

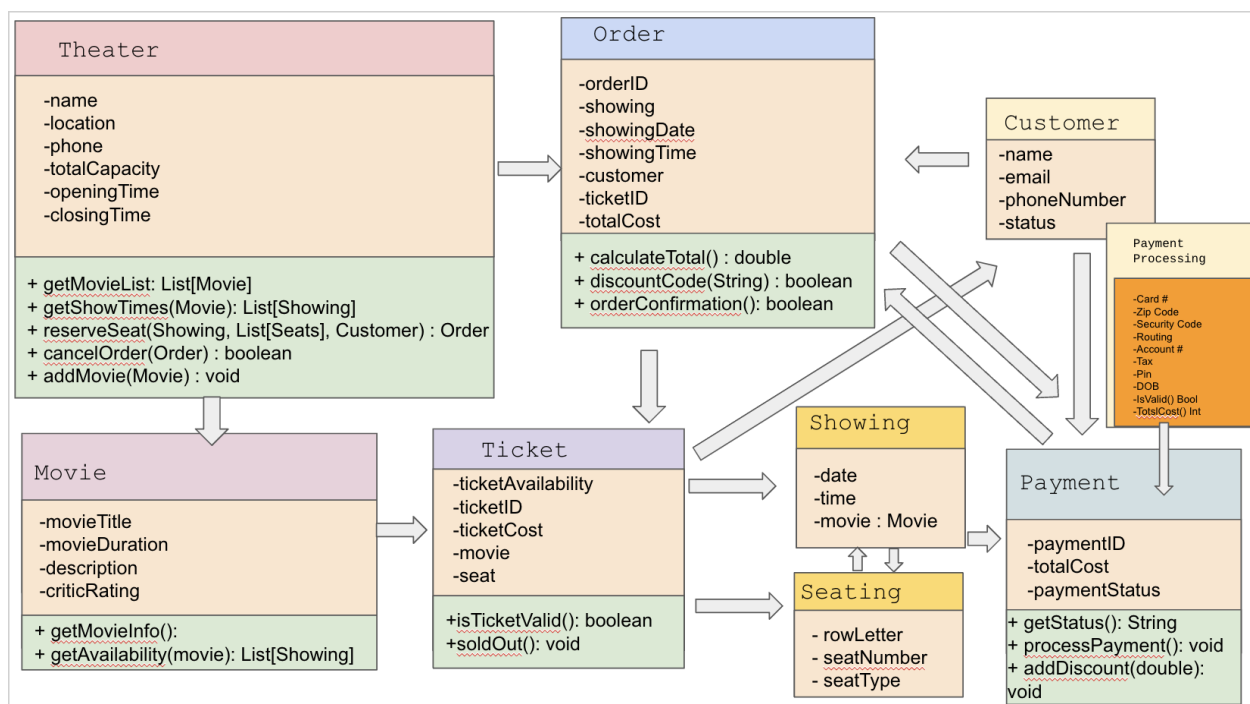
## Theater Ticketing System Class:

CS 250:

Software Systems Group Members:

Gabe Noda, John Choi, Will Blodgett, Andrew Snell

Description: Our theater ticketing system software is designed to automate the process of selling tickets for movies. It includes features such as seat reservations, pricing options, payment processing, and the ability to generate electronic tickets for customers. In addition to this, the software provides movie descriptions, ratings, and availability, along with other details for customers to see when browsing for their particular showing. The system will save customers name, email and phone number as a primary way for them to log back on to the system.



### CS250 UML Diagram Relations

The UML Diagram describes classes and how they relate to one another for a ticketing management system which could be used by a movie theater. It includes the following classes:

**Theater:** has private members which represent all attributes of a movie theater including **name**, **location**, **phone**, **max capacity**, and **closing and opening times**. It has public methods which can retrieve lists of movies and showtimes, as well as one to reserve seats, one to add movies to movie list, and another to cancel reservations made.

**Movie:** represents a movie and has members unique to each movie, such as, **movieTitle**, **movieDuration**, description, and criticRating. It has 2 public methods (**getMovieInfo()** & **getAvailability(movie)**) which are able to fetch information about a movie and check for ticket availability on a movie respectively.

**Ticket:** holds information pertaining to a ticket with private members such as: **ticketAvailability**, **ticketID**, **ticketCost**, **movie**, and **seat** (of type seating). It has public methods to check if a ticket is available, and confirm whether it is a valid existing ticket.

