Group 9
Phase 1

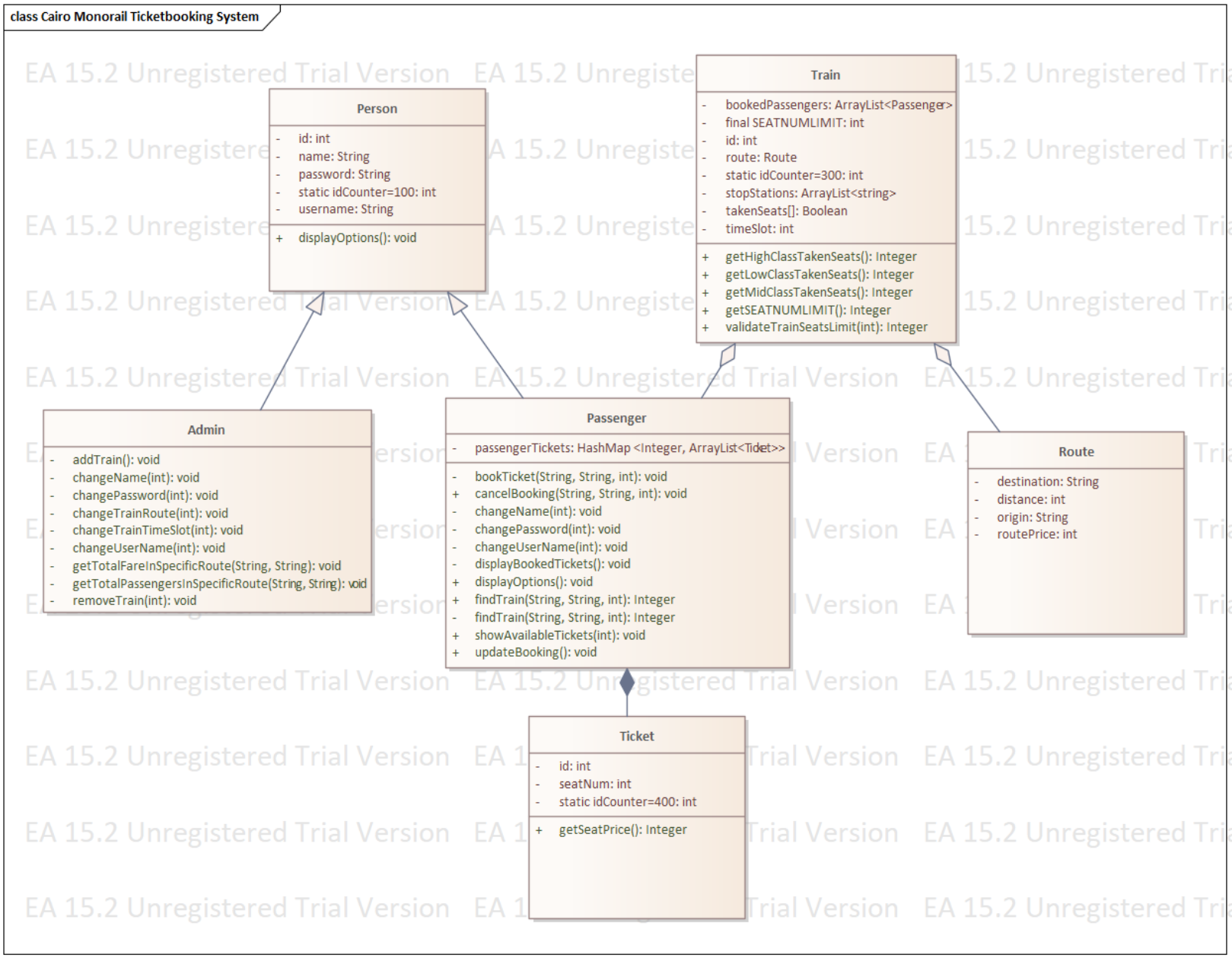
Contribution list

Ashraf 196280	1. Responsible for implementing the **Train** class. 2. Responsible for implementing the code inside the **main** method, with the **login** and **sign up** functionality. 3. Responsible **with** my colleague **Amro** in implementing the **findUsername()** function in the Station class 4. Responsible for implementing the logic of the **seats' price**: where the high class seats are 30egp, mid is class is 20egp, and low class is 10egp. 5. Responsible for overviewing the classes that were implemented by my colleagues; fixing any small bugs found in other classes like admin or ticket.
Amro 201762	1. Responsible for implementing and testing **Passenger** class. 2. Responsible for implementing **findUserName** function. 3. Responsible for fixing some logical error in **Ticket** class. 4. Suggested to initialize arrayLists inside the constructor of class **Train**. 5. Responsible for editing functions **changeName**, **changeUserName**, **changePassword** in class **Admin**.
Abdo 203795	1. Created the UML diagram for all classes. 2. Created the class station that has three static array lists for Train,Admin and passenger and checks that username is unique in our system or not and that no one has this user name by implementing a function for that.

	3. Created a class Route that has an origin, destination,distance , route price as private variables and to make sure that every distance and destination have a different price from another,This depends on the specific destination that the user will set. 4. Created a class Ticket with my colleague Ashraf the main purpose of this class is get ticket price depending on seat position all that by implementing a function that returns the price of the seat.
Mahmmoud 201573	1. Drew UML inheritance between Person super class and Admin Subclass. 2. Created Admin class that allows admin to control over station's trains by implementing functions to Add trains to the station as well as the ability to Modify or delete already created trains by passing their registered IDs in the system, also I implemented a validation logic to make sure that passed ID is already exists in our station or no, if no train associated with passed id found in our station system will raise an error message to let the admin user know that there is no train associated with passed train id, this logic is implemented on Delete train and modify train functions. There is also another validation logic I implemented on user's input data, for variables like route distance and route price, time slot, We will check whether data which user submitted is negative or no for both variables, for time slot we added validation to make sure that time won't exceed 24 hours based hour which we used in our system, this time slot will be used to identify when the train will go out of the station, this logic is implemented through modify and add train functions. 