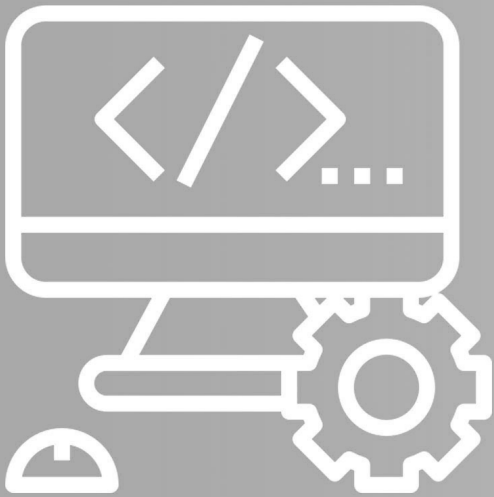


DOKUMENTATION

MODUL 318



Tamila Govduchanova
Maihofstrasse 87
6006 Luzern

S-INF18s
Michael Zurmühle

24.04.2020

INHALTSVERZEICHNIS

1. EINLEITUNG.....	4
1.1. MODULIDENTIFIKATION	4
1.1.1. HANDLUNGSZIELE & HANDLUNGSNOTWENDIGE KENNTNISSE	4
2. MANAGEMENT SUMMARY	5
3. ZWECK DES DOKUMENTS	5
4. PROGRAMMIERRICHTLINIEN FÜR C, C++, C#.....	5
4.1. WARTBARKEIT, LESBARKEIT, ÄSTHETIK	5
4.2. SOFTWARE-AKTUALITÄT	6
4.3. SPRACHE	6
4.4. DOKUMENTATION	6
4.5. NOTATIONEN	6
4.6. ZEILENLÄNGE, SONDERZEICHEN UND UMLAUTE	7
4.7. EINRÜCKUNGEN (INDENTS)	7
4.8. VERWENDUNG DER GESCHWEIFTEN KLAMMERN	8
4.9. DEKLARATIONEN UND ZUWEISUNGSZEICHEN IN DERSELBEN SPALTE	8
4.10. ENUMERATIONEN	9
5. WELCHE FUNKTIONEN WURDEN UMGESETZT?.....	10
A001.....	10
A002.....	10
A003.....	10
A004.....	10
A005.....	10
EIGENE FUNKTION	10
EIGENE FUNKTION	10
6. VISUALISIERUNG & DIAGRAMME.....	11
6.1. MOCKUP GUI.....	11
6.2. USECASE-DIAGRAMM.....	12
6.3. AKTIVITÄTSDIAGRAMM	15
6.4. UML-DIAGRAMM	16

7. <u>WARUM GEHEN IT-PROJEKTE SCHIEF?</u>	17
8. <u>TESTFÄLLE</u>	18
„EIN TESTFALL (ENGL. TEST CASE) BESCHREIBT EINEN ELEMENTAREN, FUNKTIONALEN SOFTWARETEST, DER DER ÜBERPRÜFUNG EINER Z. B. IN EINER SPEZIFIKATION ZUGESICHERTEN EIGENSCHAFT EINES TESTOBJEKTES DIENT. TESTFÄLLE WERDEN UNTER ANWENDUNG VON TESTMETHODEN UND GEEIGNETEN SOFTWARE-HILFSMITTELN ERSTELLT.“ (WIKIPEDIA, TESTFALL)	18
8.1. TEST „STATIONSSUCHE“	18
8.2. TEST „AKTUELLE VERBINDUNGEN SUCHE“	18
8.3. TEST „VERBINDUNGEN NACH BESTIMMTER STATION SUCHE“	18
8.4. TEST „VERBINDUNGEN NACH BESTIMMTER ZEIT SUCHE“	19
8.5. TEST „RESET BUTTON“	19
8.6. TEST „ABFAHRT & ANKUNFT TAUSCHEN“	19
9. <u>TESTPROTOKOLL</u>	20
9.1. TEST „STATIONSSUCHE“	20
9.2. TEST „AKTUELLE VERBINDUNGEN SUCHE“	20
9.3. TEST „VERBINDUNGEN NACH BESTIMMTER STATION SUCHE“	21
9.4. TEST „VERBINDUNGEN NACH BESTIMMTER ZEIT SUCHE“	22
9.5. TEST „RESET BUTTON“	22
9.6. TEST „ABFAHRT & ANKUNFT TAUSCHEN“	23
10. <u>INSTALLATIONSANLEITUNG</u>	23
<u>ABBILDUNGSVERZEICHNIS</u>	24
<u>SELBSTSTÄNDIGKEITSERKLÄRUNG</u>	24