**Order:** responsible for storing and managing attributes of an order such as the ticket & information with the private members being: **orderID**, **showing**, **showingDate**, **showingTime**, **customer**, **ticketID**, and **totalCost**. Has 3 public methods capable of calculating your order total, with the optional addition of a discount code, and lastly a boolean returning whether the order was successfully placed and received.

**Customer:** represents a customer with private members **name**, **phone number**, and **email**.

**Showing:** represents a specific date and time for a showing of a movie, with private member variables **date**, **time**, and **movie** (of type Movie)

**Seating:** manages the ordering and purchases of the seats for a given showtime, with private members being **rowLetter**, **seatNumber**, **seatType**.

**Payment:** responsible for implementing payments for a movie ticket. Has **paymentId**, **totalCost** (before tax), **processPayment**, **addDiscount**, and **paymentStatus** members to indicate its current state

**Process Payment:** responsible for the transaction side of buying tickets. Has variables for card information, and a function **isValid** for checking if information is properly submitted.

### **Class Relations**

**Theater**→ **Showing**: the theater can have multiple showings of different movies, however since this is only for one theater there cannot be showings for other theaters.

private:

string name

string location

long int phone

int totalCapacity

pair<int,int> openingTime

pair<int,int> closingTime

public:

getMovieList returns List

getShowTimes returns pair from a List

reserveSeat is void

cancelOrder is void

addMovie is void

Movie→Tickets: Each ticket is for a specific movie and there can be multiple, however there will never be movies playing at the theater with no tickets available (under normal circumstances).

private:

string movieTitle

pair<int,int> movieDuration

string description

double criticRating

public:

getMovieInfo returns formatted information given in variables

getAvailability searches for movie in showing list

Showing→ Seating: Each showing will have a specific number of seats available for purchase.

private:

int date

pair<int, int> time

movie Movie

Tickets→ Showing & Seating: Each ticket corresponds to a specific showtime and specific seats which are reserved.

private:

boolean ticketAvailability

long int ticketID

double ticketCost

string movie

string seat

public:

isTicketValid returns ticketAvailability

soldOut returns void

Order→Tickets: An order will always contain tickets, and can contain multiple.

Order:

private: long int orderID

Showing showing

int showingDate  
pair<int,int> showingTime  
Customer customer  
int ticketID  
double totalCost

public:  
calculateTotal returns totalCost  
discountCode returns boolean for confirmation  
orderConfirmation returns boolean for confirmation

Order→Customer: Each order has one customer

Customer→Order→Payment: Customer can place an order for tickets which will then require payment which is also associated with each order (every order requires payment)

private:  
string name  
string email  
long int phoneNumber

Payment→Order: Each payment is for a specific order; don't want to accidentally purchase someone else's movie tickets with their seats and showtime.

Payment:  
private:  
long int paymentID  
double totalCost  
boolean paymentStatus  
  
public:  
getStatus returns paymentStatus  
processPayment returns void  
addDiscount returns void

Seating:  
private:  
char rowLetter  
int seatNumber  
string seatType

Payment Processing:

private:

Long int cardNum

int zipCode

int securityCode

long int routingNum

long int accountNum

double taxPercent

List<int> dateOfBirth

public:

isValid returns boolean

## **Movie Theatre Ticketing Timeline**

### Team Members:

Andrew, William, Gabe, John

### Description:

Our theater ticketing system software is designed to automate the process of selling tickets for movies. It includes features such as seat reservations, pricing options, payment processing, and the ability to generate electronic tickets for customers. In addition to this, the software provides movie descriptions, ratings, and availability, along with other details for customers to see when browsing for their particular showing. The system will save customers name, email and phone number as a primary way for them to log back on to the system.

### Code:

Theater Class: Andrew

Movie Class: John

Ticket Class: William

Order Class: Gabe

Showing & Seating Class: Andrew and William

Customer Class: Gabe and John

Payment Class: John

### Misc. Tasks:

Arithmetic Logic Planning: John and William

Abstract Logic Planning: Andrew Gabe

Quality Checking: John

Logic Checking: John

Graphics: Gabe and William

Prototype goal: March 20th

### Testing:

Theater Class Testing: Andrew

Movie Class Testing: John

Ticket Class Testing: William

Order Class Testing: Gabe

Showing & Seating Testing: Class: Andrew and William

Customer Class Testing: Gabe and John

Payment Class Testing: John

Testing finish goal: a week before project due date