Week 8: Drawing UML class, object, sequence diagrams

Tags: W8 DG UML diagrams sequence diagram class diagram object diagram

Q&As

Questions related to notations

Q: Is visibility in front of a method/attribute required?

A: No, visibility is optional.

Q: When we show association as line, do we always need to have association role?

A: No, association role is an optional element.

Q: Can we always omit aggregation?

A: In this module, it is recommended to omit the aggregation diamond. However, you should still show the association line.

Q: Are the arrows for association and dependencies compulsory?

A: Navigability (i.e. the arrowhead in the association line) is optional. Dependencies are directional by definition (i.e. they are always shown as arrows).

Q: Is there a way to show final attributes in class diagrams?

A: You can capitalize the attribute name, that should be enough within the scope of this module.

Q: Is dependency arrow compulsory from Inventory class to Billable interface? (since there is already a way to tell that there is a dependency through the arguments of the generateBill(Billable) method)

A: Yes, it is better to show it visually, as the method signature is not a direct replacement for the dependency arrow.

Limitations in class diagrams

Q: If we show association as a line, in that case how do we show the visibility of the attribute?

A: In this case, there is no way to show the visibility of the attribute. In such cases, you need to decide the trade-off and choose which message you want your diagram to capture (i.e. to capture visibility vs to capture some message that could be delivered by showing association/attribute as a line).

Q: If we show Rating as an attribute inside the Review class, would there be a way for us to show/capture the whole-part relationship between Review and Rating since we do not explicitly show composition?

A: No. Again, you need to decide and choose which messsage you wwant your diagram to capture (i.e. to capture the whole-part relationship by showing composition vs to capture some message that could be delivered by showing association/attribute as a line).

Miscellaneous

Q: If the abstract Item class implements the bill() method in the code, then should the bill() method be included in its class in the UML model?

A: Yes.

Q: StockItem has both print() and bill() because you see it being implemented in the code. Item does not have bill() because you do not see it in the code?

A: Yes.

Q: Can a class have multiple composition relationships (e.g. there is a whole-part relationship between Review and Rating classes, and whole-part relationship between AnotherReview and Rating classes)?

A: Yes, it is possible, but the same Rating object cannot be part of two other objects. E.g. the below is possible, that doesn't mean that the same Leg object can belong to a Human and a Tiger at the same time.

Q: Why is there a dependency arrow from StockItem to Rating but association arrow from StockItem to Review?

A:

Dependency arrow: StockItem takes in Rating as a parameter in its constructor (but does not have a permanent reference to it).

Association arrow: Review attribute is shown as an association. StockItem has a reference to the Review. From the code, there is Review object as an instance variable inside StockItem.

Q: For the object diagram, can we jump straight from StockItem to Rating?

A: No. Remember that the object diagram is based on class diagram. In the class diagram, the multiplicity for Review (in the relationship between StockItem and Review) is 1. This means that one Review object is associated with one StockItem object. Hence, to jump straight from StockItem to Rating in the object diagram is wrong. It also does not align with what we have drawn in the class diagram.

Noteworthy

- Use a dependency arrow to indicate a dependency only if that dependency is not already captured by the diagram in another way (for instance, as an association or an inheritance).
- Composition implies when the whole is deleted, part is deleted too. But the reverse is not always true. Just because the implementation works like that does not mean it is composition. The vital question to ask is, "is there a real whole-part relationship between the two classes?"
- Constructors are not strictly methods, but it is OK to show them in class diagrams because they provide useful information.

- There are certain things that are shown in the class diagrams but not in object diagrams. These could include:
    - multiplicity
    - dependency
    - methods
    - inheritance

Note that **composition** can still be shown as they make sense for objects.

Interesting forum discussions

- (#331 (https://github.com/nus-cs2103-AY2223S1/forum/issues/331)) Class diagram association for binary tree node