1. EINLEITUNG

1.1. Modulidentifikation

Modul 318 - Analysieren und objektbasiert programmieren mit Komponenten

Objekt: Mit einer Komponenten-Bibliothek implementiertes Programm

Lektionen: 40

Kompetenzfeld: Application Engineering

1.1.1. Handlungsziele & Handlungsnotwendige Kenntnisse

Handlungsziel	Handlungsnotwendige Kenntnisse	
1.	1.	Kennt eine Methode für die Analyse einer Problemstellung (z.B. Use Case) und kann diese auf einfache Aufgaben anwenden.
	2.	Kennt eine Methode für die Darstellung von Programmabläufen (z.B. Struktogramm, Aktivitätsdiagramm) und kann damit einfache Sachverhalte darstellen.
	3.	Kennt die Strukturelemente Iteration und Selektion und kann deren sinngerechte Anwendung aufzeigen.
2.	1.	Kennt allgemeine Regeln der SW-Ergonomie (z.B. DIN EN ISO 9241-110) und kann aufzeigen, wie diese für die Gestaltung einer Oberfläche anzuwenden sind.
	2.	Kennt die wichtigsten Komponenten einer graphischen Oberfläche und kann diese korrekt einsetzen.
3.	1.	Kennt eine Komponenten-basierte Programmierumgebung (z.B. .NET, Java/Swing) und deren typische Einsatzgebiete.
	2.	Kennt die unterschiedlichen Möglichkeiten für Selektion (einfach/mehrfach) und Iteration (vor-/nachprüfend) und kann diese mit einer Programmiersprache umsetzen.
	3.	Kennt Prozeduren und Funktionen und kann deren Aufbau (inkl. Parameterlisten) und Einsatz erklären.
	4.	Kennt den Mechanismus der Ereignisverarbeitung eines objektbasierten Systems und kann Ereignisse in Verarbeitungsroutinen auswerten.
4.	1.	Kennt die Bedeutung von Kommentaren bei Prozeduren und Funktionen und deren Beitrag zu besser wartbarem Programmcode.
5.	1.	Kennt ein Verfahren für den statischen Test (z.B. Codereview) von Programmcode und kann die Bedeutung für die Qualität der Software aufzeigen.
	2.	Kennt ein Verfahren (z.B. Grenzwertanalyse) zur Definition von Testfällen, um die Zuverlässigkeit von Algorithmen nachzuweisen.

Abbildung 1: Modul_318_Modulidentifikation und Handlungsnotwendige Kenntnisse.pdf, 20.04.2020

2. MANAGEMENT SUMMARY

Dieses Dokument habe ich während dem ganzen ÜK318 geschrieben und fortlaufend aktualisiert. Wir haben das Thema „objektorientiertes Programmieren“ behandelt. Zuerst haben wir ein paar Testübungen programmiert, die mir später fürs Projekt geholfen haben. Zusätzlich haben wir noch einige Inputs erhalten zum Thema Projekte, Design und Testing. Das Ziel bzw. unser Projekt ist es eine Applikation zu entwickeln, die anhand von einer Datenbank mit öffentlichen Fahrplandaten, Verbindungen raussucht und diese anzeigt. Zusätzlich konnten/mussten wir noch andere Features einbauen: nur Abfahrten oder Ankünfte anzeige, Stationen auslesen, etc.

3. ZWECK DES DOKUMENTS

Das Dokument dient zum besseren Verständnis der ganzen Projektarbeit und damit der Instruktor oder jemand anderes nachvollziehen kann, was es mit der ganzen Applikation auf sich hat. In diesem Dokument sind verschiedene Dinge beschrieben: Die umgesetzten Anforderungen, Funktionen, Programmierrichtlinien, Diagramme, Testfälle, etc.

4. PROGRAMMIERRICHTLINIEN FÜR C, C++, C#

Auszüge aus: <https://www.team-electronics.com/Download/D0577-03.pdf>

Die folgenden Zeilen beschreiben die grundsätzlichen Standards, welche bei der Software-Erstellung und Wartung berücksichtigt werden sollten. Diese Standards beziehen sich besonders auf die Programmierung in den Sprachen C, C++ und C# - die meisten Regeln gelten aber auch für alle anderen Programmiersprachen.

4.1. Wartbarkeit, Lesbarkeit, Ästhetik

Bei der Erstellung von Software sollten immer folgende Faktoren in Erinnerung gerufen werden:

- Kann ich diesen Source-Code auch in einem Jahr noch verstehen?
- Ist meine Software ordentlich strukturiert/modularisiert oder gibt's haufenweise dubiose Querverbindungen zwischen den Modulen?
- Was passiert, wenn jemand anderer meinen Code betrachtet ("Code-Review") oder Änderungen vornehmen muss – wird er sich zurechtfinden?
- Erscheint meine Software aus "einem Guss" (konsequentes Anwenden von Kommentierungen, Einrückungen, Variablennamen...)?

4.2. Software-Aktualität

Wird eine Software erstellt oder geändert, so muss diese wie auch alle Dokumente und andere Unterlagen auf dem Server gespeichert werden. Nach einer eventuellen Software-Änderung ausser Hause und muss diese Software sofort bei der Rückkehr (und nicht erst Stunden später oder nie) auf den gemeinsamen Server .

4.3. Sprache

Grundsätzlich orientieren sich viele Variablennamen an der englischen Sprache und die Kommentare sind in Deutsch verfasst, sofern nicht Kundenanforderungen dies verhindern.

Wie in der Programmierung üblich, ergibt sich oft ein Mix aus deutsch/englisch, was aber im Rahmen der natürlichen Empfindung zulässig ist. Es gelten allgemein die Regeln der neuen deutschen Rechtschreibung.

4.4. Dokumentation

Lästig, aber notwendig: Inbetriebnahmeanleitungen, Beschreibung von Funktionsbibliotheken, Bedienungsanleitungen, Berechnungs-Dokumentationen (Übersetzungsverhältnisse...) usw. Verweise auf diese Dokumente finden sich im jeweiligen Info-File.

