

VERTIEFUNG DER PROGRAMMIERUNG

Projektbericht im Rahmen der Portfolio-Prüfung SoSe 2020

1. ZEITRAUM: 13.07.2020 – 09.08.2020

Gruppe 11

Gruppenmitglied: Kjell Treder

Gruppenmitglied: Marcel Sauer

1. Spezifikation

Mit der Anwendung kann der Nutzer Plätze für ein Autokino reservieren.

Ein Film und eine der verfügbaren Zeiten werden ausgewählt, um eine Ansicht der verfügbaren Plätze zu erhalten. Der Nutzer kann dann einen oder mehrere Plätze für die Reservierung auswählen. Es gibt Stellplätze für normal große Autos und Strandkörbe, sowie gegen Aufpreis beide Varianten als VIP mit besserer Sicht. Des Weiteren sind im hinteren Bereich Stellplätze für SUVs buchbar.

Nach der Auswahl optionaler Angebote wie Essen und Trinken (Catering) kann die Reservierung bestätigt werden.

Der Nutzer erhält daraufhin eine Rechnung sowie eine Reservierungsnummer, mit welcher er zum Vorstellungstermin die Kinofläche betreten kann.

Hinweise:

Die Datenbank (Database) beinhaltet in den Listen einige Filme und Caterings, welche einige ihrer Datenfelder auf null gesetzt bekommen haben oder nicht richtig gesetzt bekommen haben. Diese Filme und Caterings sind zum Testen gedacht und sollen zeigen, dass der Code auch mit solchen Fehlern umgehen kann.

Alle `System.out.println()` sind ausschließlich für Debug Prints gedacht, um den Code besser zu durchblicken. Mithilfe der Konsole kann der Programmablauf gut nachvollzogen werden um ein besseres Verständnis dafür zu bekommen, wie die Klassen miteinander kommunizieren, welche Methoden ausgeführt werden und welche Daten zu welcher Zeit gespeichert werden. Die Konsole zeigt also keine, für den Endnutzer wichtigen Daten an!

Die "Aufbau Swing Komponenten" PDF kann einen guten Überblick über die Verschachtelung der Komponenten geben.

1.1. Benutzergruppen

Die Anwendung wird hauptsächlich von Nutzern verwendet, die gerne ins Autokino gehen oder Autokinos bzw. alternative Kinoerlebnisse ausprobieren möchten und alternativ die Reservierung manuell vor Ort beim Autokino gemacht hätten.

Zu dieser Nutzergruppe gehören hauptsächlich Pärchen, Ehepaare und Familien mit Kindern, aber auch Studenten (ohne Auto), welche die Strandkörbe besetzen.

Auch, wenn der Besucherkreis Minderjährige einschließt, wird die Anwendung überwiegend von erwachsenen, bzw. Volljährigen Personen verwendet. Dies schließt neben technikversierten Personen auch ältere Menschen ein, welche mit komplexen Anwendungen nicht vertraut sind.

1.2. Funktionen

Auswahl eines Kinofilms und der Vorstellungszeit

Der Nutzer kann einen von mehreren angebotenen Kinofilmen auswählen.

Der Nutzer kann eine der gelisteten Vorstellungszeiten des ausgewählten Kinofilms auswählen.

Auswahl der Stellplätze/Strandkörbe

Der Nutzer kann einen oder mehrere Stellplätze und/oder Strandkörbe auswählen.

Der Nutzer muss das Kennzeichen des Autos für jeden ausgewählten PKW/SUV-Stellplatz angeben.

Auswahl von Essen und Trinken

Der Nutzer kann einer Reservierung jeweils bis zu neun Einheiten unterschiedlicher Snacks und Getränke hinzufügen.

Reservierung überprüfen, bestätigen und abschicken

Der Nutzer erhält final eine Übersicht seiner Auswahl und kann diese, nachdem er sie überprüft hat, endgültig abschicken.

Navigation (UI / UX)

Der Nutzer wird durch die Schaltflächen durch den Bestellvorgang geleitet.

Hierbei kann er mit den Schaltflächen „Zurück“, „Abbrechen“ und „Fortfahren“ den vorherigen und nächsten Bestellschritt einsehen, bzw. den Reservierungs-Vorgang abbrechen.

Der Nutzer kann einen vorherigen spezifischen Tab einsehen, indem dieser in der Ansicht oben angeklickt wird.

Ein Voranschreiten mit fehlenden Pflichtangaben wird unterbunden.

2. Entwurf

2.1. Klassendiagramm

Die `KinoView` erstellt das `KinoModel` und den `KinoController`. Der Controller bekommt eine Referenz auf die View und das Model im Konstruktor als Parameter mitgegeben. Das Model kennt die View und den Controller nicht. Die View besteht aus mehreren `AbstractTabs`, welche ebenfalls Zugriff auf das Model und den Controller bekommen. Für jeden Tab, der im Programm angezeigt werden soll existiert eine Klasse, welche von dem `AbstractTab` erbt. Diese Klassen müssen auch die abstrakten Methoden `build()` und `update()` vom `AbstractTab` implementieren. Jeder Tab holt sich die jeweiligen Daten aus dem Model um "sich aufzubauen". Das Model beinhaltet alle ausgewählten Daten des Users, erstellt Berechnungen für z.B. den Preis und prüft, ob eingegebene Kennzeichen genügen.

Kennzeichen müssen zwischen 5 und 9 Zeichen lang sein und müssen wie folgt aufgebaut werden:

1. 1-3 Buchstaben gefolgt von
2. einem optionalem Bindestrich gefolgt von
3. 1-2 Buchstaben gefolgt von
4. 1-4 Ziffern

Alle Whitespaces, Tabulatoren, etc. werden aus dem Kennzeichen entfernt.

Die Datenbank beinhaltet Listen mit allen Filmen und Caterings, sowie Ticket- und Bestellnummern und Kennzeichen und Bestellungen. Jeder Film besitzt neben einem Pfad für ein Bild und einem Titel auch ein Genre und eine FSK und kann beliebig viele Showtimes besitzen. Eine Showtime besteht aus einem Tag (Enum: Days) und einer Uhrzeit (Enum: Times) und kann beliebig viele Reihen und Spalten an Stellplätzen (`AbstractSeat`) besitzen. Ein `AbstractSeat` kann entweder vom Typen `CarSeat` sein, wenn dieser ein Stellplatz für PKW und SUV sein soll, oder vom Typen `BeachChairSeat`, wenn dieser ein Stellplatz für Strandkörbe sein soll. Diese Plätze haben einen Preis (Enum: Prices). Das Reservieren eines Seats ist unterschiedlich: Der `CarSeat` bekommt ein Kennzeichen zugewiesen, während der `BeachChairSeat` mithilfe des `NumberManagers` eine Ticket Nummer generiert.

Der `NumberManager` ist für die Generation von einer einzigartigen Zahl für eine spezifische Liste verantwortlich und kann daher Zahlen für Tickets und Bestellnummern generieren. Sollte nach 1000 random generierten Zahlen keine einzigartige Zahl generiert worden sein, wird ein `StackOverflow Error` geworfen.

