

# Software-Entwicklung 2

Wintersemester 2023

## PICK'N DRINK

https://gitlab.mi.hdm-stuttgart.de/mb429/se-2-projekt

## **AUTOREN**

- Malena Böckmann, mb429@hdm-stuttgart.de, Matrikelnummer: 46083
- Julia Ebert, je073@hdm-stuttgart.de, Matrikelnummer: 45877
- Elisabeth Götz, eg054@hdm-stuttgart.de, Matrikelnummer: 45827

## 1. Kurzbeschreibung

Bei unserem Projekt "PICK'N DRINK" handelt es sich um eine Bestell-App für Drinks.

Beim Öffnen unserer App startet der Benutzer im Login-Fenster. Ein neuer Nutzer kann mit einem Klick zur Registrierung gelangen, wo er eine korrekte E-Mail-Adresse, ein sicheres Passwort und sein Alter angeben muss. Falls das Alter manuell eingegeben wird, also nicht über das aufklappbare Datumsfeld, ist es wichtig, dass dies noch mit einem "Enter" bestätigt wird. Es ist zudem zu beachten, dass das Alter des Nutzers über 18 Jahren sein muss. Nach erfolgreicher Registrierung kann der Nutzer zurück in den Login Bereich und sich dort bequem anmelden, was ihn direkt zur Hauptseite weiterleitet. Hier erwartet den Nutzer eine vielfältige Auswahl an Cocktails, Shots und alkoholfreien "DriverDrinks". Sobald sich der Benutzer eine Wahl getroffen hat, kann er diese im Warenkorb überprüfen. Der Warenkorb ermöglicht es dem Nutzer, seine Bestellung mithilfe von Plus- und Minus-Buttons anzupassen. Hier wird auch der Gesamtpreis transparent angezeigt, um eine klare Übersicht zu gewährleisten. Mit einem einfachen Klick auf "Order" wird die Bestellung abgeschlossen und dem Nutzer wird eine Bestellnummer zugewiesen. Über den vorläufigen Fortschrittsbalken kann der Status der Bestellung verfolgt werden (In unserem Fall ist dieser statisch vereinfacht, die Bestellnummer würde später dem Servieren der Getränke dienen).

### 2. Startklasse

Unsere Startklasse ist unsere Driver-Klasse, diese befindet sich im "mainpackage".

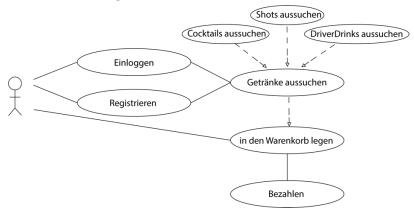
## 3. Besonderheiten

Bei der Registrierung wird genau 1 Nutzer mit passendem Passwort in eine CSV-Datei geschrieben. Jedes Mal, wenn ein Nutzer sich neu registriert, wird der alte Nutzer überschrieben (als provisorische Registrierung gedacht).

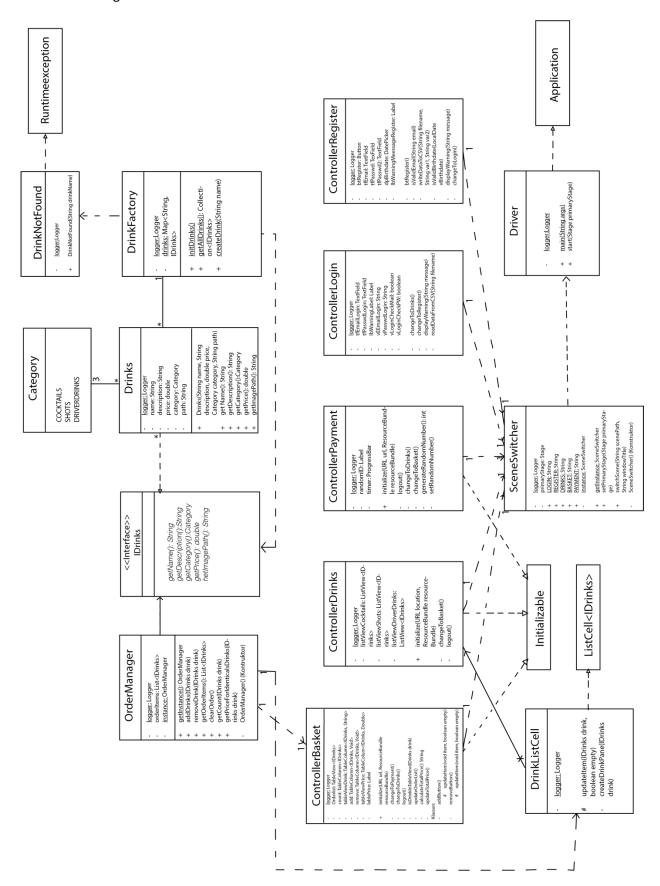
Zudem wird der Bezahlvorgang simuliert, das heißt es gibt keine reale Transaktion. Der Nutzer wird nach dem "Order"-Button auf eine Überblick-Seite gebracht, die die Bestellnummer und den Bestellstatus anzeigt, mit diesem Szenenwechsel ist die Bestellung abgeschlossen.

