Software Development

Mini Assignment 5

Due: October 25, 2018 at 23:55 on MyCourses

This mini assignment is a modified version of the question you saw in class during lecture 12. During that lecture it was presented as the last slide. I will present a similar, but more complex, problem here:

Questions: answer the following for this assignment -

- Which UML diagram(s), Class, Sequence and/or Activity, would be relevant for expressing the software design given the following use-case.
- Using UML-et, draw each solution diagram you identified as being relevant to the use-case. Provide one diagram per file.

Use-case:

ABC Airlines would like to manage their planes, flight routes, customers, passengers, tickets and employees.

They have 3 types of aircraft: Type A, Type B, and Type C. Type A are long distance passenger carriers. Type B are short distance passenger planes. Type C are cargo aircraft. Each plane contains common statistical information: maximum number of employees, maximum number of passengers, route, and the aircraft's current location (at source, in-flight, or at destination). Type A holds up to 20 crew and 300 passengers. Type B holds 3 crew and 10 passengers. Type C holds 5 crew and zero passengers. Type C contains a list (manifest) of all the cargo they are carrying.

A Route stores the name of the source location and the destination location. It defines the direction the airplane is travelling. A route is assumed to be reversible (plane travels from source to destination and then back again). A route is also replaceable (if the plan is at a source or destination location then its route can be replaced by another route. This is not true during flight).

Customers, passengers and employees (crew=employee assigned to a plane) are similar in that they have names and addresses. Specific to passengers are a valid ticket and a balance they still owe. Specific to an employee is the airplane they are assigned to (null if they are not assigned) and their position (teller, pilot, steward). Specific to a customer is a route they are interested in travelling. A customer must acquire a ticket to become a passenger.

A ticket is a token that has a unique ID, is attached to a route and an airplane. A ticket is given to a customer, this turns them into a passenger. A passenger is valid only when they have a ticket. To get a ticket a customer must interact with a teller. The teller helps them pick an available flight based on their route. Then three things might happen: (a) the customer changes their mind (or just wanted information) and leaves; (b) the customer pays for the full price of the fare, is given a ticket, and their balance is set to zero; or (c) the customer partially pays for the price of the fare, is given a ticket, and a balance is assigned to them which they owe.

Passengers can board an airplane only when they have fully paid for their ticket.

An airplane cannot be overbooked.

The management uses this application to track their aircraft, employees and passengers. The program has the following menu: (1) Teller (handles the conversion of a customer into a passenger and makes sure there is no over booking – passengers get an opportunity to pay their balance if they want), (2) Boarding (validates whether the user has paid their fee and marks then as boarded – passengers get an opportunity to pay their balance if they want – if this is a cargo plain then packages are assigned), (3) Assign Flight (matches airplane with a route and a crew – a teller cannot be part of a crew), (4) New Aircraft (creates a new empty airplane of any type), (5) New Employee (creates a new employee with their information and position), (6) Quit (exits the program).

Do not handle database issues. Do not handle files. Assume these exists in the background.

Make sure the following:

- Good object-oriented implementation (inheritance, abstract, polymorphism, encapsulation)
- Good use of contracts
- Optimal solution (design, data structures, OO, memory usage)

WHAT TO HAND IN

ZIP all the following into the file Mini5.zip (do not RAR):

- Convert your UML-et files into PDF and zip them into Mini5.zip

HOW IT WILL BE GRADED

For your assignment to be graded your diagrams must follow the assignment instructions. You are doing this assignment on your own.

- +10 – All diagrams have the correct syntax
- +10 – All diagrams depict an optimal solution that follows good OO practices (as seen to date)
- +10 – All diagrams provide a correct solution