ODEON
FANATICAL ABOUT FILM

# Odeon Cinema Ticket-management System

Student ID: 17001706

## Table of Contents

*Note: the application is preloaded with demo-data. When viewing the reports, select Dec. 2017 to view historical reports with ratings, and select Feb. 2018 to view ticket sales for shows in the future.*

## Introduction

The Odeon cinema has requested a software system to help them manage ticket allocation and payment processing. The software will be used by the Odeon staff members to gather sales data about each film, and produce monthly reports with ticket sales, customer ratings and total income generated.

## Requirements

### (a) Client-defined requirements

The cinema has provided a design brief outlining the functionality that they require.

- Shows can have different pricing, depending on the time of day
- Customers can pay using cash or credit card, and their payments are recorded in the system
- When purchasing a ticket, the customer selects the show-time they will attend
- Tickets can be transferred from one show to another, provided that the show has not already taken place
- After the show, customers can review and rate the film from 1 to 5, and this data is recorded in the system
- The software will display monthly reports for each film. The two reports are
    - Number of tickets sold, with average rating per film
    - Income generated per film for the previous month
- There is no need for the system to store data in persistent storage

### (b) Recommended Features

Based on the context, a further set of recommended features will be included, as this will improve the usability of the system. In a real-world setting, these recommended features would first be presented to the client for discussion and approval.

- Tickets should be booked for specific seats, with the available seats displayed on screen in a seating grid which can be clicked to select seats
- Ticket-transfers should also use a seating-grid
- The system should have an intuitive GUI, so that the training and induction process for new staff is kept to a minimum
- The monthly reports should be available for all months and years where there is data collected, rather than only for the previous month. This allows the cinema to track trends over time, and will help them in their strategic planning processes

## Assumptions

### (a) Security and Accessibility

The system will only be accessible to trained staff. The computer running the application will be stored in a secure location. While the revenue reports contain sensitive data, the system is only accessible to users who have the privileges to view sales reports and financial data. Therefore, no login or role-based authorization and authentication features are required.

### (b) Persistent Data Storage

The software should be designed on the basis that, in the future, it will incorporate a persistent storage mechanism

### (c) Payment processing

All credit-card payments take place with a handheld Point-Of-Sales terminal, which displays a payment reference number after completion. When logging credit card payments, the only data that needs to be stored in the system about this transaction is the payment reference number

### (d) Customer records

When customers purchase their first ticket, they give their name and are issued with a membership card containing a customer Id number. Each time they return, the customer gives this customerId number. If the customer forgets their number, the user can retrieve the customerId from the customer index.

### (e) Tickets

Tickets are physically printed, displaying the seat number and TicketId. This ticket id is required whenever a customer transfers the ticket to a different show-time, and when the customer gives a review and a rating for the film.

When a ticket is transferred to a more expensive seat, the customer incurs a surcharge. When the ticket is transferred to a seat of the same price, or cheaper, no payment transaction takes place and no refund is issued

### (f) Small Data-set

For this implementation of the system, the Gui is designed to be useable with a relatively small set of films, shows, bookings and customers. Therefore, the lists in the index-pages do not need to be filtered.

### (g) Reporting Dates

Reports are for specific time-frames of one month. This gives us the option to use either the date of payment, or the date of the ticket as the basis for the reporting. Based on the requirements for the reports, it is assumed that the reports are for the month that the film took place, rather than when the payment took place. Note that this assumption creates a discrepancy between the revenue generated as per the financial accounts, and the revenue generated as per the ticketing system reports. In a real-life situation, this option would be discussed with the client, as it requires a deeper understand of the types of decisions that the reports will need to inform.

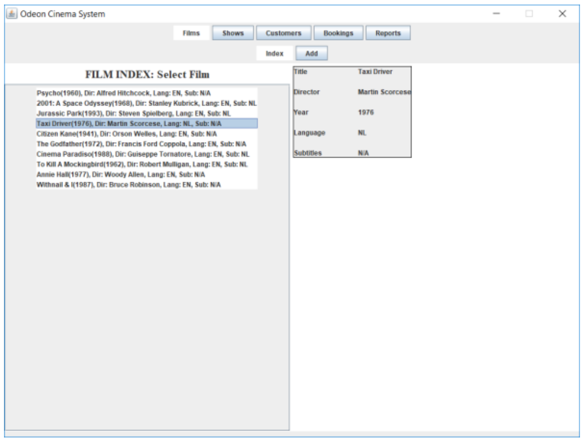
### (h) Reviews

The reviews are limited to 20 characters, so that the cinema can get short, punchy reviews from customers without adding too much data-entry overhead for their staff.