Eine Bestellung kann vom Model erstellt werden und beinhaltet alle wesentlichen Informationen über die Bestellung, sowie eine Bestellnummer. Das Model nutzt den `FileManager`, um die Bestellung in eine Datei zu schreiben. Sollte die Datei bereits existieren, wird ein Unterstrich ("_") an den Dateinamen gehangen und versucht die Datei neu zu erstellen.

Der `FileManager` ist für das Management von Dateien verantwortlich und kann Textdateien erstellen und löschen, sowie Bilder laden.

2.2. MVC

Das Model

Das Model ist dafür verantwortlich die vom User eingegebenen Daten zu speichern, zu löschen und zu managen. Dazu gehören:

- Der ausgewählte Film
- Die ausgewählte Zeit
- Die ausgewählten Plätze
- Die eingegebenen Kennzeichen
- Die Menge für jede Essen/Trinken-Option

Zusätzlich speichert und berechnet das Model Daten für die View, wie:

- Die verfügbaren Zeiten anhand des ausgewählten Filmes
- Die verfügbaren Plätze anhand der ausgewählten Zeit
- Die Anzahl an PKW/SUV-Stellplätzen (damit die View die richtige Anzahl an Textfeldern für die Kennzeichen zeigen kann)

Das Model kann anhand der Daten einen Preis kalkulieren und eine fertige Bestellung erstellen.

Das Erstellen einer Bestellung besteht aus:

1. Reservieren aller Plätze
2. Einem Update der Showtime, als Check ob das Reservieren der Plätze die Show ausverkauft hat
3. Dem Erstellen und Hinzufügen (in die Datenbank) eines neuen Bestellungs-Objekts, welches eine Bestellnummer generiert bekommt, die in der Datenbank gespeichert wird.
4. Schreiben eines neuen Dokuments mit der Beschreibung der Bestellung

Durch das Reservieren eines Platzes werden für Strandkörbe Tickets generiert, und PKW/SUV-Stellplätze Kennzeichen zugewiesen. Diese Daten werden auch in der Datenbank gespeichert.

Der Controller

Der Controller reagiert auf alle Inputs des Users und gibt die Informationen an das Model weiter. Er implementiert einen ActionListener, ItemListener und ChangeListener und erweitert den KeyAdapter. Der Controller kümmert sich um die Übergabe folgender Informationen an das Model:

- Der ausgewählte Film aus dem `ItemEvent`
- Der Index der ausgewählten Zeit aus dem *Action Command* des `ActionEvents`
- Die Koordinaten des geklickten Platzes aus dem Action Command des `ActionEvents` und ob dieser ausgewählt ist
- Eine Liste der Eingaben aller Textfelder für die Eingabe von Kennzeichen
- Eine Liste aller Mengen der verfügbaren Essen/Trinken-Optionen

Der Controller löst daraufhin das Update bei der View aus. Des Weiteren weist dieser die View zur Navigation auf, sollte der „Zurück“ oder „Fortfahren“ Button gedrückt werden.

Ebenfalls weist der Controller das Model an, das Programm zu schließen, wenn der „Abbrechen“ oder „Beenden“ Button geklickt wird.

Die View

Die View besteht aus sechs unterschiedlichen Sub-Klassen-Tabs, die alle unterschiedlich „gebaut“ und „geupdated“ werden. Es können beliebig viele Tabs hinzugefügt und erweitert werden. Jeder Tab besitzt einen kurzen Anweisungstext gefolgt von Komponenten, die für jeden Tab unterschiedlich sind und zuletzt drei universellen Buttons für die Navigation, welche im abstrakten Super-Klassen-Tab definiert sind.

Die Buttons haben folgende Eigenschaften:

- Zurück:

Der Zurück Button kann auf jedem Tab angeklickt werden, um auf den vorherigen Tab zu gelangen (Ausgeschlossen ist der erste Tab (StartTab)).

- Abbrechen:

Der Abbrechen Button kann auf jedem Tab angeklickt werden, um das Programm zu verlassen.

- Fortfahren:

Der Fortfahren Button wird immer dann genutzt, um auf den folgenden Tab zu gelangen. Dadurch wird der folgende Tab zwangsläufig „gebaut“. Die View switcht dann auf daraufhin auf diesen Tab, damit man diesen einsehen kann. Der Zugriff auf alle weiteren Tabs wird deaktiviert.

Der Fortfahren Button kann bei (den meisten) Tabs erst genutzt werden, nachdem eine bestimmte Aktion abgeschlossen wurde:

- Der Nutzer kann erst die Zeiten einsehen, wenn ein Film ausgewählt wurde
- Der Nutzer kann erst die Plätze einsehen, wenn eine Zeit ausgewählt wurde
- Der Nutzer kann erst die Essens/Trinkens-Optionen einsehen, wenn min. ein Platz ausgewählt wurde und alle erforderlichen Kennzeichen zureichend und korrekt eingegeben wurden.

Der Check, ob der Fortfahren Button für den Nutzer verfügbar ist, geschieht in der, in jedem Tab unterschiedlichen update()-Methode. Diese wird aufgerufen, wenn mit Komponenten auf dem Tab interagiert wurde. Ein Update des Tabs bedeutet zwangsläufig auch, dass alle folgenden Tabs deaktiviert werden, denn:

Beispiel:

Der Nutzer wählt einen Film aus und schaut sich die Zeiten an. Da ihm die Zeiten nicht passen, geht er einen Tab zurück und wählt einen anderen Film aus. Da das Bauen des folgenden Tabs nur durch das Switchen durch den Fortfahren Button ausgelöst wird, d.h. dass der Tab nur die aktuellen und richtigen

Informationen anzeigt, wenn der Fortfahren Button betätigt wurde, müssen alle folgenden Tabs deaktiviert werden.

Die `build()`-Methode jedes Tabs baut Komponenten abhängig von den Daten im Model auf. Sie wird jedesmal aufgerufen, wenn zu einem Tab mit dem Fortfahren Button gewechselt wird.

Beispiel:

Der Nutzer wählt einen Film und Zeiten aus und klickt auf Fortfahren. Dadurch wird der Tab aufgebaut, der dafür verantwortlich ist, die Plätze dieser Zeit anzuzeigen. Wie viele und welche Plätze die Zeit besitzt, d.h. wie viele Auswahlmöglichkeiten dem User angezeigt werden müssen ist abhängig von der ausgewählten Zeit (welche abhängig von dem ausgewählten Film ist). Sollte sich also die ausgewählte Zeit oder Film ändern, muss der folgende Tab neu gebaut werden um die neuen, richtigen Informationen anzuzeigen. Daher wird auch der Zugriff auf die folgenden Tabs deaktiviert, da diese bei einem Update möglicherweise nicht mehr die richtigen Informationen anzeigen würden.

