CS2103/T AY2223S2 – Exam Feedback

As the exam was primarily MCQ, there isn't much feedback to give, but given below are some points to note, **based on questions that had a high percentage of incorrect answers**.

[Implementation] Magic numbers are not mentioned in the coding standard

The coding standard used by the module does not mention *magic literals*. Hence, using magic numbers is not a coding standard violation.

[Revision Control] Git doesn't need login

Git doesn't require a login, but GitHub does. To access a Git repository hosted on GitHub, you need to log in to GitHub first -- but that's a GitHub requirement, not a Git requirement.

[Project Management] Our CI platform was GitHub Actions

GitHub Actions was our primary CI platform. CodeCov was an external tool our CI used to check code coverage.

[QA] tP did both validation and verification

Both *validation* (i.e., ensuring we are building the right product) and *verification* (i.e., ensuring the product is built right) are important -- so, you were expected to do both of them.

Some of you thought validation was not expected, but that's not true. For example, you were supposed to ensure the product is optimized for keyboard users -- which is in the *validation* area than *verification*. Also, there were many points at which you were expected to think from the user's perspective and ensure the product solves their problem (e.g., when you wrote user stories, PE-D, PE). Yes, we did not involve real users but that doesn't mean you did not do your own validation.

[UML: Class Diagrams] Unidirectional associations vs bidirectional associations

This is two unidirectional associations that happen to be between the same two classes but in opposite directions:

This is a bidirectional association:

They are not the same.

[UML: Class Diagrams] Associations *roles* are different from *labels*

This is an association *role*:

This is an association *label*:

They are not the same thing.

[UML: Class/Object Diagrams] Composition symbol can be omitted

Omitting the composition symbol in an object diagram does not make if *incorrect* or *non-compliant*, just less informative. For example, the object diagram below is compliant with the class diagram above.

[UML: Class Diagrams] Multiplicity can be *unspecified*

Consider the statement:

> A `Foo` is linked to exactly one `Bar`.

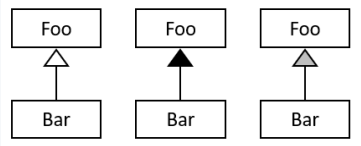
👍 It means this:

👎 It does not mean this:

That is, *unspecified/unknown* multiplicities should not be assumed as `*`.

[UML: Class/Object Diagrams] Color of the inheritance triangle does not matter

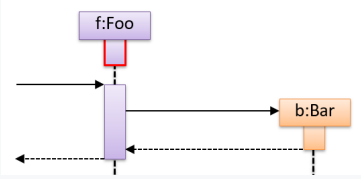
When showing inheritance, the fill color of the triangle does not matter. For example, the three diagrams below are equivalent.



Within the scope of UML notations covered in our module, the only time the fill color matter is when distinguishing between composition and aggregation.

[UML: Sequence Diagrams] Don't show constructor activation bar of existing objects

You can show the constructor activation bar for objects that are created *during* the interaction, but not for objects that existed at the start of the interaction. For example, it is incorrect to show the constructor activation bar of `f` in the diagram below.



Topic not in the textbook

While concepts such as fat JAR files, smoke testing, Pascal case naming are not specifically mentioned in the textbook, you should have encountered them during the iP and/or the tP.

Dogfooding (in the context of testing) is mentioned in the textbook, although a few mentioned they haven't encountered it before.