The reports use customer-generated film-ratings, but do not use the review text. In this implementation, the reviews are not prominently displayed, as this is not a client-requirement. Reviews can be viewed through the booking index. In a real-life situation, the collection of this data would be queried, as it is adding overhead to the staff-workload and does not appear to be a key feature of the system.

## User Interface

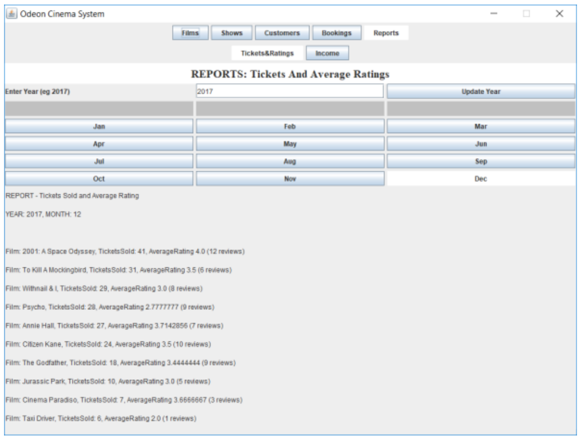
The software uses a graphical user interface to help the user navigate through the data and carry out their tasks. The GUI has several key features that make it intuitive for the user.



FILM INDEX: Select Film

Psycho(1960), Dir: Alfred Hitchcock, Lang: EN, Sub: N/A
2001: A Space Odyssey(1968), Dir: Stanley Kubrick, Lang: EN, Sub: NL
Jurassic Park(1993), Dir: Steven Spielberg, Lang: EN, Sub: NL
Taxi Driver(1976), Dir: Martin Scorcese, Lang: NL, Sub: N/A
Citizen Kane(1941), Dir: Orson Welles, Lang: EN, Sub: N/A
The Godfather(1972), Dir: Francis Ford Coppola, Lang: EN, Sub: N/A
Cinema Paradiso(1988), Dir: Guiseppe Tornatore, Lang: EN, Sub: NL
To Kill A Mockingbird(1962), Dir: Robert Mulligan, Lang: EN, Sub: NL
Annie Hall(1977), Dir: Woody Allen, Lang: EN, Sub: N/A
Withnail & I(1987), Dir: Bruce Robinson, Lang: EN, Sub: N/A

| | |
|---|---|
| Title | Taxi Driver |
| Director | Martin Scorcese |
| Year | 1976 |
| Language | NL |
| Subtitles | N/A |



REPORTS: Tickets And Average Ratings

Enter Year (eg 2017) 2017

```
REPORT - Tickets Sold and Average Rating

YEAR: 2017, MONTH: 12


Film: 2001: A Space Odyssey, TicketsSold: 41, AverageRating 4.0 (12 reviews)

Film: To Kill A Mockingbird, TicketsSold: 31, AverageRating 3.5 (6 reviews)

Film: Withnail & I, TicketsSold: 29, AverageRating 3.0 (8 reviews)

Film: Psycho, TicketsSold: 28, AverageRating 2.7777777 (9 reviews)

Film: Annie Hall, TicketsSold: 27, AverageRating 3.7142856 (7 reviews)

Film: Citizen Kane, TicketsSold: 24, AverageRating 3.5 (10 reviews)

Film: The Godfather, TicketsSold: 18, AverageRating 3.4444444 (9 reviews)

Film: Jurassic Park, TicketsSold: 10, AverageRating 3.0 (5 reviews)

Film: Cinema Paradiso, TicketsSold: 7, AverageRating 3.6666667 (3 reviews)

Film: Taxi Driver, TicketsSold: 6, AverageRating 2.0 (1 reviews)
```

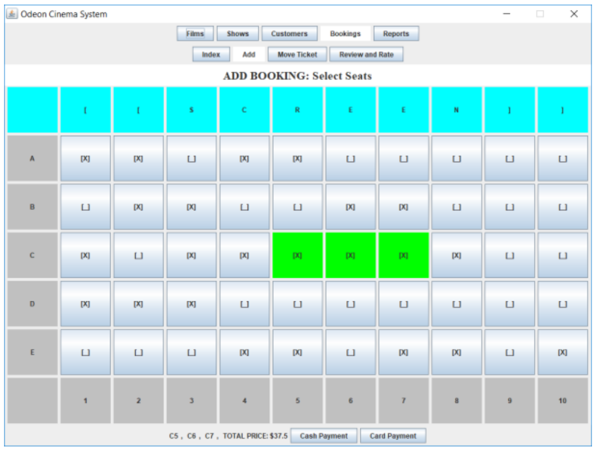
*Films Index*

*Report Generator*

### (a) Error handling

Given that the users are staff-members, error prevention measure are in place to guide user when invalid data is entered.