Merke:

Keines der Filme(-objekte) hat die gleichen Zeiten(-objekte) und kein Zeiten(-objekt) hat die gleichen Platz(-objekte). Jeder Film hat individuelle Zeiten, welche je individuelle Plätze haben.

Die View kann ebenfalls Dialoge erstellen, wenn eine Exception auftritt oder die Bestellung final abgesendet wird.

Error Dialoge werden erstellt, wenn das „Bauen“ eines Tabs nicht gelingt. Es wird eine Exception von der `build()` Methode des Tabs geworfen, die in der View „gecatched“ wird. Dies kann in zwei Fällen auftreten:

1. Der ausgewählte Film hat keine Zeiten, da diese null sind
2. Die ausgewählte Zeit hat keine Plätze, da diese null sind oder da die Anzahl der Reihen oder Spalten dieser Plätze gleich 0 sind

In einem solchen Fall wird der Dialog mit einer zugehörigen Error-Message angezeigt. Der Nutzer kann den Dialog schließen und weiterhin die Software nutzen und z.B. einen anderen Film/eine andere Zeit auswählen

Der Finishing-Dialog erscheint, wenn der User die Bestellung abschickt. Dieser Dialog beinhaltet einige Bedankung und listet, falls vorhanden, Tickets für Strandkörbe auf. Das Abschicken einer Bestellung bringt den Nutzer wieder zum ersten Tab, sodass eine weitere Reservierung möglich wäre.

3. OOP

3.1. Umsetzung der Grundelemente der OO Entwicklung

1. Kapselung

Um den Zugriff auf bestimmte Programmteile einzuschränken oder zu unterbinden wurden die Sichtbarkeiten von Variablen nach Möglichkeit reduziert. Datenfelder sind meistens `private` oder `protected` und nur `public`, wenn das Datenfeld `read-only` (`final`) ist. Es werden `getter` Methoden für den Zugriff auf die Datenfelder genutzt. Primitive Datentypen werden mithilfe der `Getter` Methoden einfach übergeben.

`Listen`, `Arrays`, `Maps`, usw. werden jedoch als Kopie übergeben, um die Änderung der Inhalte dieser `Collections` in ihren Klassen zu vermeiden (`passed by value`, nicht `passed by reference`)

Der Boolean `isReserved` ist in der `AbstractSeat` Klasse als `protected` deklariert. Ein externer Zugriff erfolgt über eine `getter` Methode. Trotzdem ist es möglich, den boolean mithilfe der abstrakten `reserve()` Methode jeder Kindsklasse des `Abstract Tabs` zu setzen. Der grundlegende Unterschied zu einer `setter` Methode ist jedoch, dass jede Kindsklasse des `Abstract Seats` die Anforderungen für das Reservieren selbst definiert. Beispielsweise bekommen `Strandkörbe` eine einzigartige Ticketnummer generiert, während `PKW-Stellplätze` ein Kennzeichen zugewiesen bekommen. Die Klasse kümmert sich also um ihr eigenes Datenfeld.

2. Polymorphie

Die `FileManager`-Klasse als Beispiel, stellt die Funktion `loadImage()` zur Verfügung welche von anderen Klassen genutzt werden kann, um Bilder in das Programm zu laden. Dies wird u.a. von der Klasse `Movie` genutzt, um den `Movie`-Objekten ihre Filmposter zuzuweisen, aber auch von der `KinoView` um dem Hauptfenster das TH-Logo hinzuzufügen.

Weiter beinhaltet die Klasse `NumberManager` die Methode `nextFor()`, welche eine einzigartige Nummer beliebiger Länge für eine Liste generieren kann. Diese Methode kann vom gesamten Package genutzt werden um entsprechende Nummern zu generieren. So nutzen beispielsweise die Methoden `generateOrderNumber()` und `generateTicketNumber()` die Funktion, um jeweils unterschiedliche Nummern für ihre jeweiligen Anwendungszwecke zu generieren.

Ähnlich arbeitet die Methode `putInContainer()`. Auch hier handelt es sich um eine polymorphe Auslagerung, in diesem Fall, um beliebige Komponenten in ein `JPanel` zu legen.

Revers kann die Methode `getComponentsFrom()` dazu genutzt werden, alle Komponenten eines `JPanel`s (auch rekursiv, falls das `JPanel` `JPanels` beinhaltet) auszulesen und in einer separaten Liste auszugeben.

3. Abstraktion

Gemäß den Grundsätzen der Objektorientierten Programmierung wurde auf die Implementierung von Abstrakten Klassen und Methoden Wert gelegt. Genannt seien hier die Klassen `AbstractTab` und `AbstractSeat`. Beide Klassen stellen ein Grundgerüst für Kindsklassen bereit, welche unterschiedliches Verhalten aufweisen.

Neben den normalen Methoden in der Elternklasse besitzen diese Klassen abstrakte Methoden, welche in den Kindsklassen implementiert werden müssen.

Der `AbstractTab` beinhaltet grundlegende Daten für alle Tabs, wie das Button Panel, welches die drei Buttons für die Navigation beinhaltet. Zusätzlich wurden noch einige Methoden für das Management der Tabs implementiert, wie z.B. `reset()`, `putInContainer()` und `getComponentsFrom()`.

Die zwei abstrakten Methoden, `build()` und `update()` sorgen für unterschiedliches Verhalten der Kindsklassen. Jede Kindsklasse des `AbstractTabs` implementiert einen eigenen Aufbau des Tabs sowie ein eigenes Update des Tabs. Die Tabs haben also dank der Abstrakten Methoden einen unterschiedlichen Aufbau, sowie ein unterschiedliches Update.

Der `AbstractSeat` beinhaltet grundlegende Daten für alle Plätze und eine abstrakte `reserve()` Methode. Die Implementierung ist auch unterschiedlich: Ein `BeachChairSeat` bekommt bei der Reservierung eine Ticketnummer generiert, während der `CarSeat` ein Kennzeichen zugewiesen bekommt.

3.2. Umsetzung der Prinzipien des OO Entwurfs 1 und 4

1. Prinzip einer einzigen Verantwortung (Single Responsibility)

Gemäß den Prinzipien der objektorientierten Programmierung folgt das Programm ebenfalls dem Prinzip der Single Responsibility.

Module, sowohl der View, als auch des Models, sind nach Möglichkeit gekapselt, sodass diese jeweils lediglich eine Aufgabe ausführen. Hierzu wurden Klassen erstellt, sowie Pakete, welche die Klassen bündeln.

Die View besteht ebenfalls aus vielen einzelnen Tabs, welche alle für sich selbst verantwortlich sind und ihre eigenen `update()` und `build()` Methoden verwalten.

Die Database-Klasse beispielsweise ist alleinig für die Vorhaltung und Ausgabe von Daten zuständig. Weiterhin bündelt die `FileManager` Klasse alle Methoden zur Interaktion mit Dateien. Hier werden unter anderem Bilder für den Film geladen. Und der View zur Verfügung gestellt.

