Tatyana Merlo – Natalie Stalder – Nadja Stadelmann

Starbucks Manager

M120 – Benutzerschnittstelle entwickeln

# Einleitung

In einem vorherigen Projekt haben wir für Starbucks ein Menümanager geschrieben. Die Mitarbeiter können Produkte zum Menu hinzufügen, bearbeiten und wieder löschen. Weiter wird das Menü natürlich auch für die Kunden angezeigt.

Bisher wurde kein GUI eingesetzt, weshalb für die Klienten von Starbucks ein eher unschönes Menü angezeigt wurde. Weiter konnte das Programm nur über die Konsole bearbeitet werden, sodass eine spezielle Mitarbeiterschulung für den Gebrauch des Starbucksmanagers nötig war.

Dies wollen wir in diesem Folgeprojekt nun ändern.

## Dokument

### Aufbau

* Analyse
  + Beschrieb und erste Aufsplittung in einzelne Teilbereiche
* Vorgehen
  + Abhängigkeiten, Aufgabenübersicht
* Design
  + Detaillierte Analyse DB-Design
  + Detaillierte Analyse Programmstruktur
* Implementation
* Testing
  + Testcases basieren auf UseCases

Inhalt

[Einleitung 1](#_Toc496634521)

[Dokument 1](#_Toc496634522)

[Aufbau 1](#_Toc496634523)

[Vision 3](#_Toc496634524)

[Analyse 3](#_Toc496634525)

[Anforderungen 3](#_Toc496634526)

[Muss-Kriterien 3](#_Toc496634527)

[Kann-Kriterien 3](#_Toc496634528)

[Use Cases 4](#_Toc496634529)

[Design 5](#_Toc496634530)

[Filestruktur 5](#_Toc496634531)

[Programm 5](#_Toc496634532)

[Model 5](#_Toc496634533)

[View 5](#_Toc496634534)

[Controller 6](#_Toc496634535)

[Überlegungen zum GUI 6](#_Toc496634536)

[Implementation 7](#_Toc496634537)

[Implementierung 7](#_Toc496634538)

[Verantwortlichkeiten 7](#_Toc496634539)

[Technischer Aufbau 7](#_Toc496634540)

[GUI 7](#_Toc496634541)

[DataHolder 7](#_Toc496634542)

[Testing 8](#_Toc496634543)

# Vision

Im Modul 326 haben wir ein Starbucks-Menümanager mit Java erstellt. Bisher konnte der jedoch nur textbasiert verwendet werden (mittels Konsole).

Unser Ziel ist nun, den Mitarbeitern das Mutieren des Menüs mit einem GUI zu erleichtern. Das GUI soll alle bisherigen Userinteraktionen abdecken, sprich:

* Menü anzeigen
* Produkt hinzufügen
* Produkt bearbeiten
* Produkt löschen
* Programm beenden

Weiter möchten wir eine Login- sowie Logoutfunktion einbauen, um die Sicherheit der Daten zu gewährleisten.

Projektauftrag einfügen

# Analyse

## Anforderungen

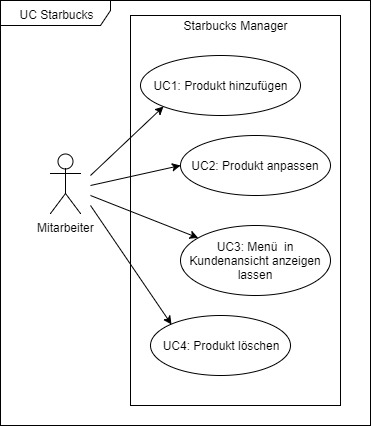
### Muss-Kriterien

1. Menüanzeige für die Kunden
   1. an Starbucksdesign angepasst
   2. gegliedert nach Produktkategorien
2. Menüanzeige für den Mitarbeiter
   1. Edit-Funktion zu jedem Produkt
   2. Delete-Funktion zu jedem Produkt
   3. selbe Gliederung wie bei der Kundenansicht
   4. Überbegriffe der einzelnen Felder sichtbar (bsp. Price)
3. Add-Funktion
   1. eigene Ansicht
   2. Auswahl der Produktkategorie
   3. Eingabe der Produktinformationen
      1. nur die relevanten Infos müssen eingegeben werden
4. Edit-Funktion
   1. Item bearbeiten
5. Delete-Funktion
   1. Item löschen
6. Layout
   1. Einheitlich
   2. Schlicht
   3. Gemäss Kapitel sowieso
7. Error-Handling
   1. genaue Informationen für den User