### (b) Seating Grid



ADD BOOKING: Select Seats

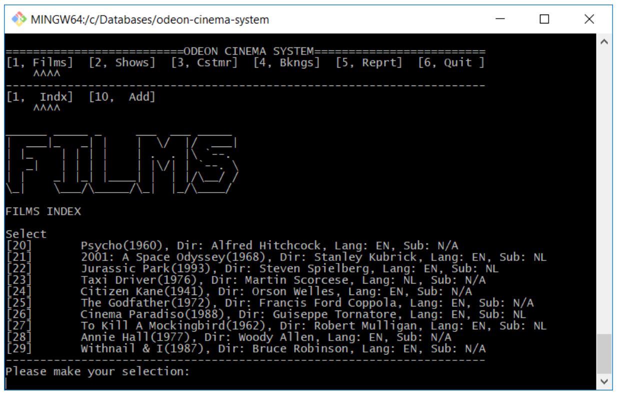
| | [ | [ | S | C | R | E | E | N | ] | ] |
|---|---|---|---|---|---|---|---|---|---|---|
| A | [X] | [X] | [ ] | [X] | [X] | [ ] | [ ] | [ ] | [ ] | [ ] |
| B | [ ] | [X] | [X] | [ ] | [ ] | [X] | [X] | [ ] | [ ] | [ ] |
| C | [X] | [ ] | [X] | [X] | [X] | [X] | [X] | [X] | [ ] | [ ] |
| D | [X] | [X] | [X] | [ ] | [ ] | [ ] | [ ] | [ ] | [ ] | [ ] |
| E | [ ] | [ ] | [ ] | [X] | [X] | [ ] | [X] | [X] | [ ] | [X] |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

C5 , C6 , C7 , TOTAL PRICE: $37.5

*Seating Grid with three seats selected*

The seating grid is a graphical interface with 50 buttons, one for each seat in the screen. When adding or transferring tickets from one show to another, the user selects the seats by clicking on the buttons in this grid. The seats are temporarily stored in the system, and can be deselected as required, until the booking or transfer is finalised. For the Booker, multiple seats can be selected and deselected. Selected seats are highlighted with a green background. For the Transferer, a single new seat can be selected. If the selected show is the same as the original show for the ticket, then the original seat is displayed as orange, indicating that it cannot be selected.

### (c) CLI



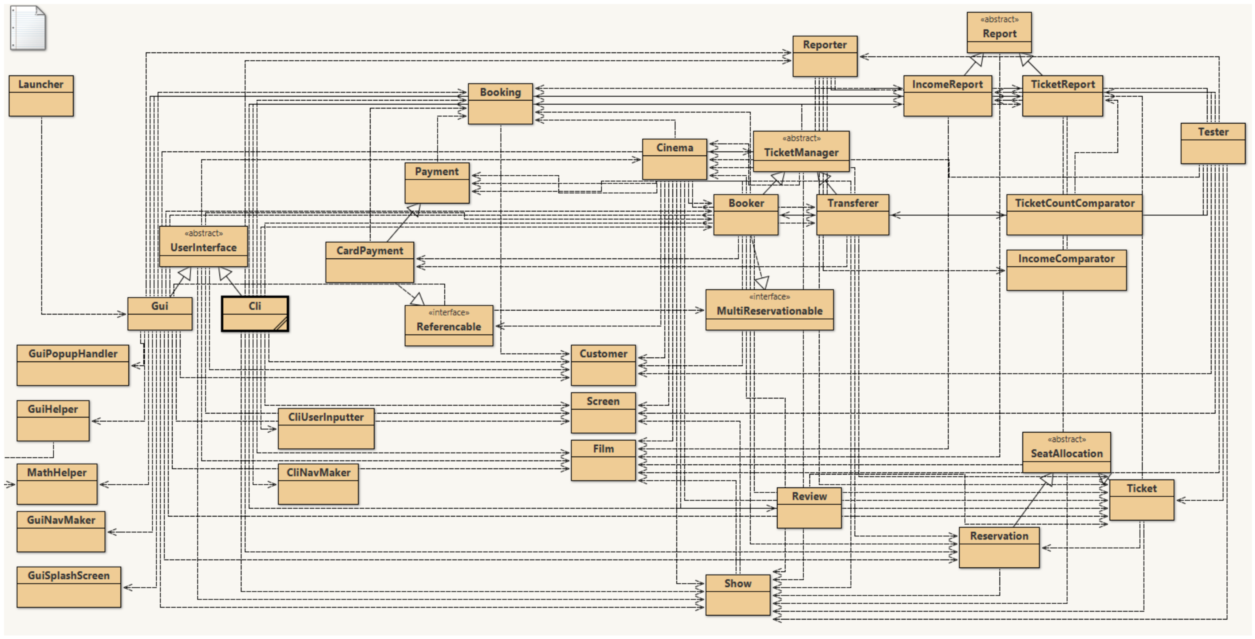
MINGW64:/c/Databases/odeon-cinema-system

```
========================ODEON CINEMA SYSTEM========================
[1, Films]  [2, Shows]  [3, Cstmr]  [4, Bkngs]  [5, Reprt]  [6, Quit ]
    ^^^^
-------------------------------------------------------------------
[1,  Indx]  [10,  Add]
    ^^^^

FILMS INDEX

Select
[20]       Psycho(1960), Dir: Alfred Hitchcock, Lang: EN, Sub: N/A
[21]       2001: A Space Odyssey(1968), Dir: Stanley Kubrick, Lang: EN, Sub: NL
[22]       Jurassic Park(1993), Dir: Steven Spielberg, Lang: EN, Sub: NL
[23]       Taxi Driver(1976), Dir: Martin Scorcese, Lang: NL, Sub: N/A
[24]       Citizen Kane(1941), Dir: Orson Welles, Lang: EN, Sub: N/A
[25]       The Godfather(1972), Dir: Francis Ford Coppola, Lang: EN, Sub: N/A
[26]       Cinema Paradiso(1988), Dir: Guiseppe Tornatore, Lang: EN, Sub: NL
[27]       To Kill A Mockingbird(1962), Dir: Robert Mulligan, Lang: EN, Sub: NL
[28]       Annie Hall(1977), Dir: Woody Allen, Lang: EN, Sub: N/A
[29]       Withnail & I(1987), Dir: Bruce Robinson, Lang: EN, Sub: N/A
-------------------------------------------------------------------
Please make your selection:
```