Ebenfalls in der View wurde der Aufbau der einzelnen Tabs über eine abstrakte „Tab“ Klasse implementiert. Da jedoch jeder Tab unterschiedliche Elemente aufweist, sind Kinds-Klassen für jeden Tab erstellt worden, welche sich um die Funktionen des jeweiligen Tabs kümmern.

Auch die NumberManager-Klasse ist nur für eine Funktion, das Generieren von Zahlen, zuständig.

2. *Offen für Erweiterungen, geschlossen für Änderungen* (Open-Closed-Principle)

Das Programm ist nach dem Open-Closed-Principle so aufgebaut, dass bestimmte Module einfach erweiterbar sind. Dies gilt besonders für Klassen, welche sich häufig ändernde Daten enthalten.

Es können die Listen ALL_MOVIES und ALL_CATERINGS der Database Klasse, welche alle Filme und alle Caterings beinhalten, um weitere Einträge erweitert werden ohne Code an weiteren Stellen anpassen zu müssen. Das Programm pflegt die Ergänzungen beim nächsten Start automatisch ein.

Neue Typen von Stellplätzen und Tabs können ebenfalls einfach hinzugefügt werden, da diese Klassen die AbstractSeat und AbstractTab Klasse einfach erweitern können.

Neue Tabs können ganz simpel in der View dem Array hinzugefügt werden, um diese in die Anwendung zu integrieren.

Die Vocab-Klasse beinhaltet alle wesentlichen Bezeichnungen und Dialoge des Programms. Auch hier können somit Fehlermeldungen, Warnungen, Hinweise, aber auch Bezeichnungen einfach geändert, angepasst und bedingt auch hinzugefügt werden.

Ebenfalls den Prinzipien objektorientierter Programmierung folgend, sind trotz der guten Erweiterbarkeit des Codes, sensible Datenbereiche vor unberechtigtem Zugriff und Manipulation geschützt. So sind alle Methoden und variablen gekapselt und besitzen lediglich die maximal notwendige Sichtbarkeit. Besonders die Preise für reservierbare Plätze und Snacks sind sowohl final als auch private um eine Manipulation zugunsten des Nutzers zu verhindern.

Unter anderem durch die, in den vorherigen Abschnitten Beschriebenen Verhaltensweisen des Programms, wurden, wenn möglich abstrakte Klassen/Methoden verwendet. Dadurch werden zum einen Subklassen gezwungen essenzielle Methoden entsprechend ihrem Verhalten zu implementieren, zum anderen wird so aber auch das Verwenden von Indirektionen gewährleistet.

Abstrakte Klassen wie AbstractTab oder AbstractSeat definieren durch Hooks wie build(), update() oder reserve() Erweiterungsfunktionen in ihren Subklassen.

4. GUI/Screenshots

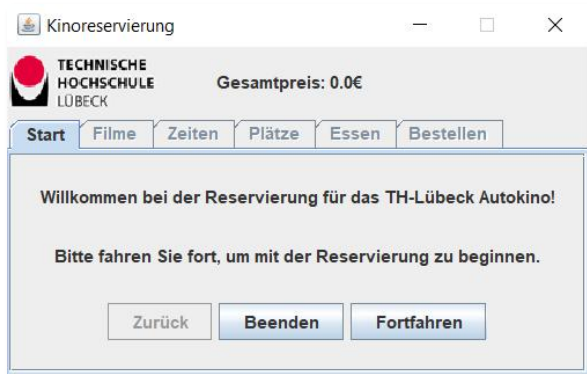


Abbildung 1: Startbildschirm der GUI

Startbildschirm

Der Nutzer startet auf dem ersten der sechs oben zu sehenden Tabs. Dort wird der Nutzer mit einer Nachricht begrüßt und aufgefordert fortzufahren. Der Nutzer kann nur Fortfahren oder das Programm beenden, jedoch nicht zurück gehen oder einen anderen Tab auswählen. Der Gesamtpreis ist oben zu sehen und bis jetzt noch 0€

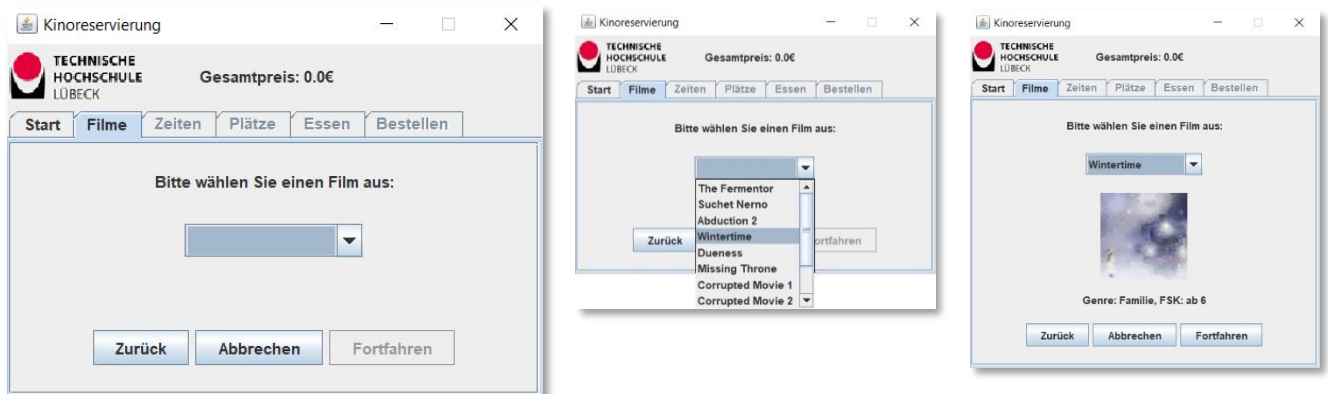
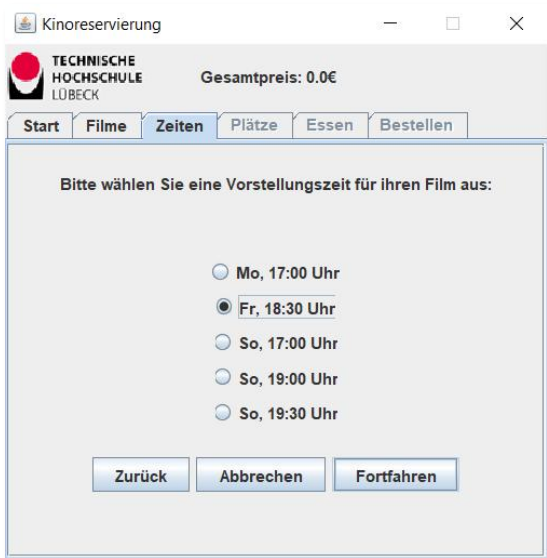


Abbildung 2: Filmauswahldialog

Filmauswahl

Klickt man auf "Fortfahren" auf dem ersten Tab (StartTab) gelangt man zum nächsten Tab. Der Nutzer kann jetzt auch zurück zum ersten Tab mit dem "Zurück" Button gelangen oder indem dieser oben in der Übersicht ausgewählt wird. Auf dem Filme Tab muss der Nutzer einen Film aus einem Dropdown auswählen, bevor er fortfahren kann. Standardmäßig ist kein Film ausgewählt. Wählt der Nutzer einen Film, werden dessen Bild und Beschreibung angezeigt. Sobald ein Film ausgewählt ist kann der Nutzer auch fortfahren.