### Kann-Kriterien

1. Preis-Eingabe
   1. Dialogfenster mit Taschenrechnerlayout
2. Add-Funktion
   1. Speicherung relevante Daten beim Wechsel der Produktkategorie
3. Edit-Funktion
   1. Produktkategorie anpassen können
4. Help-Funktion
   1. Beschrieb der Funktionen
   2. Beschrieb des Files product.txt
5. Error-Handling
   1. Dialogfenster
      1. Message
      2. OK-Button als Userbestätigung

## Use Cases



|  |  |
| --- | --- |
| **Identifikation** | **UC1: Produkt hinzufügen** |
| **Beschreibung** | Ein Produkt wird hinzugefügt. |
| **Primärer Actor** | Mitarbeiter |
| **Vorbedingungen** | - |
| **Trigger** | Mitarbeiter wählt Menü „Add“ |
| **Standardablauf** | 1. Wählt Menü „Add“  2. Wählt Produktkategorie  3. Gibt Produktdaten ein  4. Produkt wird hinzugefügt |
| **Alternative Abläufe** | 2a. falsche Produktkategorie gewählt 🡪 User kann Kategorie anpassen ohne Datenverlust  3a. Produkt bereits vorhanden 🡪 Fehlermeldung ausgeben  3b. Falsche Zeichen eingegeben 🡪 Fehlermeldung ausgeben |

|  |  |
| --- | --- |
| **Identifikation** | **UC2: Produkt anpassen** |
| **Beschreibung** | Ein bestehendes Produkt wird editiert |
| **Primärer Actor** | Mitarbeiter |
| **Vorbedingungen** | Produkt vorhanden  Menü „Modify“ ausgewählt |
| **Trigger** | Mitarbeiter wählt Option „edit“ |
| **Standardablauf** | 1. Wählt Option „edit“  //noch anschauen wie natalie es handelt  2. Produkt wird angepasst |
| **Alternative Abläufe** | 2a. Falsche Zeichen eingegeben 🡪 Fehlermeldung ausgeben |

|  |  |
| --- | --- |
| **Identifikation** | **UC3: Menü in Kundensicht anzeigen** |
| **Beschreibung** | Das Menü wird strukturiert und im Starbucksdesign ausgegeben |
| **Primärer Actor** | Mitarbeiter |
| **Vorbedingungen** | - |
| **Trigger** | Mitarbeiter wählt Menü „Home“ / Programmstart |
| **Standardablauf** | 1. Wählt Menü „Home“ / startet Programm  2. Produkte werden nach Typ sortiert ausgegeben |
| **Alternative Abläufe** | - |

|  |  |
| --- | --- |
| **Identifikation** | **UC4: Produkt löschen** |
| **Beschreibung** | Ein bestehendes Produkt wird entfernt |
| **Primärer Actor** | Mitarbeiter |
| **Vorbedingungen** | Produkt vorhanden  Menü „Modify“ ausgewählt |
| **Trigger** | Mitarbeiter wählt Option „delete“ |
| **Standardablauf** | 1. Wählt Option „delete“  //Noch anschauen wie natalie es handelt  2. Produkt wird entfernt |
| **Alternative Abläufe** | - |

wenige, aber klare aus Sicht des Benutzers

## Story-Bord

inklusive mockups (inklusive Dialogspezifikation)

🡪 hier erwartete Eingaben und Ausgaben genau beschreiben

# Design

## Filestruktur

Der bisherige Starbucksmanager hat keine Datenbank, welche er verwaltet. Die Daten werden in einem einfachen File abgespeichert. Dennoch ist die Struktur des Files natürlich extrem wichtig für die Datenverarbeitung, weshalb sie hier kurz deklariert ist.

Jede Zeile steht für einen Menü-Eintrag, der in einzelne Komponenten unterteilt ist. Je nachdem, welche Attribute der Eintrag mitführt, wird er in eine andere Kategorie unterteilt.