*Command-line interface*

Prior to the development of the GUI, a Command-line interface was developed. See Appendix Item for further details about the CLI.

## Class Design



*Abridged UML Class Diagram – See Appendix Item 2 for detailed UML class diagram*

- The Booker and Transferer classes manage the booking of new tickets and the transfer of tickets after purchase
- Show class stores the scheduling of a Film on a Screen
- Ticket object is a purchased ticket. Reservation is a temporary object that is created while the booking and transfer process is taking place. Booker implement the MultiReservationable interface, as it store a temporary array of multiple reservation objects.

- Each Film object relates to the physical copy of a film. If the cinema shows two copies of the same film simultaneously, the user adds two instances of the film to the system and assigned one instances to each show.
- The Reporter class gathers together a set of Reports, one report object per film, and assembles them into an ordered list, using the Comparator classes.
- The Gui and Cli classes each have a set of helper and utility methods
- The Cinema class is the main implementation class. It stores hashmaps of films, screens, shows, customers, tickets and all other objects.
- Payments are stored in either the Payment class or the CardPayment class. CardPayment implements the Referencable interface, with a getReferenceNumber method

## Data Structures and Implementation

### (a) HashMaps

All objects are stored in a hashmap in the cinema class, with the objectId set as the hashmap key. Hashmaps allow for easy querying based on objectId. The assumption is made that the system will be migrated to persistent data storage in the future. Therefore, the hashmaps are all stored centrally to make it easier to refactor and migrate the system.

## Algorithms

### (a) Single TicketManager object

Only one Booker and one Transferer can be allowed to run as one time. As the Booker and Transferer objects are stored by the Cinema class, the Singleton pattern cannot be implemented in the Booker/Transferer classes. However, the getNewBooker() and getNewTransferer() methods in the cinema class remove any Booker or Transferer object present and replace it with a new object, ensuring that only one of each object can exists.

### (b) List Comparator

The Reporter generates a list of Report objects. This list is ordered by the compare methods in the TicketCountComparator and IncomeComparator classes.

### (c) Reports and Ratings

The TicketReport objects contain the average rating for the film. Many ticket-holders will not submit a review and rating. Therefore, to calculate the average rating, a count of the ratings is kept, and the sum of the ratings is divided by this ratings-count.

## Data Validation

### (a) User Input validation

All user input is validated by the Gui.class. String lengths are limited, and numeric values are tested in try/catch blocks before the object is created. The values for the Calendar date objects are tested for validity before creating the Calendar object. If the data is invalid, the process is interrupted and a dialogue-box is displayed to guide the user towards the correct input.

### (b) Show-creation input validation

Shows cannot be created if the Film or the Screen has already been allocated to a show at that time. This implementation uses the start-time of the show to determine if the film and screen are available. In a future implementation, this validation can be improved by storing the duration of the film, and using this duration to calculate a blackout period when the film and screen are unavailable.

## Testing

The Tester object tests the system by calling runAllTests().

### (a) Unit Testing

If a method carries out calculation to manipulate data object, then the method is black-box tested. If the method creates new objects, the Tester checks whether the objects have been successfully created. If calculations take place, the Tester focusses testing on the boundary-values.

