nur noch abgefragt werden.

Beim Testen von Methoden ist es wichtig ein breites Spektrum an Szenarien zu überprüfen, dabei gibt es üblicherweise 5 Tests welche die korrekte Funktion einer Methode prüfen.

* Unter dem Minimum, das heisst ein Tank kann nicht weniger beinhalten als 0 Liter und das muss geprüft werden.
* Genau das Minimum, das heisst ein Tank soll wenn er leer ist genau 0 Liter beinhalten.
* Im Wertebereich, das heisst der Tank beinhaltet etwas zwischen 0 Litern und der maximalen Füllmenge.
* Genau das Maximum, der Tank soll genau soviel Liter beinhalten wie er fassen kann.
* Über dem Maximum, das bedeutet der Tank soll wenn er voll ist nicht mehr beinhalten als sein Füllvolumen ist.

Die Tests welche über dem Minimum und unter dem Maximum sind können sind nötig wenn durch eine Usereingabe mehr hinzugefügt oder weggenommen wird als möglich ist. Denn wenn der Maximal und Minimalwert über oder unterschritten wird, kann es zu schwerwiegenden Fehlern kommen.

**Siehe JUnit.**

# Fazit

## Was haben wir erreicht?

Wir haben eine Simulation einer Modelleisenbahn auf die Beine gestellt und diese mit   
JUnit-Tests versehen um einfacher die verschiedenen Klassen und deren Funktionalitäten zu prüfen.

Ausserdem haben wir zum Programm eine JAVADOC-Dokumentation erstellt und diverse Diagramme gezeichnet damit wir einfacher implementieren konnten.

## Was konnten wir nicht fertigstellen?

Nichts.

## Welche Probleme hatten wir?

Wir hatten Probleme bei der Zusammenführung unserer Dokumente, dies gelang mit der Zeit ein wenig besser, aber noch nicht optimal. Ebenfalls fiel es uns zum Anfang schwer das Klassendiagramm zu verstehen und zu zeichnen.

## Was muss nächstes Mal besser gelingen?

Die Zusammenarbeit muss sich noch verbessern und jeder sollte sich mit dem Programm auskennen, welches wir zur Dateiverwaltung und Dateiversionierung benutzt haben.

## Was haben wir gelernt?

Wir haben gelernt wie ein objektorientiertes Projekt abläuft und ausgeführt wird. Leider mussten wir auch feststellen, dass in einer Gruppe nicht immer alles optimal läuft, doch glücklicherweise wissen wir nun alle wie ein solches Projekt in etwa abgewickelt wird.

## Fazit David

Vorerst, als wir die Information zugeteilt bekamen, dass das ganze Modul 326 eine Projektarbeit ist, war ich alles andere als begeistert. Jedoch hat sich meine Begeisterung ein bisschen gesteigert bis zum Ende Projekt. Im Nachhinein kann ich sagen ist es doch eine gute Idee den Lernstoff so zu übermitteln. Man konnte einiges lernen, gerade beim Ablauf und bei der Abwicklung eines solchen Projekts. In dieser Art hatte ich dies noch nie. Ich lernte auch mehr über das objektorientierte Analysieren und die weiteren Schritte die man durchlaufen muss.

Probleme gab es bei mir bei der Klassendiagrammerstellung und auch mit der Implementation anhand eines objektorientierten Designs habe ich noch Mühe. Bei beiden Tätigkeiten fehlen mir Praxis und manchmal auch Verständnis, was aber durchaus durch Nachfragen bei Teamkollegen verbessert werden konnte. Ich werde mich aber künftig noch mehr mit dem Thema beschäftigen müssen.

Negative Aspekte, die das Projekt ein bisschen geprägt hatten, waren beispielsweise die Zeit. Wir mussten definitiv zu viel Freizeit dafür investieren. Man sollte in der Schule genügend Zeit für solche Projekte bekommen und tun wir dies nicht, so sollten die Projekte kleiner werden.

Zudem war der Lerneffekt in allen Bereichen, die dieses Projekt umschloss nicht gerade allzu gross, weil wir fünf Personen sind die daran gearbeitet hatten.

Wir oder zumindest ich wurde ein bisschen ins kalte Wasser geschmissen mit dem Projekt, kann aber nun sagen, dass ich aus diesem Projekt einiges mitnehmen kann für die Zukunft, falls es in dieser Form in meinem Arbeitsleben wieder auftreten wird.