**Kategorie Recordstruktur**

Beverage Name | Preis | null | true (hot) / false (cold)  
beverage¦2.3¦null¦true

Coffee Name | Preis |Zutaten  
coffee¦1.4¦ingr

Extra Name | Preis  
extra¦2.0

Food Name | Preis |Zutaten | Vegi / Vegan / Glutenfrei / usw.  
food¦1.5¦Ingr¦Vegi

## Programm

### Model

Wie bereits erwähnt werden die Daten in einem einzigen File gehandhabt. Das File und sein Handling besteht bereits. Dennoch müssen am Datenmodel noch einiges angepasst werden, um die korrekte Zusammenarbeit mit dem GUI sicherzustellen.

### View

Das GUI ist komplett neu umzusetzen. Bisher waren nur Kommandozeileneingaben zur Steuerung des Programmes da. Dies soll neu über ein GUI geregelt werden, um den Mitarbeitern einen leichteren Einstieg zu ermöglichen.

So stellen wir uns das Ergebnis des GUIs vor:

Die Kundenansicht soll dem Starbucksdesign angepasst sein. In ihrem Logo ist ein spezielles Grün zu finden. Weiter zeigt es eine Weisse Figur auf schwarzem Hintergrund. Um den Kunden eine möglichst angenehme Ansicht bieten zu können, wird für das GUI das Grün als Hintergrund eingesetzt. Darauf positionieren werden die Artikel in weisser, serifenloser Schrift. Diese garantiert einen guten Lesefluss. Um dem Kunden die Auswahl möglichst zu vereinfachen, werden die Produkte in vier Kategorien unterteilt. Die Kaffees sind zu Oberst zu finden, da die meisten Kunden Starbucks wegen diesen Produkten besuchen. Danach folgen die Beverages (weitere Getränke), die Extras und zum Schluss noch die Muffins und Cookies unter der Kategorie Food.

Für die Mitarbeiteransicht wollen wir ein einheitliches Layout erstellen, damit der User sich schnell auf allen Seiten zurechtfinden. Es soll ausserdem übersichtlich sein, das heisst für uns, dass der User immer nur das angezeigt bekommt, was er wirklich braucht. Als Beispiel ist hier die Funktion ‘Produkt hinzufügen’ super geeignet. Ein Kaffee oder ein Extra benötigen weniger Angaben als ein neues Produkt der Kategorie Food. Also sollen auch nur diese Angaben abgefragt werden, welche notwendig sind.

Für all diese Views verwenden wir CSS für JavaFX, um die verschiedenen Teile des GUIs einheitlich gestalten zu können.

### Controller

Die Controller bestehen teilweise, es wurden bereits Usereingaben verarbeitet. Dennoch müssen sie an das GUI angepasst werden, da das Programm bisher nur über die Konsole bedient werden konnte. Teils müssen sie ganz ausgewechselt werden.

Einige Überprüfungen der Usereingaben werden neu vom GUI bereits abgefangen. So wird die Businesslogik entlastet, da nicht mehr alle Eingaben auf ihre Richtigkeit überprüft werden müssen. Ein Preis, welcher vom GUI weitergeliefert wird, ist beispielsweise immer ein double wenn er im Controllerbereich ankommt.

## Klassendiagramm

### GUI

### Interaktion mit Model

# Überlegungen zum GUI

Da sich dieses Modul vor allem um die GUI-Implementierung und das Design des GUIs dreht, gehen wir in diesem Kapitel genauer auf unsere Überlegungen dazu ein.

## Klärung der Ebenen

Als erstes müssen einige Aspekte geklärt werden, um genau eruieren zu können, was das GUI alles können muss. Dies geschieht mittels verschiedenen Ebenen, welche hier genauer erläutert und auf unser GUI bezogen bearbeitet werden.

Rot sind die Aspekte aus den Aufgaben—PDFs!