Kinoreservierung

TECHNISCHE HOCHSCHULE LÜBECK

Gesamtpreis: 0.0€

Start Filme **Zeiten** Plätze Essen Bestellen

Bitte wählen Sie eine Vorstellungszeit für ihren Film aus:

☐ Mo, 17:00 Uhr

☒ Fr, 18:30 Uhr

☐ So, 17:00 Uhr

☐ So, 19:00 Uhr

☐ So, 19:30 Uhr

Zurück Abbrechen Fortfahren

Abbildung 3: Vorstellungsauswahlseite

Vorstellungswahl

Hier werden dem Nutzer die verfügbaren Zeiten angezeigt. Von diesem kann nur und muss eine ausgewählt werden. Daraufhin kann der Nutzer fortfahren.

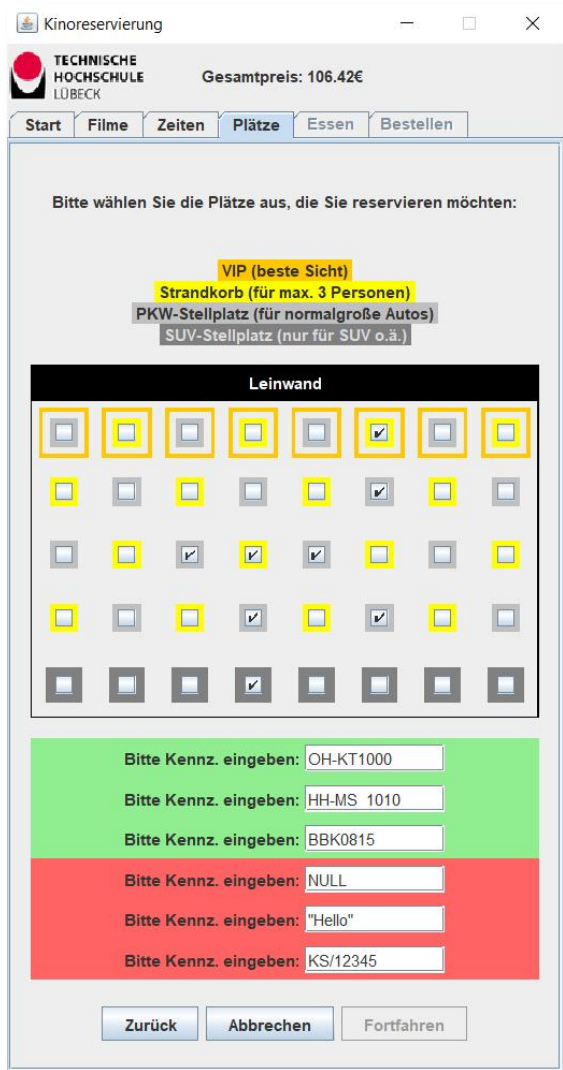
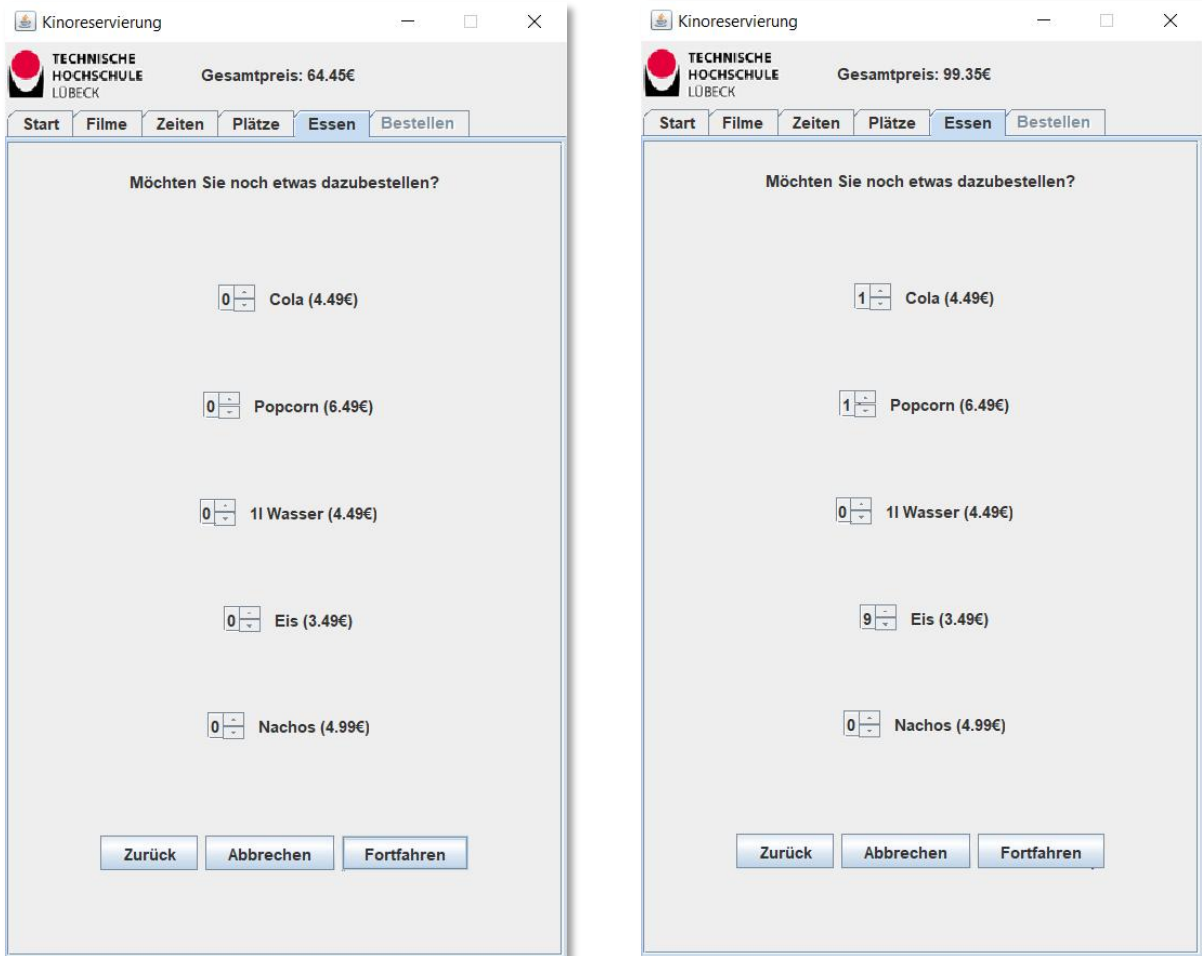