4.5. Notationen

Grundsätzlich existieren diese Notationen (Schreibstile) von Bezeichnern:

Notation	Bemerkung	Beispiel
Pascal	Der erste Buchstabe jeweils gross.	MyIdentifier
Kamel	Am Anfang klein, danach jeweils der erste Buchstabe gross.	myIdentifier
Ungarisch	Wie Pascal, aber mit einem vorangestellten kleinengeschriebenen Typ-Präfix	nMyIdentifier
Gross	Alles groß, Worttrennung durch Unterstrich	MY_IDENTIFIER
Klein	Alles klein, Worttrennung durch Unterstrich	my_identifier

Je nach Bezeichnertyp (Methode, Konstante, Variable...) werden unterschiedliche Notationen angewendet. Die Notation "Klein" sollte nur noch bei Standard C-Programmen verwendet werden und keinesfalls in C++ oder C# - Code Einzug finden.

4.6. Zeilenlänge, Sonderzeichen und Umlaute

Die Zeilenlänge (Breite des Codes) sollte maximal 100 Zeichen betragen. Üblich und historisch bedingt (DOS- Fenster, Nadeldrucker...) sind hingegen 80 Zeichen, was auch im Zeitalter hochauflösender Bildschirme eine sinnvolle Begrenzung darstellt. Sonderzeichen und Umlaute wie "ß", "@", "ä" oder "\$" sind natürlich nur innerhalb von Strings oder Kommentaren zulässig und auch erlaubt.

Ausnahme: Info-Files und "alte C-Files" sollten keines dieser Zeichen verwenden, da diese oft mit alten Editoren bearbeitet werden, welche ein anderes Sonderzeichen-Format besitzen.

4.7. Einrückungen (Indents)

Einrückungen erfolgen in Schritten von 2 Leerzeichen, wobei diese nicht als Tabulatorzeichen definiert sein sollten. Dies kann in den meisten Entwicklungsumgebungen mit Einstellungen wie "Insert spaces for tabs" vorgenommen werden.

Beispiel:

```
namespace Test
{
    public class MyClass
    {
        private int Sum (int nA, int nB)
        {
            return (nA + nB);
        }
    }
}
```

und niemals so:

```
namespace Test
{
public class MyClass
{
    private int Sum (int nA, int nB)
    {
        return (nA + nB);
    }

} }
```

4.8. Verwendung der geschweiften Klammern

gut	schlecht
<pre>if (nX > 0) { for (int i = 0; i < 10; i++) { adValueX[i]++; adValueY[i]--; } }</pre>	<pre>if (nX > 0) { for (int i = 0; i < 10; i++) { adValueX[i]++; adValueY[i]--; } }</pre>

Code-Blöcke, welche nur aus einer einzigen Anweisung bestehen, dürfen auf die geschweiften Klammern verzichten:

Richtig	Ebenso richtig
<pre>if (nX > 0) dValue = 1;</pre>	<pre>if (nX > 0) { dValue = 1; }</pre>

4.9. Deklarationen und Zuweisungszeichen in derselben Spalte

Typdeklarationen, Klasseninstanzen, Zuweisungszeichen usw. sollten für zusammenhängende Code-Abschnitte in derselben Spalte stehen.

gut	schlecht
<pre>double dTestVar1; float fTestVar2; long nValue; int nValueLessOne; bool bEnable; dTestVar1 = 1.89; fTestVar2 = -17.89; nValue = (long)dTestVar1; nValueLessOne = (int) nValue - 1; bEnable. = true;</pre>	<pre>double dTestVar1; float fTestVar2; long nValue; int nValueLessOne; bool bEnable; dTestVar1 = 1.89; fTestVar2 = -17.89; nValue = (long)dTestVar1; nValueLessOne = nValue - 1; bEnable = true;</pre>

4.10. Enumerationen

Enumerationen erhalten das Präfix **E**. Die einzelnen Enumerationswerte werden je nach Typ entweder in den Notationen "Groß" oder "Pascal" definiert. "Groß" wird verwendet, wenn die Enumerationswerte den Charakter von Konstanten haben, ansonsten kann die Notation Pascal benutzt werden.

Instanzen von Enumerationen erhalten das Präfix **e**.

Beispiel:

Enumeration mit Konstanten	Wertunabhängige Enumeration
<pre>public enum EDigitalInputs : ushort { START = 1<<0, n STOP = 1<<1, nNOTAUS = 1<<2, } EDigitalInputs eDiBoard1; if ((eDiBoard1 & EDigitalInputs.START) != 0) { // ... }</pre>	<pre>public enum EColor { Red, Blue, Green, } EColor eColor; eColor = EColor.Red;</pre>

5. WELCHE FUNKTIONEN WURDEN UMGESETZT?

A001 Als ÖV-Benutzer möchte ich Start- und Endstation mittels Textsuche suchen können, damit ich nicht alle Stationsnamen auswendig lernen muss.

A002 Als ÖV-Benutzer möchte ich die aktuellen, d.h. mindestens die nächsten vier bis fünf Verbindungen zwischen den beiden gefundenen und ausgewählten Stationen sehen, damit ich weiss wann ich zur Station muss, um den für mich idealen Anschluss zu erwischen.

A003 Als ÖV-Benutzer möchte ich sehen, welche Verbindungen ab einer bestimmten Station vorhanden sind, damit ich bei mir zuhause eine Art Abfahrtstafel haben kann.

A004 Als ÖV-Benutzer möchte ich, dass schon während meiner Eingabe erste Suchresultate erscheinen, damit ich effizienter nach Stationen suchen kann.