## 3. UML Diagramme

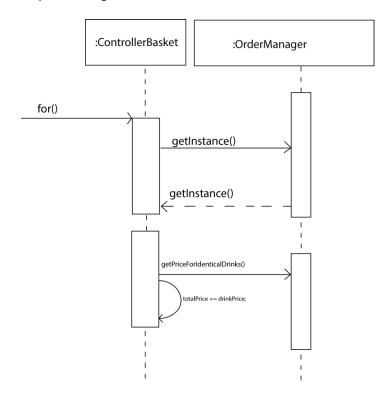
Die UML-Diagramme finden Sie zusätzlich im git-Repository unter "Anhänge". UML Use Case Diagramm



## **UML Klassendiagramm**



## Sequence Diagramm



## 4.Stellungnahme

## Interfaces/Architektur:

- Vererbung wurde eingebaut: Beispielsweise ControllerBasket-Klasse implementiert die Initializable Klasse, die DrinkListCell-Klasse erbt von ListCell<IDrinks>
- Interface → Klasse IDrinks
- Das Interface IDrinks wird von der Klasse Drinks implementiert.
- Factory → DrinkFactory

#### Clean Code:

- In Bezug auf das Singelton Pattern wurde zum Beispiel in der OrderManager-Klasse ein static verwendet (siehe Zeile 26).
- Getter geben in unserem Projekt generell keine direkten Referenzen zurück → Ausnahmen hier: Singelton-Pattern in der SceneSwitcher-Klasse und in der OrderManager-Klasse
- Lose Kopplung → Durchgängige Verwendung des Interfaces IDrinks (z.B.: sichtbar in der ControllerDrinks-Klasse)

## **Tests**

- Tests sind unter src.test.java.mainpackage zu finden und wurden mit JUnit Tests erstellt
- Es wurden mehrere Klassen getestet: die Drinks-Klasse und die DrinkFactory.

- Die Tests testen jeweils die Methoden auf ihren Normalfall sowie auf einen Fehlerfall → Negativtest, Hierbei wird zum Beispiel getestet ob bei einem Fehlerfall eine bestimmte Exception ausgelöst wird.

#### GUI

- Es sind insgesamt 5 verschiedene Szenen vorhanden. Unter src.main.java.mainpackage.GUI können Sie die Controller-Klassen finden. In dem src.main.resources-Ordner können Sie die passenden FXML-Dateien entnehmen. Es wurden verschachtelte Oberflächen genutzt und in einer Vielzahl erstellt.
- Event-Handling → ist bei den Buttons zu sehen, zum Beispiel Zeile 102 in der ControllerDrinks-Klasse: es wird durch den Warenkorb-Button im Menü eine changeToBasket() Methode aufgerufen.

## Logging/Except.

- Logging in die "A1.log" File, Konfiguration in main.resources → log4j2.xml
- In jeder Klasse ist Logging mit passendem Logging-Framework vorhanden.
- Je nach Bedarf wird in unserem Code auf allen drei verschiedenen Leveln geloggt (info, warn, error).
- In der ControllerDrinks-Klasse werden in Zeile 57, 73 und 89 die Threads geloggt.
- In der DrinkFactory-Klasse wird in der Methode getDrink() zum Beispiel eine Exception geloggt, falls der catch-Block ausgeführt wird (siehe Zeile 67)
- In der gleichen Methode (getDrink()) wird unsere selbst erstellte DrinkNotFound-Exception geworfen (siehe Zeile 62) und im catch-Block gefangen.

#### **Threads**

- In der ControllerDrinks-Klasse werden in der initialize-Methode drei Threads erstellt und gestartet: "cocktailsThread", "shotsThread" und "driverDrinksThread" → diese erstellen die jeweiligen GUI-Oberflächen für die Reiter "Cocktails", "Shots" und "DriverDrinks".

#### Streams

- In der ControllerDrinks-Klasse sind die Streams zu finden, jeweils in den einzelnen Threads.
  - → Zeile 46, 62 und 78
- Lambdas → zum Beispiel in der ControllerDrinks-Klasse > Methode initialize() > Zeile 44 > new Thread (() -> {...}), oder Zeile 51 > Platform.runLater(() -> {...}

## 5. Ausgefüllter Bewertungsbogen (Excel) für das Projekt

	А	В	С	D	Е	F	G	Н	1	J	К	L	М	N	0	Р	Q
1	ID	Vorname	Nachname	Kürzel	Matrikelnummer	Projekt	Interfaces/	Clean Code	Doku	Tests	GUI	Logging/Except.	υмі	Threads	Streams	Nachdenkzette	Summe - Projekt
2		Julia	Ebert	je073	45877	PICK'N DRINK	3	3	3	3	3	3	3	3	3	3	30,00
3		Elisabeth	Götz	eg054	45827	PICK'N DRINK	3	3	3	3	3	3	3	3	3	3	30,00
4		Malena	Böckmann	mb429	46083	PICK'N DRINK	3	3	3	3	3	3	3	3	3	3	30,00