Abbildung 4: Platzwahl und Kennzeicheneingabe

Platzwahl und Kennzeicheneingabe

Hier werden dem Nutzer alle Plätze angezeigt. Gelb hinterlegte Plätze sind Strandkörbe, hellgrau hinterlegte Plätze sind PKW-Stellplätze und dunkelgrau hinterlegte, etwas größere Plätze sind SUV-Stellplätze. Die Bezeichnungen der Plätze werden dem Nutzer angezeigt, wenn dieser mit der Maus über einen Sitzplatz "hovered" (Tooltip). Der Nutzer kann die gewünschten Plätze anklicken. Wenn ein PKW oder SUV Stellplatz ausgewählt wurde, muss der Nutzer ein valides Kennzeichen eingeben. Der Preis jedes ausgewählten Platzes wird zusammengerechnet und oben in real-time angezeigt. In dem Beispiel sind zwei Strandkörbe, fünf PKW-Stellplätze und ein SUV-Platz ausgewählt. Daher muss der Nutzer auch sechs Kennzeichen eingeben. Strandkörbe benötigen kein Kennzeichen. Im Beispiel sind nur die grün hinterlegten Kennzeichen valide. Da drei der sechs benötigten Kennzeichen fehlen (nicht valide sind) kann der Nutzer auch nicht fortfahren. Sollte der Nutzer alle Kennzeichen valide eintragen, kann er fortfahren. Sollte der Nutzer keine Auto-Stellplätze ausgewählt haben (also keine Kennzeichen eingegeben haben), kann er ebenfalls fortfahren, solange min. ein Platz ausgewählt wurde.



Kinoreservierung — □ ×

TECHNISCHE HOCHSCHULE LÜBECK Gesamtpreis: 64.45€

Start Filme Zeiten Plätze **Essen** Bestellen

Möchten Sie noch etwas dazubestellen?

0 Cola (4.49€)

0 Popcorn (6.49€)

0 1l Wasser (4.49€)

0 Eis (3.49€)

0 Nachos (4.99€)

Zurück Abbrechen Fortfahren

Kinoreservierung — □ ×

TECHNISCHE HOCHSCHULE LÜBECK Gesamtpreis: 99.35€

Start Filme Zeiten Plätze **Essen** Bestellen

Möchten Sie noch etwas dazubestellen?

1 Cola (4.49€)

1 Popcorn (6.49€)

0 1l Wasser (4.49€)

9 Eis (3.49€)

0 Nachos (4.99€)

Zurück Abbrechen Fortfahren

Abbildung 5: Getränke und Snack-Auswahl

Essen und Trinken-Auswahl

Hier kann der User jede Option 0-9 mal bestellen. Dafür werden einfach die Pfeile neben der Optionen genutzt. Der Nutzer kann auch ohne ausgewählte Option (bzw. alle Optionen = 0) fortfahren, da das Catering optional ist. Der Preis jedes ausgewählten Caterings multipliziert mit seiner Anzahl wird mit den Preisen der ausgewählten Sitzplätze zusammengerechnet und oben in real-time angezeigt.

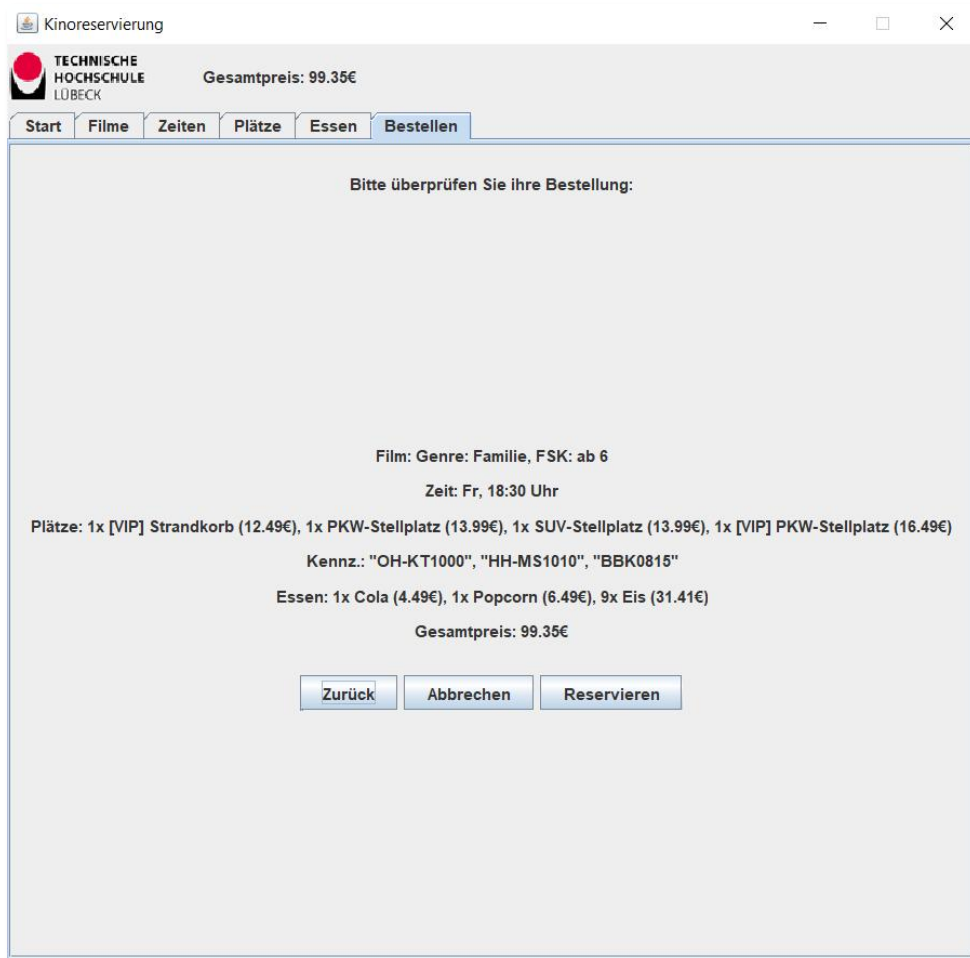


Abbildung 6: Bestellzusammenfassung

Bestellzusammenfassung

Zuletzt wird dem Nutzer eine Zusammenfassung seiner Bestellung angezeigt. Hier kann der Nutzer seine Reservierung bestätigen und abschicken. So wie bei jedem anderen Tab auch könnte der User auch mithilfe der Buttons oder der Übersicht oben zwischen den Tabs navigieren und seine Auswahl ändern.



Abbildung 7: Bestellbestätigung und Startbildschirm

Bestätigungsmeldung und Rückkehr zum Startscreen

Sollte der Nutzer reserviert haben, werden ihm zwei Dank-Nachrichten angezeigt. Unter diesen Nachrichten sind auch alle Tickets für ausgewählte Strandkörbe zu finden. Sollten keine Strandkörbe ausgewählt worden sein, bekommt der Nutzer auch kein Ticket angezeigt, bei 3 Strandkörben sind es 3 Tickets, usw.