A005 Als ÖV-Benutzer möchte ich nicht nur aktuelle Verbindungen suchen können, sondern auch solche zu einem beliebigen anderen Zeitpunkt, damit ich zukünftige Reisen planen kann.

Eigene Funktion: Als ÖV-Benutzer möchte ich einen „Reset“-Button haben.

Eigene Funktion: Als ÖV-Benutzer möchte ich Start- & Endstation tauschen können.

6. VISUALISIERUNG & DIAGRAMME

6.1. Mockup Gui

„Ein Vorführmodell ist ein komplettes Produkt oder eine Attrappe, die genutzt wird, um Design und/oder Funktionen eines geplanten oder bereits eingeführten Produktes zu demonstrieren. Es ist oft ein maßstäbliches Modell bzw. eine Nachbildung zu Präsentationszwecken.« (Wikipedia, Vorführmodell)

 = schlussendlich umgesetzt


 = zusätzlich im Verlauf der Arbeit hinzugefügt

TransportApp

Abfahrt

Ort...

/ /




↔

Ankunft

Ort...

/ /



Verbindungen suchen


nur Ankünfte suchen

nur Abfahrten suchen

RESET

Karte anzeigen

Karte anzeigen

 Suchergebnisse weiterleiten

Abfahrtszeit	Abfahrtsort	Ankunftszeit	Ankunftsort

Abbildung 2: Mockup Gui erstellt mit "Balsamiq", 21.04.2020

6.2. UseCase-Diagramm

„Ein Anwendungsfalldiagramm, auch Nutzfalldiagramm, ist eine der 14 Diagrammarten der Unified Modeling Language, einer Sprache für die Modellierung der Strukturen und des Verhaltens von Software- und anderen Systemen. Es stellt Anwendungsfälle und Akteure mit ihren jeweiligen Abhängigkeiten und Beziehungen dar.“ (Wikipedia, Anwendungsfalldiagramm)

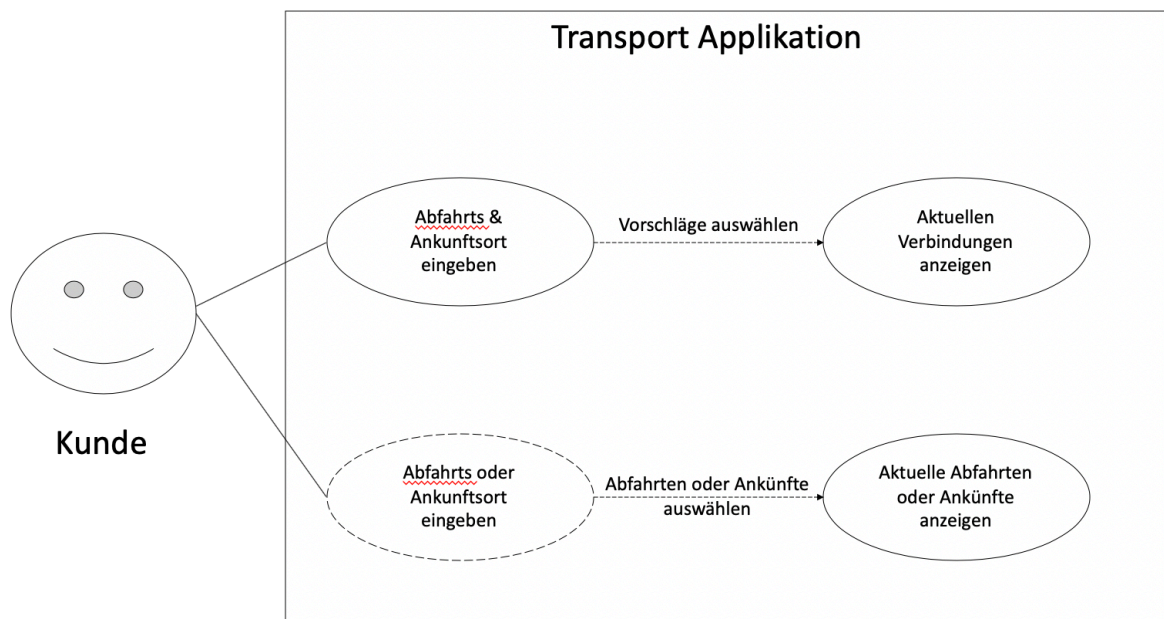


Abbildung 3: UseCase-Diagramm, 21.04.2020

Use Case 1	Abfahrts & Ankunftsort eingeben
Beschreibung	Der Benutzer gibt seinen Abfahrts- und Ankunftsort in das vorgegebene Feld ein.
Akteur(e)	ÖV-Benutzer
Auslöser	Der ÖV-Benutzer möchte von A nach B reisen.
Vorbedingungen	Der ÖV-Benutzer muss Abfahrts- & Ankunftsort kennen.
Ablauf	<ol style="list-style-type: none"> 1. Programm öffnen 2. Abfahrtsort eingeben (Vorschläge auswählen) 3. Ankunftsort eingeben (Vorschläge auswählen) 4. «Verbindungen suchen» drücken
Alternativer Ablauf #1	Kein
Ergebnis	Verbindungen werden angezeigt.

