Introduction, Updated Design, and Discussion :

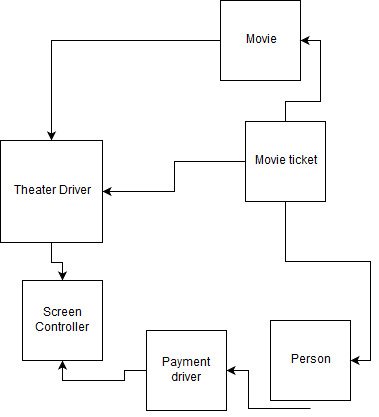
Group 26 consists of Max Bond, Sean Roach, Jason Esquivel, and Connor Simchak and we chose the Movie Theater Project. This project consists of creating a java program to handle functions related to running a movie theater, such as purchasing movies, accepting payment, and updating information. The original design focused on using GUI however once code started to come together it became clear a console application would be much easier to implement. We also decided to scrap the theater window as it was smarter to break the different functions into separate perspectives. There were certain requirements we handled in clever ways, for example, matinee pricing. The requirements asked us to use the current time to decide whether to give a matinee pricing, however in the real world that discount is decided by when the movie shows not when someone buys a ticket. Another clever solution to a requirement was how we created special deals. If two adults and any number of children attempt to buy tickets then the children's tickets receive a discount. Also, children receive a free ticket when attending a matinee. The majority of the main design structure from the initial design carried over to the final product.

In terms of testing one of the challenges we faced was how to give the program state persistence. We solved this by splitting the data into like parts, which allowed us to test each part of the data separately. Testing was done in a "build, break, fix" fashion, in other words, we built the entire system then, tried to break it in as many ways as possible. Breaking it allowed us to review bad code and fix the system. Artifacts of this testing process were left in the payment driver file. Since the program is interactive minimal hard-coded tests were needed to verify that interaction properly worked.

To better handle the different perspectives we built a screen driver class to facilitate changing between the screens. This differs from the UML diagram which attempted to use separate drivers for each screen. We also got rid of the theater class altogether as we decided to scrap theater capacity as a whole and have capacity handled by the number of tickets available for each. Saving and Loading data was handled by a PrintWriter that would write the contents of the lists that store the theater data in the theater driver class. When the program is started a constructor for Theater Driver is called and in that constructor the files are read and reloaded into the corresponding lists. Each object with a corresponding list has a function in it called "saveData" that returns a string of all the instance variables of that object. The close function is called, save data is called on every object of that type in each list.

The differences in the UML diagrams are mainly the addition of a bunch of additional functions within the theater driver class. These functions are by and large mostly just data shuttles to get parts of the theater driver data out of it that each individual class would then use. Another big difference is the removal of the theater class altogether as it was mostly turning into duplicate information that was already stored in the theater driver class and seemed unnecessary.

Updated UML diagram:



|  |
| --- |
| TheaterDriver |
| movieList : ArrayList<Movie>  personList :ArrayList<Person>  ticketList: ArrayList<MovieTicket>  screens: ScreenController  authenticated: false |
| startScreen()  engageManagerMode(int count)  close()  importMovieData()  importPersonData()  importTicketData()  viewTicketList()  viewMovies()  viewTickets(Person p)  purchaseTicket(Person p, MovieTicket t)  addMovie(Movie m)  editMovieGetter()  enterCustomerMode()  getPersonList()  addTicketToList(MovieTicket t)  addPersonToList(Person p)  Boolean personInList(String firstName, String lastName)  Person getPersonFromList(String firstName, String lastName)  Movie getMovieFromList(String title)  Person createPerson() |

|  |
| --- |
| Person |
| firstName : String  lastName : String  Age : Int  Minor : Boolean  Senior : Boolean  ticketList : ArrayList<movieTicket>  dtf : DateTimeFormatter  now : LocalDateTime  dayOfWeek : String |
| Person(String fn, String ln, int age);  Person(String fn, String ln);  Person buyTicket(MovieTicket t);  String getReceipt()  Person applyDeals()  ageVerified(ArrayList<MovieTicket> lmt)  String viewTickets()  String saveData() |

|  |
| --- |
| Movie |
| Title : String  Genre : String  Decription: String  TicketPrice: Double  String: Rating |
| Movie(String title, String genre, String description, double price, int runtime, String rating)  Movie(String title, String genre)  Movie(String title)  Movie()  saveData()  editMovie(TheaterDriver) |

|  |
| --- |
| MovieTicket |
| Price : Double  movie : Movie  showTime : Int  Type: String |
| MovieTicket(double, Movie, Int)  MovieTicket()  saveData()  applyMatineePrice()  getShowTimeAsString() |

|  |
| --- |
| Payment Driver |
| Customer : Person  Movie : Movie  PaymentType : String  Dtf: DateTimeFormatter  Now: LocalDateTime  dayOfWeek: String |
| PaymentDriver(Person, MovieTicket)  PaymentDriver()  purchase(Person, TheaterDriver) |

Initial UML Diagram:

|  |
| --- |
| TheaterDriver |
| movieList : ArrayList<Movie>  customerList : ArrayList<Customer>  theaterList : ArrayList<Theater>  ticketList : ArrayList<MovieTicket> |
| seatsAvailable(Theater t);  engageManagerMode(Person p, String username, String password)  Person purchaseTicket(Person p, MovieTicket t)  String viewMovies()  String viewTickets(Person p)  addMovie(Movie m)  editMovie(Movie M)  enterCustomerMode(Person p)  saveData()  close() |

|  |
| --- |
| Person |
| Money : Double  firstName : String  lastName : String  Age : Int  Minor : Boolean  Senior : Boolean  Manager : Boolean  ticketList : ArrayList<movieTicket> |
| Person(String fn, String ln, int age, double money);  Person(String fn, String ln, double money);  Person buyTicket(MovieTicket t);  String viewTickets()  String toString(); |

|  |
| --- |
| Movie |
| Title : String  Genre : String |
| Movie(String title, String genre)  Movie(String title) |

|  |
| --- |
| MovieTicket |
| Price : Double  movie : Movie  showTime : Long |
| MovieTicket(double price, Movie movie, long time) |

|  |
| --- |
| Theater |
| Capacity : Int  Movie : Movie |
| Theater(int capacity, Movie movie) |

|  |
| --- |
| Theater Window |
|  |
| TheaterWindow() |

|  |
| --- |
| Payment Driver |
| Customer : Person  Movie : Movie  PaymentType : String |
| PaymentDriver(Person, MovieTicket)  TicketList() |