Die Tickets für jeden Strandkorb können auch in der Bestellbestätigung eingesehen werden. (siehe unten)

```
orders > Bestellung-31066193.txt
1  Vielen Dank, dass Sie unseren Service nutzen!
2  Im Folgenden finden Sie die Informationen zu ihrer Reservierung; Bestellnummer: 31066193
3
4  Film: Wintertime
5
6  Zeit: Fr, 18:30 Uhr
7
8  Plätze:
9  - [VIP] Strandkorb (12.49€), Ihr Ticket: 7041-6783
10 - PKW-Stellplatz (13.99€), Kennz.: "QH-KT1000"
11 - SUV-Stellplatz (13.99€), Kennz.: "HH-MS1010"
12 - [VIP] PKW-Stellplatz (16.49€), Kennz.: "BBK0815"
13
14 Essen:
15 1x Cola (4.49€)
16 1x Popcorn (6.49€)
17 9x Eis (31.41€)
18
19 Gesamtpreis: 99.35€
```

Abbildung 8: Gespeicherte Bestellung

Bestellspeicherung

Hier ist eine beispielhafte Bestellung als Textdokument zu sehen. In ihr sind alle wichtigen Informationen zu finden, sowie das Ticket jedes Strandkorbes und das Kennzeichen jedes Autostellplatzes.

Spezialfälle

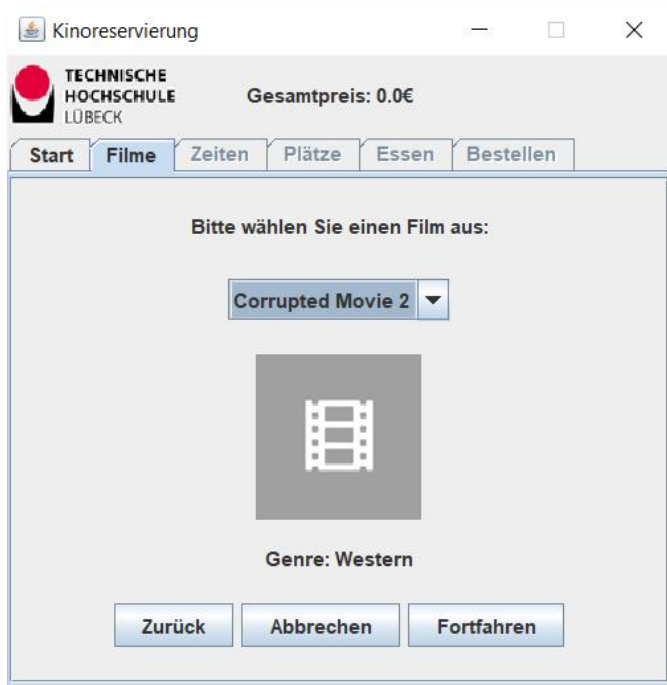


Abbildung 9: Film ohne Vorschaubild

Der Film hat kein Bild, wenn der Pfad gleich null ist oder der Pfad für das Bild nicht existiert (also wenn der FileManager das Bild nicht laden kann). Ebenfalls fehlt diesem Film eine FSK Angabe. Diese wird nicht mit ausgegeben.

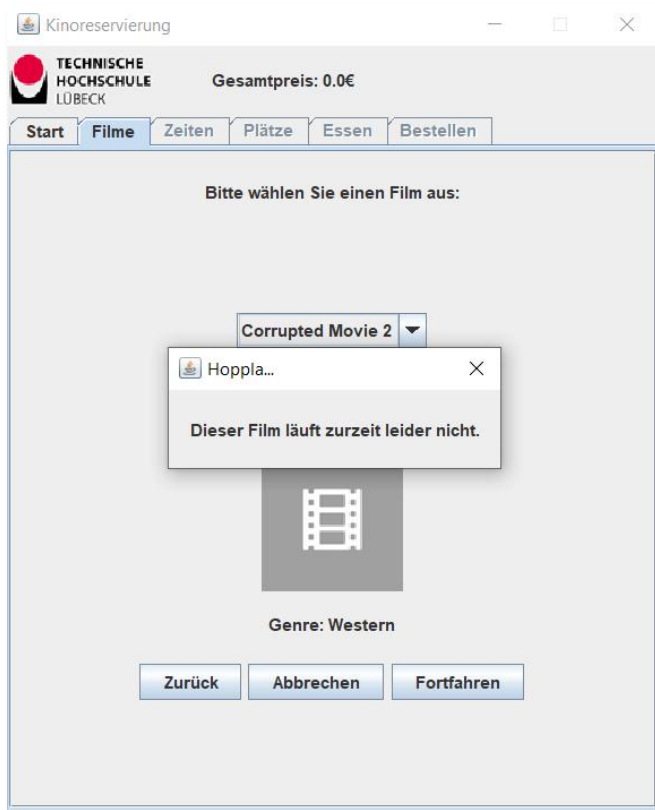


Abbildung 10: Film ohne Zeiten

Der Film hat keine Vorstellungszeiten, wenn diese gleich null sind. Dem Nutzer wird dann stattdessen eine entsprechende Nachricht angezeigt, anstelle auf den nächsten Tab zu switchen.

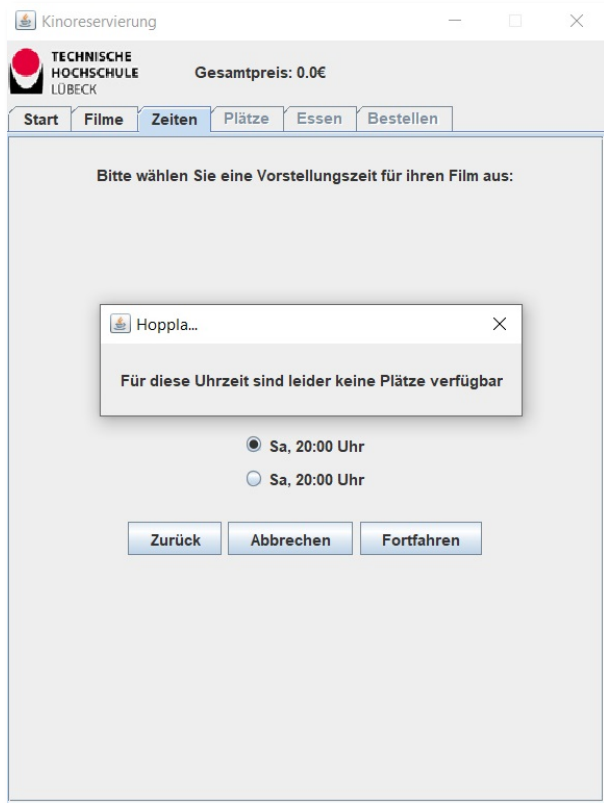
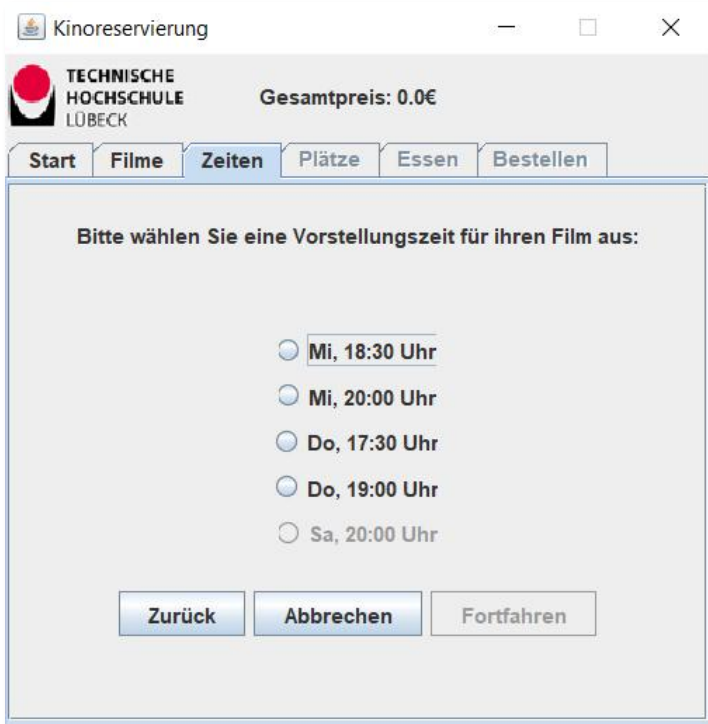