Use Case 2	Aktuelle Verbindungen anzeigen
Beschreibung	Dem Benutzer werden alle Verbindungen angezeigt, die zwischen A und B verkehren.
Akteur(e)	ÖV-Benutzer
Auslöser	Der ÖV-Benutzer möchte seine nächsten Verbindungen sehen.
Vorbedingungen	Der ÖV-Benutzer muss Abfahrts- & Ankunftsort kennen und den «Verbindungen suchen» -Button gedrückt haben.
Ablauf	<ol style="list-style-type: none"> 1. Use Case Fall «Abfahrts & Ankunftsort eingeben» 2. Verbindungen werden angezeigt.
Alternativer Ablauf #1	Kein
Ergebnis	Verbindungen werden angezeigt.

Use Case 3	Abfahrts- oder Ankunftsort eingeben
Beschreibung	Der Benutzer gibt seinen Abfahrts- oder Ankunftsort in das vorgegebene Feld ein.
Akteur(e)	ÖV-Benutzer
Auslöser	Der ÖV-Benutzer möchte alle Abfahrten oder Ankünfte sehen.
Vorbedingungen	Der ÖV-Benutzer muss Abfahrts- oder Ankunftsort kennen.
Ablauf	1. Programm öffnen 2. Abfahrts- oder Ankunftsort eingeben (Vorschläge auswählen) 3. «nur Ankünfte suchen» oder «nur Abfahrten suchen» drücken
Alternativer Ablauf #1	Kein
Ergebnis	Abfahrten oder Ankünfte werden angezeigt.

Use Case 4	Aktuelle Abfahrten oder Ankünfte anzeigen
Beschreibung	Dem Benutzer werden alle Abfahrten oder Ankünfte angezeigt, die von/nach A und B verkehren.
Akteur(e)	ÖV-Benutzer
Auslöser	Der ÖV-Benutzer möchte seine nächsten Abfahrten/Ankünfte.
Vorbedingungen	Der ÖV-Benutzer muss Abfahrts- oder Ankunftsort kennen und den entsprechenden Button gedrückt haben.
Ablauf	1. Use Case Fall «Abfahrts- oder Ankunftsort eingeben» 2. Abfahrten/Ankünfte werden angezeigt.
Alternativer Ablauf #1	Kein
Ergebnis	Abfahrten oder Ankünfte werden angezeigt.

6.3. Aktivitätsdiagramm

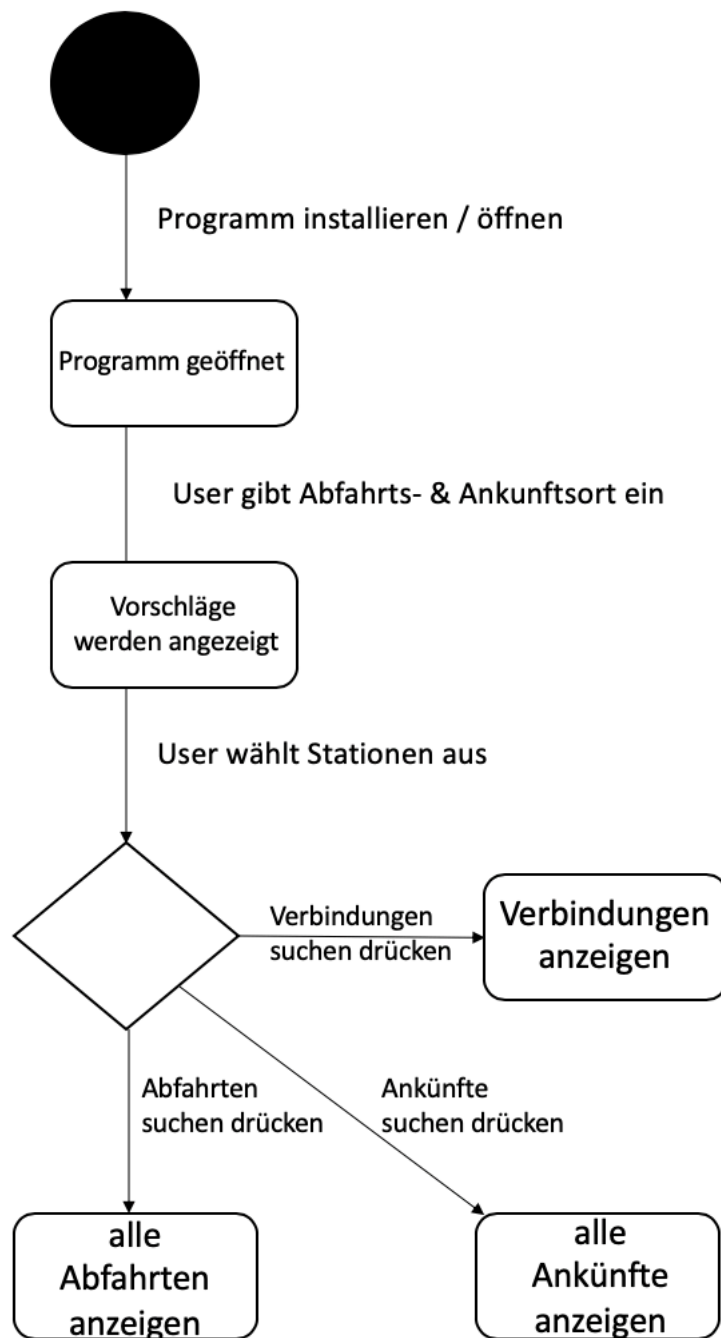


Abbildung 4: Aktivitätsdiagramm, 21.04.2020

„Ein Aktivitätsdiagramm ist ein Verhaltensdiagramm der Unified Modeling Language, einer Modellierungssprache für Software und andere Systeme, und stellt die Vernetzung von elementaren Aktionen und deren Verbindungen mit Kontroll- und Datenflüssen grafisch dar.“ (Wikipedia, Aktivitätsdiagramm)

