Assignment 2

Bus Station

Objective

- Design a complete object-oriented system.
- Draw a UML Class Diagram.
- Practice on the concept of inheritance and polymorphism.
- Get to know files and exceptions.
- More practice on GUI.

Description

This project is almost a complete simulation for a real-life bus station. **First**, we have the bus station itself with a complete set of vehicles like a bus, minibus, limousine, ...etc. **Second**, there are the employees who are divided into drivers and managers. **Third**, the passengers (customers) who are trying to buy a ticket for a specific trip.A customer can reserve one-way trip or round trip. **Fourth**, we have two main types of trips; internal (between cities) and external (between countries), they differ in price, distance, ...etc. Also, a trip can be nonstop, one-stop or many stops.

You are required to develop a simulator for such a system in Java using the OO concepts you have covered in the course till now. However, you also required to be creative and do some brainstorming with your partner about how a real system works, remember, this is mainly a design course, soa good use of object-oriented concepts like inheritance, polymorphism, abstract class, and interfaces will be graded significantly.

You are also required to develop a **GUI** that should support two modes of operations for two main actors: passengers and employees.You should work with **files** too to store the trips information, passengers' list for each bus, …etc.

You will be required to deliver a **Class Diagram** for your work and the detailed division of labor among the two of you.

Scenario

Your program should follow the following scenario atleast (you are free to add features as you see fit) :

1. When starting the program you get a prompt to select which kind of user is using the program a passenger or an employee.
2. If a passenger is selected,he then is asked to enter a username and a password for authentication and then he opens his profile.
3. From the passenger profile he is able to select the trip he wants to make (source,destination,one-way,round-trip,number of stops … etc) from a list of available trips.
4. When the passenger books a ticket (if there are available seats) he is shown a price for the selected ticket(s) and then proceeds to buy them.
5. The passenger is able to review and cancel his tickets from his profile.
6. If an employee is selected,that employee can be a manager or a driver
7. If you log in with a driver credentials you are directed to the drivers profile with some basic information about the driver and the trips that are assigned to him by the manager.
8. If you log in as a manager you are able to review all trips in the system,you are able to add / remove trips and assign drivers to the trips in the system.

Tasks

- You should concentrate on the **inheritance** and **polymorphism**part, take your time in the design phase so you can reach a good object-oriented model. You are required to support at least:
    - Three types of vehicles.
    - Two types of employees.
    - One type of customers. (or more)
    - Two types of trips, with 3 different flavors (nonstop, one-stop, many stops).
    - Two types of tickets, and note that a round trip ticket should be cheaper than a single ticket.

- You are encouraged to add more classes and extend more types in your project.
- You should at least provide one **abstract class** and one **interface**, you are free to add more depending on your design.
- Your code should **read and write to files** to save and load data
- You should load some data from some files at the program startup, so you will have some trips and customers to play with.
- You can save the updates automatically to the files, or provide a save button to save changes to the files.
- You should provide a good GUI which will ease the use of the system, you are free to use any GUI framework, and you are free to make any design you like.
- Your GUI should support two modes of operations, one for the customers to buy a ticket and the other for the manager or a driver to update a trip or a ticket price, …
- You should draw a class diagram of your project.
- You should handle errors and show error massages to the user, use try and catch to help you make your program stable.
- You should apply all the object-oriented concepts like **encapsulation**, …etc. And don't use static unless you really need it.
- Be creative! The required features are only the beginning of what you can do, add more features or spice up the required ones, bonus marks will be given to those with eye-catching extra features and user-friendly interfaces.
- This project will consume some time in developing, but it has minor difficult parts, so enjoy what you are doing and try to learn as much as you can. It is meant to be easy and to help you learn.
- It is better to use a git version control service like githubto coordinate with your teammate.
- Interface can be used as a contract between the two of you,it will make the teamwork smoother.

- You should maintain a clean code for your program, like: variables, classes, functions naming, code indentation, function length, …etc.
- Again, google is your best friend the debugger is your second-best friend XD

Integration

You can use these interfaces as a guide for you, you are free to edit or remove them.

```
publicinterfaceAdminActions
{
/**
     *@return list of vehicles (3 types at least)
     */
publicArrayList<Vehicle>listVehicles();

/**
     *@return list of trips
     */
publicArrayList<Trip>listTrips();

/**
     *@return list of drivers
     */
publicArrayList<Person>listPersons();

/**
     *@param list
     *            save list of vehicles
     */
publicvoidsaveVehicles(ArrayList<Vehicle> list);

/**
     *@param list
     *            save list of trips
     */
publicvoidsaveTrips(ArrayList<Trip> list);

/**
     *@param list
     *            save list of drivers
     */
publicvoidsavePersons(ArrayList<Person> list);
}
```

```
publicinterfaceCustomerActions {
/**
     * Customer can search for trips and list them based on some criteria
     *
     *@param filter
     *          used to handle trip search
     *          if null, then return all trips
     *@return list of trips that matches the search criteria
     */
publicArrayList<Trip>listTrips(Map<String, Object> filter);

/**
     *@param selected
     *          the selected trip by the customer
     *@param selected2
     *          this parameter will be null if one-way trip
     *@paramnumOfSeats
     *          if null, check for a single available seat
     *@return true if there is available seats
     */
publicbooleancheckAvailability(Trip selected, Trip selected2, IntegernumOfSeats);

/**
     *@param selected
     *          the selected trip by Customer
     *@param selected2
     *          this parameter will be null if one-way trip
     *@param customer
     *          the customer currently using the system
     *@paramnumOfSeats
     *          it can't be null, and it must be > 0
     *@return the ticket with the total amount of money paid
     */
public Ticket reserve(Trip selected, Trip selected2, Person customer,
IntegernumOfSeats);
}
```

Deliverables

- You should work in groups of two.
- You should develop this assignment in Java.
- You are free to use any GUI framework you like.
- You should submit this assignment online on Github, you should include the following:
  - Your **code** folder.
  - **A self-executable jar file (JRE 1.8)**: The program should be executable by simply double clicking the icon, given that you have a running JRE.
    (Make sure your jar file is running with no conflicts on different computers)
  - **Class diagram**:You are permitted to use any software that allows you to draw the diagram itself not to automatically generate it from java class files.
  - A **readme.txt** file with any assumptions you need to run the program, and with the detailed division of labor among the two of you.
- A discussion date will be determined later.
- Bonus marks will be awarded for the following:
  - Using databases for saving and loading data
  - Using Git correctly for source control of your code showing multiple commits and branching behaviours.
  - Assuring integrity of data in your code (ex: If a manager deletes a certain trip then any passenger who booked this ticket will either be notified that their trip has been cancelled or it won't show in their booked trips)
  - Any extra behavior added to your project that enhances the experience.
- Late submission is accepted for only for 3 days after the deadline and will be graded from 70% of the full grade.
- [**Cheating Policy**] Delivering a copy will be severely penalized for both parties, so delivering nothing is so much better than delivering a copy.

Good luck