CS2103/T Practice Exam - Part 1 (Model Answers)

> ! **You are strongly encouraged to attempt this exam via Examplify** at least once before looking at these model answers. The password for the exam is `hello123`.
>
> This file is provided to you for convenience, because Examplify exam performance report (that you can access via https://examsoft.com) does not show images or give answers for short-answer questions.

○ [p1.01] requirements: use cases (text)

? Which step in the following use case does something a well-written use case should **not** do? ____

```
System: PosiIt (an online forum)
Use case: Post comment in a thread
Actor: user
Main success scenario:
    1. User clicks the 'post' button.
    2. PostIt requests for the comment details.
    3. User enters the comment details.
    4. PostIt requests for confirmation.
    5. User confirms.
    6. PostIt saves the comment and displays the updated discussion thread.
  Use case ends
```

What's the problem? ____

A1:

- ✓ 1
- ○ 2
- ○ 3
- ○ 4
- ○ 5
- ○ 6

A2: Contains UI details.

○ [p1.02] Design fundamentals

? Which statement is **incorrect**? ____

Why is it incorrect? ____

A1:

○ *Abstraction* helps us deal with information overload. We can even use multiple levels of abstraction.

○ Loose *coupling* is better than tight coupling.

✓ We can achieve *zero coupling* if we do a good job with the design.

○ High *cohesion* is better than low cohesion.

○ It is possible to improve both *coupling* and *cohesion* at the same time.

A2: It is not realistic to achieve *zero* coupling.

Examiner note: Different parts of the software need to work together in any non-trivial software.

○ [p1.03] design: about models

? Which is the **incorrect** statement about *models* (e.g., UML diagrams) used in software projects? ____

Why is it incorrect? ____

A1:

○ Models can be used as a tool to deal with complexity of software projects.

○ Models can be used for documenting software.

✓ Models should be as detailed as possible.

○ An architecture diagram is a model.

○ A sequence diagram is an abstraction.

A2: Models should omit details not relevant to the purpose of the diagram.

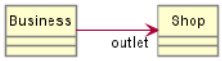
Examiner note: Models are used for dealing with complexity; adding too much detail will make the model more complicated and harder to understand.

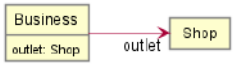
○ [p1.04] UML: CD: attribute vs association

? Which of these diagram is **not** equivalent to the others? ____

Why is it different? ____

A1:

○ 

✓ 

○ 

○ 

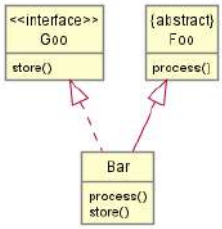
○ 

A2: The association is shown as both an attribute and a line.

Examiner note: An association can be shown as a line between the two classes or as an attribute in one of the classes (but not both).

○ [p1.05] UML: CD: interpret diagram

? Consider the UML diagram below. Assume all methods are shown in the diagram and the `Goo` interface doesn't have any default method implementations.



Which statement is **incorrect**? ____

Why is it incorrect? ____

A1:

○ `Bar` class is overriding the `process()` method.

○ Removing the `process()` method from the `Bar` class will not cause a compile error.

○ Removing the `store()` method from the `Bar` class will cause a compile error.

✓ The code below causes a compile error because we cannot instantiate abstract classes.
`Foo foo = new Bar();`

A2: Abstract classes *can* be used as a type.

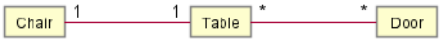
☐ [p1.06] UML: OD to CD

?

[**Select all that apply**] This object diagram is compliant with which class diagrams?