6.4. UML-Diagramm

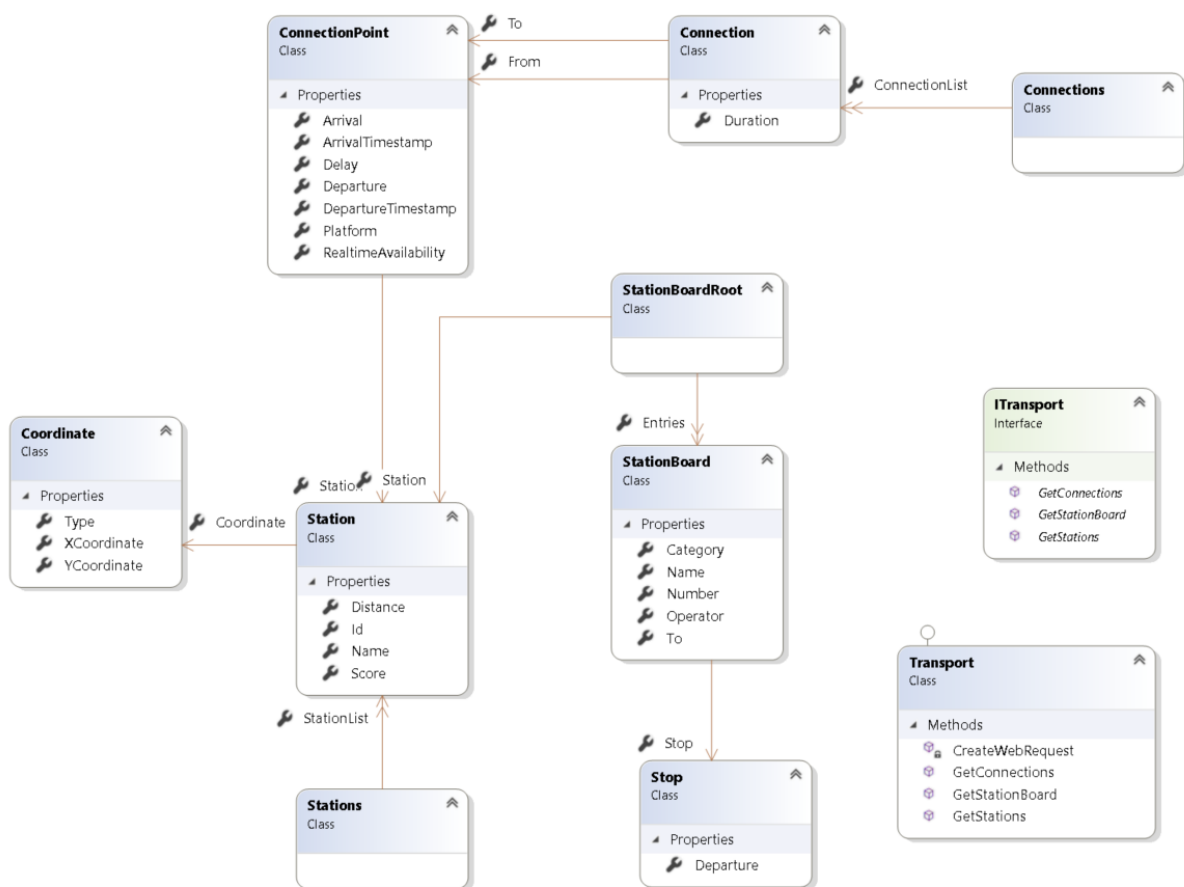


Abbildung 5: UML-Diagramm, 21.04.2020

„Die Unified Modeling Language, kurz UML, ist eine grafische Modellierungssprache zur Spezifikation, Konstruktion, Dokumentation und Visualisierung von Software-Teilen und anderen Systemen. Sie wird von der Object Management Group entwickelt und ist sowohl von ihr als auch von der ISO genormt.“ (Wikipedia, Unified Modeling Language)

7. WARUM GEHEN IT-PROJEKTE SCHIEF?

Problem	Lösung
fehlende Erfahrung in der Leitung	verstärkte Zusammenarbeit, flache "Rangordnung"
zu wenig Planung	mehr Zeit ins Planen investieren (verhindert Scheitern aufgrund Kosten, etc.)
nicht genug intensiv getestet	mehr Tests machen
wenig Engagement	kann super geplant sein, aber bringt nichts wenn kein Engagement gezeigt wird
schlechte Kommunikation (Hierarchie)	bei schlechter Organisation funktioniert auch die Kommunikation nach oben/unten nicht
kein klares Ziel	Überlegen, was und wer man mit der Software erreichen möchte
unklare Anforderungen	von Anfang an definieren

8. TESTFÄLLE

„Ein Testfall (engl. Test case) beschreibt einen elementaren, funktionalen Softwaretest, der der Überprüfung einer z. B. in einer Spezifikation zugesicherten Eigenschaft eines Testobjektes dient. Testfälle werden unter Anwendung von Testmethoden und geeigneten Software-Hilfsmitteln erstellt.“ (Wikipedia, Testfall)

8.1. Test „Stationssuche“

Der User möchte mit dem Textfeld Abfahrts- & Ankunftsort suchen.

