Alamein University
Faculty of Computer science & Engineering

Object Oriented Programming
Course Code: CSE015

Transportation Company System

1. Objectives:

⇒ Design a UML and convert it into Code.
⇒ Apply the Object-Oriented Analysis and Design concepts studied in class.
⇒ Handle Errors and Exceptions.
⇒ Use Files (Read and Write) using different modes.
⇒ Analyze and enhance the application performance.
⇒ Write a well-tested, documented, and clean code.
⇒ Write a maintainable, reusable, and readable code.
⇒ Develop a simple GUI to input and display application data.

2. System Requirements:

1. Implement a system that has a username and password authentication.
2. A System has a group of users that are either Passenger or Employee [drivers, managers], List of vehicles (bus, minibus, limousine) and List of trips (internal, external).
3. A user can login or register [Passenger or Employee].
4. Each Passenger has a name, ID, TicketType (one-way, round trip) and TripDetails (information about the reserved trip) and does some actions like select trips, booking tickets, reviewing/canceling tickets, and displaying the passenger profile.
5. Each Employee has a name, ID,Type (driver, manager) and does some actions like view assigned trips (for drivers), managing (add/cancel) trips, AddVehicle, AddEmployee, GenerateReport: [Generates a report containing information about vehicles, employees, and trips] (for manager), and accessing basic information (for drivers).
6. Each trip includes attributes such as Type (internal, external), price source, destination, one-way/round-trip, number of stops, available seats, and price and adding/removing trips and assigning drivers (for manager).
7. Each Vehicle has a type (bus, minibus, limousine), Capacity, LicensePlate and does some action like setType, setCapacity, DisplayInfo.
8. You must work with files to store the trips, passengers, employees, vehicles information and store information about reserved tickets.
9. Passengers, Employees registered in the system must be kept every time the program re-runs i.e. you have to keep a database in files.

3. Scenario/Final Product:

Your program should follow the following scenario

- A user will start the program with either registering a new account or logging in and then select which kind of user is using the program a passenger or an employee.
- If a passenger is selected,he then is asked to enter a username and a password for authentication and then he opens his profile.
- 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.
- 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.
- The passenger is able to review and cancel his tickets from his profile.
- If an employee is selected,that employee can be a manager or a driver
- 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.
- 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.

4. Steps:

a. UML class diagram:

- Design a UML Class Diagram using the notation studied in class.
- Make sure you understand the meaning of different relationships (Aggregation, composition, inheritance, association ...) before implementing them in your class diagram.
- Include all meaningful attributes, methods, multiplicity in your class diagram (ignore getters and setters).

b. Implement your classes:

- Implement classes shown in the class diagram in Java.
- Use clean, readable code.

c. Testing

- Test each method you have implemented.
- Enhance the performance of the program using Sorting and Searching Algorithms studied in class.

d. GUI (Bonus 5 Marks):

- Provide GUI for the program showing different views of the system.

5. Deliverables:

a. **Full Source code for the program**
b. **A project documentation including:**
   - UML class diagram (drawn with an online tool ex. Visual paradigm)
   - All data structures / algorithms you used.
   - Any assumptions or decisions you made.
   - A description about each method you are using and how does it work, avoid (Setters and Getters).
   - State all test cases you tried while making your tests for each method.
   - Screenshots for the program that shows every functionality, with different scenarios and test cases.
   - A description about how you used the files to store and load the state.
   - A description about the kinds of exceptions you are handling in each part.
   - State all performance enhancements you did, and how did you achieve that. What are the complexities you saved after using it and comparing it with the previously used ones.
   - **(Bonus)** Provide screenshots of different views of your GUI and a brief description of frames you used.

6. Notes:

- Develop this project using JAVA.
- You should work in groups of 3.
- You should show the contribution of each group member while submitting each report.
- Feel free to add any extra requirements to implement or to add any extra features to your application, this extra work will be considered as a bonus.
- You are allowed to use any GUI libraries/toolkits like JavaFX or Swing.
- Cheating Policy: All actual programming should be an independent effort. If any kind of cheating is discovered, penalties will apply to the participating students by zero in the project, so delivering a non-working program is so much better than delivering a copy.
- Submission deadline: **4th May 2024.**
- Discussion will be held on 5th and 6th May in your labs.
- Submitting one day late is allowed with penalty 25%-mark reduction, and late submission after 6th of May is **not allowed at all.**

7. Final Notes:

- Start with the class diagram, it makes everything easier when you look first abstractly on the problem.
- Ask Google! Site like Stack overflow has answers to any error that you may have, and you have to learn how to use it on your own.
- This is the type of project that goes up on your CV, so work hard to be proud of it.