Grading criteria: Readability

Grade: 8

Comment text: The UML chart is designed in high quality, very clear, and straightforward. Different classes, attributes, and class methods are presented in a concise and professional way.

Grading criteria: Validity

Grade: 5

Comment text: This chart fails to cover all the requirements mentioned in the instruction document, and many key aspects are missing, including functionalities related to creating a new order, buying a new drone, and maintaining existing drones.

Grading criteria: Fluidity

Grade: 5

Comment text: The chart introduces a new class called “register” which does not exist in the requirement, and this design of using a central register to process all orders is unnecessary and not efficient.

Comment #1

Course Topic: UML class model->aggregation

Course Content Reference: P2L3, P2L5

Comment Text: The design presents most of the relationships as required in the instruction, and most of the entities are connected with the appropriate relationships, with the exception of several minor errors. Drone pilots control drones to deliver orders, and there is not an explicit “part of” relationship between these to, so using aggregation between them seems to be inappropriate. Order and store should also be connected as any given order is placed in a particular store, and the association relationship is needed to identify the source store for a particular order. One drone pilot cannot be employed by multiple stores at the same time, so the numerical relationship between drone pilots should be 0…:1 rather than 0…:1…

Comment #2

Course Topic: Object Oriented Analysis->Candidate class

Course Content Reference: P2L2

Comment Text:

According to the lecture, we should extract the nouns, verbs, and adjectives from the requirement document for object-oriented analysis. The current UML design has a class called Register, whose usage is similar to a central book and can be used to calculate the cost of outstanding orders or total order weights. This design does not follow the guideline, as this Register never appears in the requirement document. In addition, having a central book doing all the calculations will negatively impact the performance of the system if we need to process a large amount of information. A better design is to add the calculate weight method to the order or drone class and add the calculate cost method to the order/store/customer class, and this allows those classes to do the calculations in a decentralized and potentially parallel manner.

Comment #3

Course Topic: UML Class->attributes

Course Content Reference:

Comment Text:

The design is good in general, and I only have a few minor suggestions. The number of months employed is directly included as an attribute in the Employee class, which means the system needs to update the value of the feature every month, and the value is not flexible regarding different ways of calculating the number of months employed (e.g.: 30 days/end of the month etc.). A better implementation is to just store the starting date and do the calculation when needed. Unit price should be an attribute of the item itself by nature rather than an attribute of the line item. Also, the design needs to reflect some operations mentioned in the instruction, such as stores allocating drones for deliveries to an order, stores buying new drones, or maintaining existing drones, etc.

Grading criteria: Readability

Grade: 7

Comment text: The UML chart is designed in a clear and straightforward manner, with different classes, attributes, and class methods presented in a concise and professional way. Changing some of the unnecessary polylines to straight lines can further improve the readability, for example, the connection between the drone pilot and the store.

Grading criteria: Validity

Grade: 5

Comment text: This chart fails to cover all the requirements mentioned in the instruction document, and many key aspects are missing, including functionalities related to buying new drones, and maintaining existing drones. Certain relationships between classes are also missing from the diagram, such as the pilots controlling drones to deliver orders.

Grading criteria: Fluidity

Grade: 7

Comment text: Good fluidity in general as the UML chart uses the proper data structures to store the information, with the exception of a few minor cases (e.g., the salary of drone pilots may not necessarily be an integer). The system is engineered efficiently without redundant elements.

Comment #1

Course Topic: Relationships->Cardinalities

Course Content Reference: P2L2, P2L3, P2L5

Comment Text: The design is not very accurate in terms of cardinalities, especially when some corner cases (or the initial state of those entities) are considered. Each order has to be issued by a store, but a store may not have any orders yet. Thus the cardinality should be 1 for store and 0… for order. Similar problems in cardinalities apply to the relationship between store/drone, store/item, store/drone pilot, and customer/order. There should be a link (association) between drones and orders as all orders are delivered by drones. Similarly, there should be a link (association) between drones and drone pilots since every drone is controlled by a drone pilot.