### (b) System and Integration Testing

The Tester runs tests on processes that use multiple methods, and also tests full use-cases.

Three examples of tests in the test-suite:

1. testBookingAndCardPaymentProcess() – This test runs a full booking process, from show selection, to seat selection, to card-payment processing and the recording of the transaction reference number.
2. testTransfererCashPaymentHigherPrice() – This test runs a full transfer process, where a ticket is transferred to a higher priced seat. The ticket is generated, a new seat selected that is more expensive than the ticket price, a cash payment is processed for the difference and the ticket price is updated to reflect the new price.
3. testGetOrderedReportLists() – This test runs the reporting process. It generates a Reporter object, which creates two report-lists and sorts them by ticket sales or income generation.

## Version Control

The application was developed using Git version control. The workflow has a feature-branch where new features are created. When the feature is ready, this is merged into the develop branch, where the system is tested to ensure it functions as expected. The develop branch is pushed to the remote repository's develop branch. When a set of new features have been merged to the develop branch, it is merged into the master branch and pushed to the repository.

## Conclusion

The system meets the needs of the client as described in the brief, and has added functionality which makes the application more user-friendly. The easy-to-use GUI will reduce time spent training new staff, and will help to ensure that the data stored in the system is reliable, and the reports generated can be confidently used for strategic planning purposes.

### Further features that would improve the system

Several features can be added to future implementations to improve the usability of the system.

The Gui is designed with the assumption that the number of films, shows and customers will remain relatively small. However, as the show- and film-lists grow, an archiving feature will need to be added to the Film and Show objects. Otherwise, the index lists will quickly become cluttered with redundant data.

The customer list will also grow quickly. As it is inappropriate to archive a customer, a search-feature or filter-by-initial feature should be added.

## Self-reflection

Before taking this course, I had very little experience developing in Java, and this was the first project I ever created using the language. I had never used the Java Swing library, so there was a steep learning curve involved in order to develop an appropriate user-interface. There are several design decision that I made early in the process which I learned are not best practice, and a future implementation of the system would be designed differently in these areas.

### (a) Class sizes

The Cinema, Gui and Cli classes each contain too many methods. I tried to follow best practice by creating methods that fit within the computer monitor, and break up functionality into multiple methods if possible.

During the development of the system, I refactored the Cinema, Cli and Gui classes in order to reduce the number of methods that they contain. During the process, I refactored the Cinema and Cli classes. As this was my first project using the Java Swing library, the ActionListeners and MouseListeners made it more difficult to break up the class. Therefore, the number of methods in the Gui class is high, and has low cohesion. In a future implementation, I will be more familiar with the Swing library, and better approaches to structuring a Gui class file

### (b) The Law of Demeter

My design for the TicketManager classes, (Booker and Transferer) appear to be breaking the law of Demeter. These TicketManager objects carry out their tasks by using methods in the Cinema class, which in turn carries out methods in the other object classes. The situation arises as a result of the structure of the Cinema class, where all hashmaps are located, and where many of the object-manipulation methods are stored. I realised towards the end of the project that this was bad practice.

### (c) Exception Handling

As the project progressed, I became more familiar with the best practices for exception-handling. All data inputted through the Gui is validated before reaching the object class. While the system is robust, the project has highlighted the need for me to become familiar with correct exception-handling measures in Java.

### (d) User Interface Design

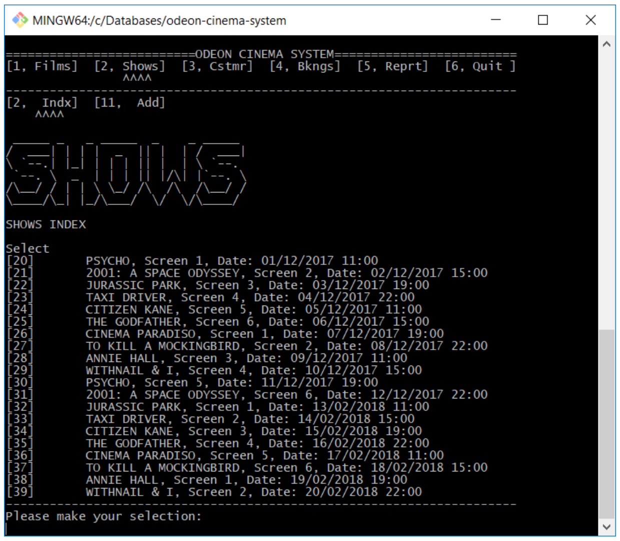
When I started work on the project, I developed it with a command-line interface in mind. Therefore, the exceptions and user prompts were all created by System.out calls, printed to console. After the Cli class had been developed, I decided to try to build a Gui using the Swing library. I ran into an issue at this point, as all notifications, including exceptions and the seatingGrid, were being printed to the console. In order for the Gui class to be able to use the same object classes as the Cli classes, I had to restructure how notifications are created and displayed. This was a big task, and in the future I will ensure that the objects are UI neutral, so that they can be used for GUI and CLI.