Schritt	Aktivität	erwartetes Resultat
1	Abfahrtsort „Luzern, Bahnhof“ eingeben	„Luzern, Bahnhof“ wird vom Autocomplete erkannt und eingegeben.
2	Ankunftsort „Luzern, Maihof“ eingeben	„Luzern, Maihof“ wird vom Autocomplete erkannt und eingegeben.

8.2. Test „aktuelle Verbindungen suchen“

Der User möchte die aktuellen Verbindungen zwischen Abfahrts- & Ankunftsort sehen.

Schritt	Aktivität	erwartetes Resultat
1	Abfahrtsort „Luzern, Bahnhof“ eingeben	„Luzern, Bahnhof“ wird vom Autocomplete erkannt und eingegeben.
2	Ankunftsort „Luzern, Maihof“ eingeben	„Luzern, Maihof“ wird vom Autocomplete erkannt und eingegeben.
3	alle Verbindungen suchen drücken	Es werden die nächsten 4-5 Verbindungen zwischen „Luzern, Bahnhof“ und „Luzern, Maihof“ angezeigt

8.3. Test „Verbindungen nach bestimmter Station suchen“

Der User möchte die Verbindungen ab Abfahrtsort angezeigt bekommen.

Schritt	Aktivität	erwartetes Resultat
1	Abfahrtsort „Luzern, Bahnhof“ eingeben	„Luzern, Bahnhof“ wird vom Autocomplete erkannt und eingegeben.
2	„alle Abfahrten anzeigen“ drücken	alle aktuellen Verbindungen ab „Luzern, Bahnhof“ werden angezeigt.

Der User möchte die Verbindungen an Ankunftsort angezeigt bekommen.

Schritt	Aktivität	erwartetes Resultat
1	Ankunftsort „Luzern, Maihof“ eingeben	„Luzern, Maihof“ wird vom Autocomplete erkannt und eingegeben.
2	„alle Ankünfte anzeigen“ drücken	alle aktuellen Verbindungen an „Luzern, Maihof“ werden angezeigt.

8.4. Test „Verbindungen nach bestimmter Zeit suchen“

Der User möchte die Verbindungen zwischen Abfahrts- & Ankunftsort nach Zeit suchen.

Schritt	Aktivität	erwartetes Resultat
1	Abfahrtsort „Luzern, Bahnhof“ eingeben	„Luzern, Bahnhof“ wird vom Autocomplete erkannt und eingegeben.
2	Ankunftsort „Luzern, Maihof“ eingeben	„Luzern, Maihof“ wird vom Autocomplete erkannt und eingegeben.
3	Abfahrtszeit eingeben	Abfahrtszeit wurde im dafür vorgesehenen Feld eingegeben.
4	„alle Verbindungen suchen“ drücken	Es werden die aktuellen Verbindungen zwischen „Luzern, Bahnhof“ und „Luzern, Maihof“ angezeigt.

8.5. Test „Reset Button“

Der User möchte seine Ortseingaben nicht manuell löschen müssen, er fordert einen Button.

Schritt	Aktivität	erwartetes Resultat
1	Abfahrtsort „Luzern, Bahnhof“ eingeben	„Luzern, Bahnhof“ wird vom Autocomplete erkannt und eingegeben.
2	Ankunftsort „Luzern, Maihof“ eingeben	„Luzern, Maihof“ wird vom Autocomplete erkannt und eingegeben.
3	„RESET“-Button drücken	„Luzern, Bahnhof“ und „Luzern, Maihof“ werden aus dem Feld gelöscht.

8.6. Test „Abfahrt & Ankunft tauschen“

Der User möchte den Abfahrts- & Ankunftsort tauschen.

Schritt	Aktivität	erwartetes Resultat
1	Abfahrtsort „Luzern, Bahnhof“ eingeben	„Luzern, Bahnhof“ wird vom Autocomplete erkannt und eingegeben.
2	Ankunftsort „Luzern, Maihof“ eingeben	„Luzern, Maihof“ wird vom Autocomplete erkannt und eingegeben.
3	„SWAP“-Button drücken	„Luzern, Bahnhof“ und „Luzern, Maihof“ werden getauscht.

9. TESTPROTOKOLL

9.1. Test „Stationssuche“

Durchgeführt von: Tamila Govduchanova

Durchgeführt am: 23.04.2020

Der User möchte mit dem Textfeld Abfahrts- & Ankunftsort suchen.

Schritt	Aktivität	erwartetes Resultat	tatsächliches Resultat	Status
1	Abfahrtsort „Luzern, Bahnhof“ eingeben	„Luzern, Bahnhof“ wird vom Autocomplete erkannt und eingegeben.		✓
2	Ankunftsort „Luzern, Maihof“ eingeben	„Luzern, Maihof“ wird vom Autocomplete erkannt und eingegeben.		✓

9.2. Test „aktuelle Verbindungen suchen“

Durchgeführt von: Tamila Govduchanova

Durchgeführt am: 23.04.2020

Der User möchte die aktuellen Verbindungen zwischen Abfahrts- & Ankunftsort sehen.