Abbildung 11: Fehlende Platzangaben eines Filmes

Ist beim Einpflegen eines Filmes in die Datenbank diesem keine Veranstaltungsfläche zugewiesen, also das Array der Seats "null", wird eine *Exception* geworfen und beim Versuch im Programm fortzuschreiten eine Fehlermeldung ausgegeben, welche darauf hinweist, dass für diesen Zeit Slot keine Sitzplatzauswahl verfügbar ist.



Kinoreservierung

**TECHNISCHE
HOCHSCHULE
LÜBECK** Gesamtpreis: 0.0€

Start Filme **Zeiten** Plätze Essen Bestellen

Bitte wählen Sie eine Vorstellungszeit für ihren Film aus:

☐ Mi, 18:30 Uhr

☐ Mi, 20:00 Uhr

☐ Do, 17:30 Uhr

☐ Do, 19:00 Uhr

☐ Sa, 20:00 Uhr

Zurück Abbrechen Fortfahren

Abbildung 12: Ausverkaufter Film

Sind zu einer Vorstellungszeit alle Plätze reserviert, ist die Vorstellung also ausverkauft, so ist dieser Zeit Slot ausgegraut und nicht auswählbar.

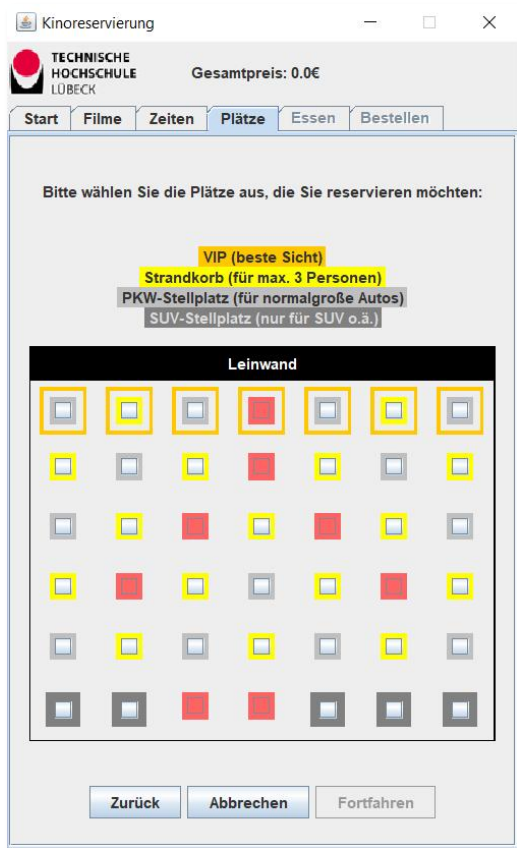


Abbildung 13: Reservierungsmarkierungen in der Platzwahl

Sind bei einer Reservierung Plätze bereits belegt, werden diese rot hervorgehoben. Ebenfalls passt sich der Tooltip der Plätze entsprechend an. Diese Plätze kann man auch nicht mehr auswählen.

5. Zusammenfassung

5.1. Aufgabenteilung

Besonders durch Corona war das gemeinsame Arbeiten an Projekten schwierig. Um dies zu einem gewissen Maß zu kompensieren, haben wir neben der Kommunikation über Discord, welche Text-, Sprach- und Videochat, sowie Screenshare ermöglichte auch Github genutzt, um den Code effektiv zu bearbeiten und zu verwalten.

5.2. Ausblick

- Charmant wäre eine Ausgabe von Platzkarten für die Gäste gewesen. Da dies jedoch nicht zwingend notwendig ist, wurde dieses Feature aus Zeitgründen verworfen.
- Sollte für Essen/Trinken keine Option ausgewählt sein, ist die Map = null, dieser Zustand wird genauso behandelt wie, wenn jeder Key (Catering) einen Value (Menge/Amount) von 0 hätte.
- Der Fortfahren Button erzwingt das Resetten (und builden) des nächsten Tabs. Dies ist normalerweise nicht nötig, wenn der User beispielsweise gar keine Änderungen vorgenommen hat, sondern nur einige Tabs zurückgesprungen ist. Sollte der User jetzt wieder zurück, zu einem höher gelegenen Tab gelangen wollen, müsste dieser alle Daten der Tabs, die oberhalb des Tabs mit dem geklickten Fortfahren Button liegen, neu eingeben. Dies ist der Architektur des Programms der build()-Methode geschuldet.
- Das Eintippen von Input in Textfelder für Kennzeichen speichert den Input des Users bei jedem neuen Zeichen erneut. Das Speichern der Kennzeichen müsste eigentlich aber grundsätzlich nur geschehen, wenn die Bestellung angezeigt/abgeschickt wird. Das gleiche gilt für die Auswahl der Caterings. Dies ist der Architektur des Programms, genauer der update()-Methode des Tabs geschuldet, da der SeatingTab diese Methode nutzt, um die Textfelder mit einem grünen oder roten Indikator zu hinterlegen. Hierfür muss der Input zeichenweise geprüft werden.
- Im Nachhinein stellte sich heraus, dass es für den Nutzer angenehmer wäre, würden die Navigationselemente eine fixe Position am Fensterboden besitzen und das mittlere Panel scrollbar sein. Dies würde variable Fenstergrößen, besonders bei großen Reservierungen vermeiden und zur Übersichtlichkeit beitragen.
- Die Bestellübersicht hätte, besonders bei Verwendung eines ScrollPane, vertikal ausgerichtet werden sollen, sodass bei großen Reservierungen durch diese gescrollt werden kann.