### (e) Version Control

This was my first time using Git as the version control for a project. As the project progressed, I became more familiar with different workflows, and I learned to used the master/develop/feature-branch workflow that is widely used in team-development. This workflow helped me break the project into features, and I will use it in my future projects.

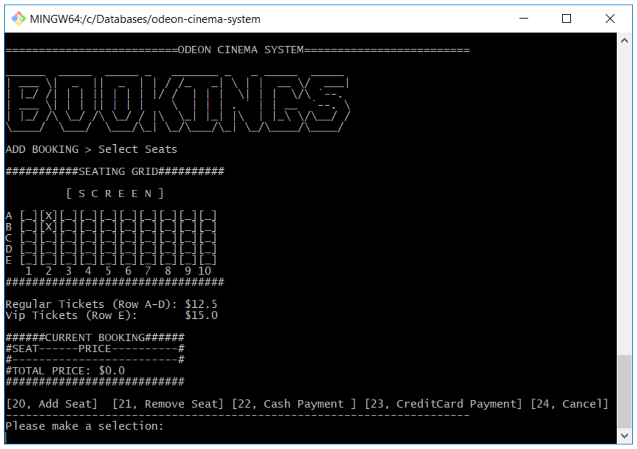
## Appendix

### Appendix Item 1: Command-line Interface



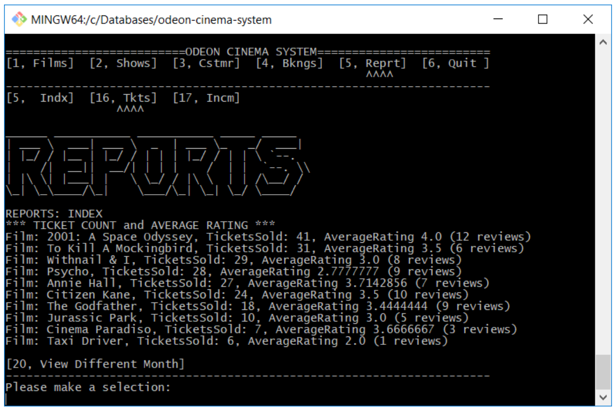
```
MINGW64:/c/Databases/odeon-cinema-system

=========================ODEON CINEMA SYSTEM=========================
[1, Films]  [2, Shows]  [3, Cstmr]  [4, Bkngs]  [5, Reprt]  [6, Quit ]
                ^^^^
----------------------------------------------------------------------
[2,  Indx]  [11,  Add]
   ^^^^

SHOWS INDEX

Select
[20]        PSYCHO, Screen 1, Date: 01/12/2017 11:00
[21]        2001: A SPACE ODYSSEY, Screen 2, Date: 02/12/2017 15:00
[22]        JURASSIC PARK, Screen 3, Date: 03/12/2017 19:00
[23]        TAXI DRIVER, Screen 4, Date: 04/12/2017 22:00
[24]        CITIZEN KANE, Screen 5, Date: 05/12/2017 11:00
[25]        THE GODFATHER, Screen 6, Date: 06/12/2017 15:00
[26]        CINEMA PARADISO, Screen 1, Date: 07/12/2017 19:00
[27]        TO KILL A MOCKINGBIRD, Screen 2, Date: 08/12/2017 22:00
[28]        ANNIE HALL, Screen 3, Date: 09/12/2017 11:00
[29]        WITHNAIL & I, Screen 4, Date: 10/12/2017 15:00
[30]        PSYCHO, Screen 5, Date: 11/12/2017 19:00
[31]        2001: A SPACE ODYSSEY, Screen 6, Date: 12/12/2017 22:00
[32]        JURASSIC PARK, Screen 1, Date: 13/02/2018 11:00
[33]        TAXI DRIVER, Screen 2, Date: 14/02/2018 15:00
[34]        CITIZEN KANE, Screen 3, Date: 15/02/2018 19:00
[35]        THE GODFATHER, Screen 4, Date: 16/02/2018 22:00
[36]        CINEMA PARADISO, Screen 5, Date: 17/02/2018 11:00
[37]        TO KILL A MOCKINGBIRD, Screen 6, Date: 18/02/2018 15:00
[38]        ANNIE HALL, Screen 1, Date: 19/02/2018 19:00
[39]        WITHNAIL & I, Screen 2, Date: 20/02/2018 22:00
----------------------------------------------------------------------
Please make your selection:
```