Comment #2

Course Topic: Object Oriented Analysis->Action verbs

Course Content Reference: P2L2, P2L5

Comment Text: The design is good in general, and I only have a few minor comments. DronePilot is only one particular type of employee according to the instruction, and adding the Employee class as a middle layer makes the system more flexible and scalable, considering that we may have more different types of employees later on. The design does not reflect some of the operations mentioned in the instruction, which means that part of the data flow cannot be captured by the system. Some examples include stores buying new drones (adding a new drone to the drones set in Store class), stores maintaining existing drones (resetting the trips remaining attribute in Drone class), drone pilots finishing a new delivery (updating the successfulDeliveries), etc.

Comment #3

Course Topic: UML Class->Name Compartment

Course Content Reference: P2L2, P2L3

Comment Text: The design is good in general, and I only have a few minor suggestions. The number of months employed is directly included as an attribute in the Employee class, which means the system needs to update the value of the feature every month, and the value is not flexible regarding different ways of calculating the number of months employed (e.g., 30 days/end of the month, etc.). A better implementation is to just store the starting date and do the calculation when needed. Some attribute/method names are confusing, e.g.: whether the “orders” attribute in the customer/store class means all historical orders or the current outstanding orders; whether the “revenue” attribute in the store class means the historical revenue or pending revenue; whether the totalCostOfOrders method in customer class means the cost of the current order or the cost of all historical orders. An improvement is to use slightly longer attribute/method names and make them more explicit regarding the intent of those attributes/methods.

Grading criteria: Readability

Grade: 8

Comment text: The UML chart is designed in high quality. It is easy for me/the reader to understand the structure of the system, the intent of different classes/attributes/methods and the interconnection relationships between classes.

Grading criteria: Validity

Grade: 8

Comment text: This UML chart, in general, covers all the requirements mentioned in the instruction document. Most of the attributes/operations and relationships are expressed properly.

Grading criteria: Fluidity

Grade: 8

Comment text: The UML chart successfully reflects most the key aspects as shown in the lectures and it uses the proper data structures to store the information and the entire system is engineered efficiently.

Comment #1

Course Topic: UML Relationship->dependency, association

Course Content Reference: P2L2, P2L3, P2L5

Comment Text: The design is very good in general, and I only have a few minor suggestions. According to the requirement document, an order can consist of one or multiple items (lines), and aggregation seems to be more appropriate compared to dependency in this setting. Any order can be linked to a particular store, and we know orders cannot exist without stores. So a composition relationship between order and store seems to be more appropriate compared to the association relationship. In addition, each order has to be issued by a store, but a store may not have any orders yet. Thus the cardinality should be 1 for the store and 0… for the order. Similar logic applies to Customer/Order as it is possible that a customer hasn’t placed an order yet.

Comment #2

Course Topic: Design Analysis->Data type

Course Content Reference: P2L5

Comment Text: The design is good in terms of choosing the correct data structures storing information in general, and I only have a few minor comments. Money is used as the data type of several attributes in several classes (credit attribute of customer class, revenue attribute of store class, salary attribute of employee class) and as the return type of several methods in several classes (displayIncomingRevenue method of Store class, displayCost method of Order class). It should be defined as a utility class similar to the class “Line” in the UML, but it is not. So I am not able to know the exact property of those attributes or methods. Similar problems apply to the data type “Weight”. In addition, the unit price should be an attribute of the item itself rather than the attribute of the Line.

Comment #3

Course Topic: Design studies->variables

Course Content Reference: P2L3, P2L5

Comment Text: The design is very good in general, and I only have a few minor comments. The UML diagram introduces a class called Line to track the quantity and unit price of the item. But the unit price, by nature, should be an attribute of the item itself, and the current design will lead to unnecessary repetition and waste of storage/memory. In addition, the item and lines are filled into two separate array lists in the order class, and their mapping is not strictly maintained. This may lead to a mismatch between item and line (i.e., confusion regarding which item a particular line corresponds to), and I think it is better to use a hash map data structure.