## Fazit Silvan

Anfangs war ich mittelmässig von dieser Projektarbeit überzeugt, doch in den ersten Phasen schien es sehr gut zu funktionieren. Insgesamt kann ich sagen, dass es gut ist etwas Praktisches zu tun, jedoch wird es sehr schnell chaotisch wenn 5 Personen miteinander an den gleichen Dokumenten arbeiten.

Ich fand es ein wenig Schade, dass nicht alle beim jedem Diagramm dabei sein konnten, denn die Zeit reichte dafür nicht und somit hat nicht jeder praktische Erfahrung in einem Klassendiagramm oder Ähnlichem.

Ich kam vorerst nicht sehr gut mit dem Projekt klar, konnte jedoch vieles mitnehmen und werde dies im Betrieb in den nächsten Wochen auch einsetzen können.

## Fazit Simon

Ich freute mich eigentlich ziemlich auf das Projekt, denn es war neben der Prüfung im Herbst, das erste Mal, wo wir selbst etwas implementieren konnten. Doch diese Freude legte sich wieder als ich den Umfang und das Niveau dieser Projektarbeit erkannte. Dieses fand ich nämlich zu hoch, für mich persönlich war es kein allzu grosses Problem, aber ich denke für Leute die noch nie Objektorientiert programmiert haben war der Unterschied zwischen den abgeschriebenen Implementationen anfangs dieses Semesters und der Implementation die hier notwendig war doch horrend.

Meiner Meinung nach hatten wir vor allem bezüglich dem Programmieren schlicht zu wenig Praxis um ein Projekt von dieser Komplexität in gerade mal 8 Schulstunden zu implementieren. Natürlich ist es die Idee, dass man diese Aufgabe aufteilt, aber wenn nur ein oder zwei Leute einer Gruppe noch nie ein eigenes Java Programm erstellt haben, dann fehlen bereits 30% - 40% der Arbeitskraft.

Ich finde allerdings bei der Erstellung der Diagramme, also vor allem im OOA und OOD hatten wir dank der Lektionen schon ein wenig Erfahrung und konnten diese zwei Teile doch sehr gut absolvieren.

Ich persönlich nehme aus diesem Projekt vieles mit. Ich habe gemerkt wie wichtig ein strukturierter Aufbau einer Klasse ist, um diese zu ergänzen und zu verändern. Bei einer sauberen Struktur fallen einem solche Dinge gleich viel einfacher. Ich konnte auch einiges in Sachen Zusammenarbeit und Aufgabenverteilung lernen.

## Fazit Marco

Das Gruppenprojekt war interessant und lehrreich. Ich hab noch nie zuvor so ein grosses Projekt in einem 5-er Team realisiert. Es war schwer, den Überblick über das ganze Projekt zu behalten, beziehungsweise schauen, dass alle etwas zu tun habe und niemand das gleiche macht.

Bei den Diagrammen habe ich gelernt, dass man nicht allzu genau planen muss. Manchmal denkt man schon viel zu weit und verliert sich im Detail.

Gut gelungen ist die Kommunikation zwischen uns. Wir hatten nie Konflikte und lösten Probleme schnell.

Jedoch fand ich wir hatten viel zu wenig Zeit für das ganze Projekt, da einige von uns keinen Plan von Java haben und dadurch das ganze Projekt in die Länge gezogen wurde. Wir musste fast die ganze Implementation zu Hause erledigen.

## Fazit Remo

Dank dem Projekt habe ich angefangen GitHub zu benutzen. Dies finde ich den grössten Gewinn denn ich durch das Projekt erlangt habe. Im Bereich des Programmierens habe ich leider nichts Neues gelernt. Im Bereich der Dokumentation habe ich meine Fertigkeiten bezüglich Use Case, Sequenzdiagramm verfestigt und verbessert. Generell hat mir das Projekt mehr Erfahrung gegeben. Ich fand, dass es zu grosse Gruppen waren. Die Arbeitsaufteilung ist schwer unter so vielen Leuten in solch einem kleinen Projekt. Generell fand ich das Projekt auch zu fern ab der Realität der Arbeitswelt.

# Schlusswort

# Anhang

|  |  |
| --- | --- |
| **Beschreibung** | **Dokument** |
| Kapitel Analyse,Design,Implementation!!!!!!!!!! |  |
|  |  |