*Shows > Index with Navigation options as integers*



```
MINGW64:/c/Databases/odeon-cinema-system

=========================ODEON CINEMA SYSTEM=========================

ADD BOOKING > Select Seats

##########SEATING GRID##########

         [ S C R E E N ]

A [_][X][_][_][_][_][_][_][_][_]
B [_][X][_][_][_][_][_][_][_][_]
C [_][_][_][_][_][_][_][_][_][_]
D [_][_][_][_][_][_][_][_][_][_]
E [_][_][_][_][_][_][_][_][_][_]
   1  2  3  4  5  6  7  8  9 10
################################

Regular Tickets (Row A-D): $12.5
Vip Tickets (Row E):       $15.0

######CURRENT BOOKING######
#SEAT------PRICE----------#
#-------------------------#
#TOTAL PRICE: $0.0
##########################

[20, Add Seat]  [21, Remove Seat] [22, Cash Payment ] [23, CreditCard Payment] [24, Cancel]
----------------------------------------------------------------------
Please make a selection:
```

*Seat Selection feature for ticket booking*



```
MINGW64:/c/Databases/odeon-cinema-system

=========================ODEON CINEMA SYSTEM=========================
[1, Films]  [2, Shows]  [3, Cstmr]  [4, Bkngs]  [5, Reprt]  [6, Quit ]
                                                  ^^^^
----------------------------------------------------------------------
[5,  Indx]  [16, Tkts]  [17, Incm]
                ^^^^

REPORTS: INDEX
*** TICKET COUNT and AVERAGE RATING ***
Film: 2001: A Space Odyssey, TicketsSold: 41, AverageRating 4.0 (12 reviews)
Film: To Kill A Mockingbird, TicketsSold: 31, AverageRating 3.5 (6 reviews)
Film: Withnail & I, TicketsSold: 29, AverageRating 3.0 (8 reviews)
Film: Psycho, TicketsSold: 28, AverageRating 2.7777777 (9 reviews)
Film: Annie Hall, TicketsSold: 27, AverageRating 3.7142856 (7 reviews)
Film: Citizen Kane, TicketsSold: 24, AverageRating 3.5 (10 reviews)
Film: The Godfather, TicketsSold: 18, AverageRating 3.4444444 (9 reviews)
Film: Jurassic Park, TicketsSold: 10, AverageRating 3.0 (5 reviews)
Film: Cinema Paradiso, TicketsSold: 7, AverageRating 3.6666667 (3 reviews)
Film: Taxi Driver, TicketsSold: 6, AverageRating 2.0 (1 reviews)

[20, View Different Month]
----------------------------------------------------------------------
Please make a selection:
```

*Ticket Report list with tickets sold and average rating*

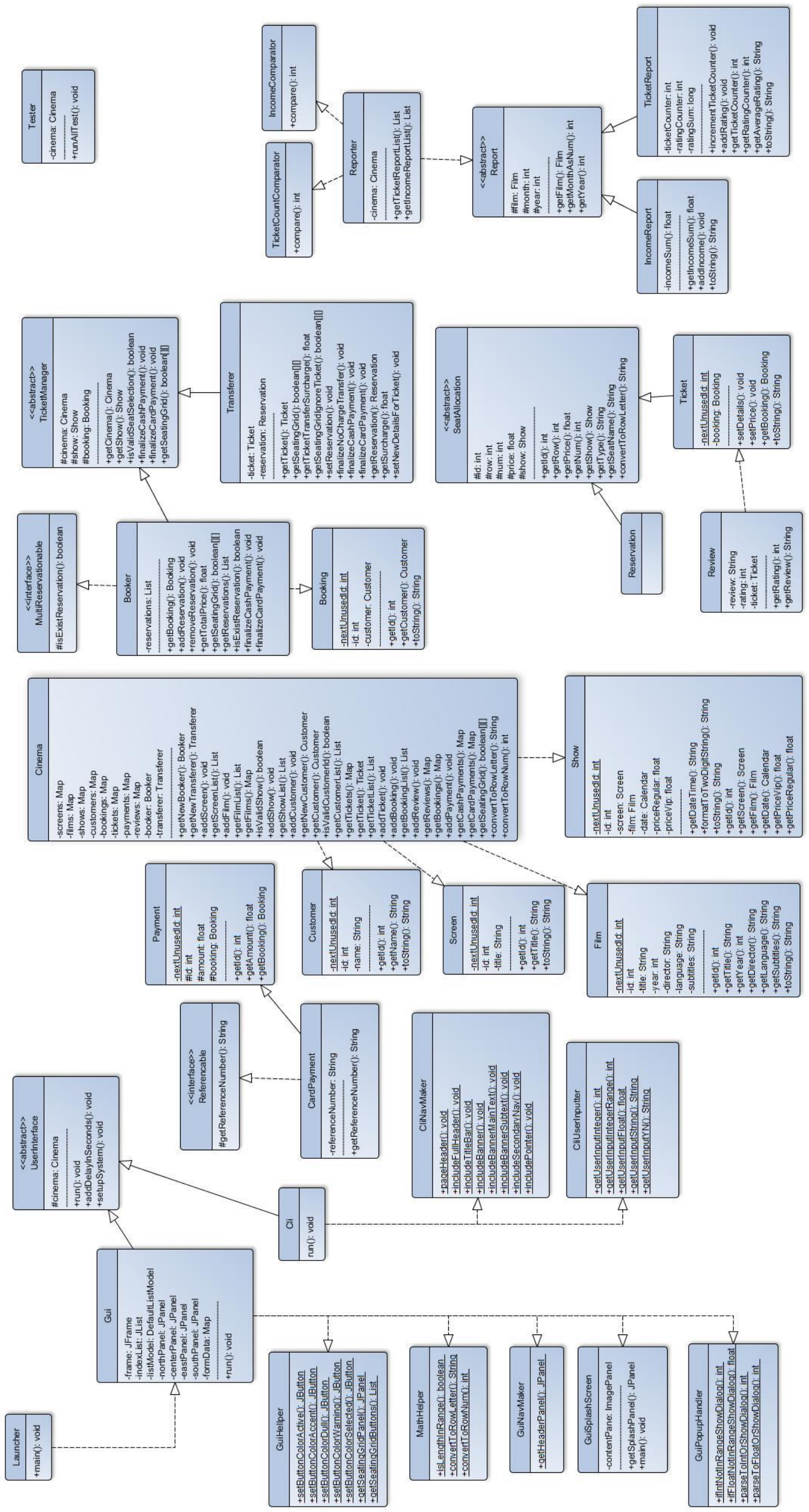
When first designed, the application was developed with a command-line interface in mind. The command-line interface is located across three files, Cli.class, CliUserInputter.class and CliNavMaker.class. This CLI is a graphic user interface that runs in the console.