3. Created options in Admin class to allow logged in admin to Modify his account 's username and password,Account name, The functions which I used are inherited from Person class. 4. Gave admin users the ability to generate reports for specific routes by passing their associated origin and destination, before asking the user to submit the origin and destination of the route admin wants to generate report about, I gave the admin two options, first one to generate reports about

	Passengers numbers across this route, second one to generate report about total fare of tickets across this route. In the report which I generate for both options I collect all trains which are associated with this origin and route and then generate a report with trains and numbers of passengers as well as number of tickets according to their categories (High,Mid,Low) class, after that I generate report with total fare for all tickets categories, and multiply them with their constance price saved in our system.

UML Diagram


class Cairo Monorail Ticketbooking System
Person
- id: int
- name: String
- password: String
- static idCounter=100: int
- username: String
+ displayOptions(): void
Train
- bookedPassengers: ArrayList<Passenger>
- final SEATNUMLIMIT: int
- id: int
- route: Route
- static idCounter=300: int
- stopStations: ArrayList<string>
- takenSeats[]: Boolean
- timeSlot: int
+ getHighClassTakenSeats(): Integer
+ getLowClassTakenSeats(): Integer
+ getMidClassTakenSeats(): Integer
+ getSEATNUMLIMIT(): Integer
+ validateTrainSeatsLimit(int): Integer
Admin
- addTrain(): void
- changeName(int): void
- changePassword(int): void
- changeTrainRoute(int): void
- changeTrainTimeSlot(int): void
- changeUserName(int): void
- getTotalFareInSpecificRoute(String, String): void
- getTotalPassengersInSpecificRoute(String, String): void
- removeTrain(int): void
Passenger
- passengerTickets: HashMap <Integer, ArrayList<Ticket>>
- bookTicket(String, String, int): void
+ cancelBooking(String, String, int): void
- changeName(int): void
- changePassword(int): void
- changeUserName(int): void
- displayBookedTickets(): void
+ displayOptions(): void
+ findTrain(String, String, int): Integer
- findTrain(String, String, int): Integer
+ showAvailableTickets(int): void
+ updateBooking(): void
Route
- destination: String
- distance: int
- origin: String
- routePrice: int
Ticket
- id: int
- seatNum: int
- static idCounter=400: int
+ getSeatPrice(): Integer