### Aufgaben-Ebene

Als erstes stellen wir uns die Frage, welches nun die primären Aufgaben des Benutzers sind. Nur wenn wir wissen, für was der Benutzer das Programm im Endeffekt benützen wird, können wir es auch entsprechend gestalten. In unserem Fall sind dies folgende Aufgabe:

* Starbucks-Menü kundenfreundlich und aktuell anzeigen lassen

Um diese Endaufgabe erreichen zu können, muss der Benutzer auch folgende Aufgaben bewältigen können:

* Produkte hinzufügen
* Produkte bearbeiten
* Produkte löschen

Welches ist die primäre Aufgabe des Benutzers?

### Semantische Ebene

Weiter ist es wichtig zu wissen, welche Werkzeuge dem Benutzer bereits zur Verfügung stehen.

Welche Werkzeuge stehen zur Verfügung?

### Syntaktische Ebene

Auf dieser Ebene ist die Frage nach dem Wissen des Benutzers extrem wichtig. Dies kann von Benutzer-Art zu Benutzer-Art unterschiedlich ausgeprägt sein. Je nach Komplexität des Programmes und nach Arbeitsteilung muss das Programm dementsprechend angepasst werden. Allenfalls müssen die Benutzer sogar geschult werden, damit das Programm im Arbeitsalltag (oder auch im Privatgebrauch) korrekt und effizient eingesetzt werden kann.

In unserem Fall gibt es zwei Benutzer-Arten (mehr dazu weiter unten).

Welches Wissen hat der Benutzer?

### Interaktionsebene

Wie kann der Benutzer mit dem System kommunizieren?

## Benutzer-Arten

Neue / Geübte / Profis 🡪 was für Erkenntnisse gewinnen wir daraus?

## Weitere Aspekte

• Welche Erwartungen hat der Benutzer auf die Interaktion zur Lösung seiner Aufgaben?

• Welche Metaphern können verwendet werden?

• Müssen Style-Guides verwendet werden? Sind Design-Vorlagen vorhanden?

• Müssen bestimmte Richtlinien umgesetzt/berücksichtigt werden?

# Implementation

## Implementierung

### Verantwortlichkeiten

Für die Implementierung haben wir unser GUI in drei Verantwortlichkeitsbereiche unterteilt. Für den jeweiligen Verantwortlichen bedeutet dies, dass die Implementation sowie das Testing der jeweiligen Funktionen seine Aufgabe ist. Natürlich bestehen einige Abhängigkeiten zwischen den einzelnen Bereichen, die wir miteinander besprechen werden. So wird auch unsere Zusammenarbeit gewährleistet.

Die Kundenansicht, welche mittels WebView (???) gestaltet wird, ist unter Tatyana Merlo’s Verantwortlichkeit. Auch das Grundlayout gehört in ihren Bereich. Das Hinzufügen der Daten und die Anzeige der ‘Help’-Seite liegt unter Nadja Stadelmann’s Bereich. Natalie Stalder ist für die Mitarbeiteransicht zuständig, welche gleichzeitig die Funktionen ‘Produkt bearbeiten’ und ‘Produkt löschen’ beinhaltet.

### Technischer Aufbau

#### GUI

Unser Template beinhaltet das Fenster der Applikation inklusive Titel, Menüleiste und Footer. Je nach Auswahl der Menübuttons durch den User werden nun die eigentlichen Fensterinhalte ausgetauscht. Der Inhalt der Kundenansicht wird betreffend des Layouts speziell behandelt. Alle Mitarbeiteransichten werden mittels demselben CSS-Stylesheet formatiert. Dadurch stellen wir sicher, dass der User bei der Anwendung eine einheitliche Ansicht zu sehen bekommt.

#### DataHolder

Um die Datenintegrität sicherzustellen, wird ein DataHolder verwendet. Er beinhaltet die möglichen Daten eines einzelnen Produktes.

Die Views AddItem und ChangeItem verwenden diesen, um für den User sicherzustellen, dass seine Daten zwischengespeichert sind. So klickt der User beispielsweise auf Add > Coffee und gibt den Namen eines neuen Produktes der Kategorie Beverage ein. Um ihm den Kategorienwechsel zu vereinfachen, werden die bereits eingegebenen Daten aus dem DataHolder in die neue Anzeige eingespeist.

Entwicklungsprozess beschreiben

# Testing

Neben Tests auch ein Benutzer-Feedback verfassen