Schritt	Aktivität	erwartetes Resultat	tatsächliches Resultat	Status
1	Abfahrtsort „Luzern, Bahnhof“ eingeben	„Luzern, Bahnhof“ wird vom Autocomplete erkannt und eingegeben.		✓
2	Ankunftsort „Luzern, Maihof“ eingeben	„Luzern, Maihof“ wird vom Autocomplete erkannt und eingegeben.		✓
3	alle Verbindungen suchen drücken	Es werden die nächsten 4-5 Verbindungen zwischen „Luzern, Bahnhof“ und „Luzern, Maihof“ angezeigt		✓

9.3. Test „Verbindungen nach bestimmter Station suchen“

Durchgeführt von: Tamila Govduchanova

Durchgeführt am: 23.04.2020

Der User möchte die Verbindungen ab Abfahrtsort angezeigt bekommen.

Schritt	Aktivität	erwartetes Resultat	tatsächliches Resultat	Status
1	Abfahrtsort „Luzern, Bahnhof“ eingeben	„Luzern, Bahnhof“ wird vom Autocomplete erkannt und eingegeben.		✓
2	„alle Abfahrten anzeigen“ drücken	alle aktuellen Verbindungen ab „Luzern, Bahnhof“ werden angezeigt.		✓

Durchgeführt von: Tamila Govduchanova

Durchgeführt am: 23.04.2020

Der User möchte die Verbindungen an Ankunftsart angezeigt bekommen.

Schritt	Aktivität	erwartetes Resultat	tatsächliches Resultat	Status
1	Ankunftsart „Luzern, Maihof“ eingeben	„Luzern, Maihof“ wird vom Autocomplete erkannt und eingegeben.		✓
2	„alle Ankünfte anzeigen“ drücken	alle aktuellen Verbindungen an „Luzern, Maihof“ werden angezeigt.		✓

9.4. Test „Verbindungen nach bestimmter Zeit suchen“

Durchgeführt von: Tamila Govduchanova

Durchgeführt am: 23.04.2020

Der User möchte die Verbindungen zwischen Abfahrts- & Ankunftsort nach Zeit suchen.

Schritt	Aktivität	erwartetes Resultat	tatsächliches Resultat	Status
1	Abfahrtsort „Luzern, Bahnhof“ eingeben	„Luzern, Bahnhof“ wird vom Autocomplete erkannt und eingegeben.		✓
2	Ankunftsort „Luzern, Maihof“ eingeben	„Luzern, Maihof“ wird vom Autocomplete erkannt und eingegeben.		✓
3	Abfahrtszeit eingeben	Abfahrtszeit wurde im dafür vorgesehenen Feld eingegeben.		✓
4	„alle Verbindungen suchen“ drücken	Es werden die aktuellen Verbindungen zwischen „Luzern, Bahnhof“ und „Luzern, Maihof“ angezeigt.		✓

9.5. Test „Reset Button“

Durchgeführt von: Tamila Govduchanova

Durchgeführt am: 23.04.2020

Der User möchte seine Ortseingaben nicht manuell löschen müssen, er fordert einen Button.

Schritt	Aktivität	erwartetes Resultat	tatsächliches Resultat	Status
1	Abfahrtsort „Luzern, Bahnhof“ eingeben	„Luzern, Bahnhof“ wird vom Autocomplete erkannt und eingegeben.		✓
2	Ankunftsort „Luzern, Maihof“ eingeben	„Luzern, Maihof“ wird vom Autocomplete erkannt und eingegeben.		✓
3	„RESET“-Button drücken	„Luzern, Bahnhof“ und „Luzern, Maihof“ werden aus dem Feld gelöscht.		✓

9.6. Test „Abfahrt & Ankunft tauschen“

Durchgeführt von: Tamila Govduchanova

Durchgeführt am: 23.04.2020

Der User möchte den Abfahrts- & Ankunftsort tauschen.

Schritt	Aktivität	erwartetes Resultat	tatsächliches Resultat	Status
1	Abfahrtsort „Luzern, Bahnhof“ eingeben	„Luzern, Bahnhof“ wird vom Autocomplete erkannt und eingegeben.		✓
2	Ankunftsort „Luzern, Maihof“ eingeben	„Luzern, Maihof“ wird vom Autocomplete erkannt und eingegeben.		✓
3	„SWAP“Button drücken	„Luzern, Bahnhof“ und „Luzern, Maihof“ werden getauscht.		✓

10. INSTALLATIONSANLEITUNG

1. Öffnen Sie diese Seite: <https://github.com/antivitamila/modul-318-student>
2. Wählen Sie den Ordner „Installation“
3. Installieren Sie den Ordner auf Ihren PC/Laptop.
4. Starten Sie die „SBBv2“ (.application) Datei.
5. Der Download startet automatisch.

ABBILDUNGSVERZEICHNIS

Abbildung 1: Modul_318_Modulidentifikation und Handlungsnotwendige Kenntnisse.pdf, 20.04.2020	4
Abbildung 2: Mockup Gui erstellt mit "Balsamiq", 21.04.2020	11
Abbildung 3: UseCase-Diagramm, 21.04.2020	12
Abbildung 4: Aktivitätsdiagramm, 21.04.2020	15
Abbildung 5: UML-Diagramm, 21.04.2020	16

SELBSTSTÄNDIGKEITSERKLÄRUNG

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne unerlaubte fremde Hilfe verfasst und keine anderen Hilfsmittel als die angegebenen verwendet habe.

Insbesondere versichere ich, dass ich alle wörtlichen und sinngemässen Übernahmen aus anderen Werken als solche kenntlich gemacht habe.

Luzern, 24.04.2020
Ort, Datum


Unterschrift