Each page is represented by a method, displaying the element in the console using System.out. Users navigate the application by entering numbers. The integers from 1 to 19 are designated as pageId navigation codes. Whenever a user enters one of these values, the application redirects to the relevant page. Number input above 19 are page-function codes, which carry out the tasks specific to the current page.

The application has all the same features as the Gui, including the seatingGrid where the user can view available seats, book seats and transfer tickets.

To load the Odeon Cinema System in CLI-mode, launch the odeon-cinema-system-cli.jar file from your console or terminal. Alternatively, you can edit the Launcher.class file and compile.

## Appendix Item 2: Detailed UML Diagram



## References and Bibliography

### References

Codethulu – Stackoverflow User (2009) Creating a JButton with background color https://stackoverflow.com/questions/1065691/how-to-set-the-background-color-of-a-jbutton-on-the-mac-os Accessed on 26-DEC-2017, StackOverflow.com.

Dyndrilliac – Stackoverflow User (2013) Clear Console Screen on MacOS and Windows https://stackoverflow.com/questions/2979383/java-clear-the-console Accessed: 18-DEC-2017, Stackoverflow.com.

Java Made Easy – YouTube channel (2017) How to sort objects using Comparator Interface: Java https://www.youtube.com/watch?v=u8D2fydghj4 Accessed 23-DEC-2017, Stackoverflow.com.

Movie Theatre Prices (2017) Odeon Logo https://www.movietheaterprices.com/odeon-cinema-prices/ Accessed on 26-DEC—2017.

nIcE cOw – Stackoverflow User(2012) Loading Image resource in Java Swing https://stackoverflow.com/questions/9864267/loading-image-resource/9866659#9866659 Accessed on 26-DEC-2017, Stackoverflow.com.

Robin – Stackoverflow User (2012) Passing variable to ActionListener in Java using Client Property https://stackoverflow.com/questions/11037622/pass-variables-to-actionlistener-in-java Accessed on 26-DEC-2017, Stackoverflow.com.

Tarnowski, A. (2013) The Law Of Demeter https://www.youtube.com/watch?v=OMIZt2W_T_I Accessed on 01-JAN-2018.

UHerts (2017) MouseListener and ActionListener implementation for CCS: PPD course materials University of Hertforshire.

### Bibliography

Driessen, V. (2010) A successful Git branching model http://nvie.com/posts/a-successful-git-branching-model/ Accessed on 16-NOV-2017.

Gassner, D. (2017) Java Essential Training http://www.lynda.com Accessed NOV-2017.

Sierra, K., Bates B. (2017) Head First Java – 2nd Ed. Reilly Pub., US.

Stephens, R. (2015) Beginning Software Engineering John Wiley & Sons., US.