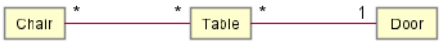
☑

☑



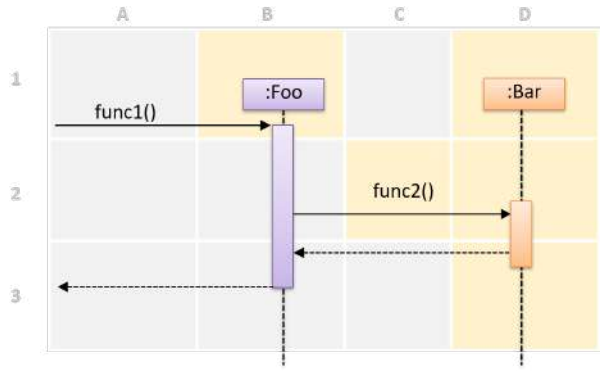
☐

☐



☐ [p1.07] UML: SD: detect notation errors

? [**Select all that apply**] Which cells contain notation errors?



☐ B1

☐ C2

☐ D1

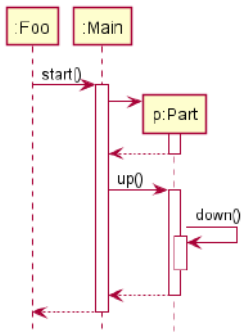
☑ D2

☑ D3

Examiner note: D2: activation bar starts before the method call arrives; D3: activation bar continues after the method call returns.

○ [p1.08] UML: SD: Interpret notation

?



Which is the **incorrect** statement about the above diagram? ____

Why is it incorrect? ____

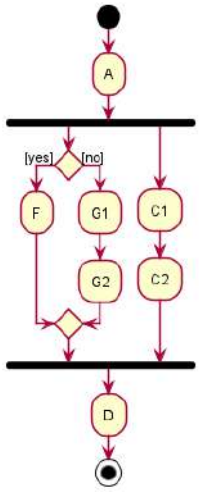
A1:

- ✓ The `Main` object is calling its own `start()` method.
- ○ The `down()` method belongs to the `Part` class.
- ○ Only one object in the diagram has been given a name.
- ○ There were only two objects when this interaction started.
- ○ At least one constructor is involved in this interaction.

A2: The start() method is called by the Foo object.

☐ **[p1.09]** UML: AD: interpret forks and branches

?



[**Select all that apply**] Which sequences are **allowed** by this activity diagram?

☐ A C1 C2 D

☑ A C1 C2 F D

☑ A C1 C2 G1 G2 D

☑ A C1 F C2 D

Examiner note: A C1 C2 D path cannot happen because both parallel paths need to be completed before D can be reached.

☐ [p1.10] code/design quality attributes

? [**Select all that apply**] Which of these are to be avoided/minimized?

- ☐ SLAP
- ☐ cohesion
- ☐ abstraction
- ☑ magic numbers
- ☑ coupling

☐ [p1.11] Java: coding standard

? [**Select all that apply**] Which three lines in the following code have **coding standard violations** (as per the coding standard using in the module)?

You may ignore any violations related to indentation, or spacing around operators.

```
1  /*
2   * Print the given string with the prefix ">".
3   *
4   * @param abcd String to be printed.
5   */
6  public void printWithPrefix(String abcd) {
7      if(abcd.length() > 3)
8          print(">" + abcd);
9  }
```

☐ 4

☐ 6

☐ 8

☑ 1

☑ 2

☑ 7

○ [p1.12] implementation: refactoring

? Given below are some changes Tom did to his code. Which one is **not** a *refactoring*? ____

Why is it not a refactoring? ____

A1:

- ○ Tom merges the `Greet` class and the `Prompt` class into one bigger class.
- ○ Tom finds that the variable name `find` is misleading. He changes it to `isFound`.
- ○ Tom thinks the `add()` function is too long. He applies the SLAP technique to it.
- ✓ Tom finds the `sort()` function doesn't work when the list is empty. He adds an exception to handle that case.
- ○ Tom removes braces around an `if` block because there is only one statement in the block.

A2: It changes the behavior.

○ [p1.13] Java: JUnit keywords

? Which of these Java keywords or method calls is **not** likely to be in the UI class of your software? ____

Why not? ____

A1:

- ○ assert
- ● assertEquals
- ○ catch
- ○ instanceof
- ○ throw
- ○ throws

A2: This is a JUnit method; should only be used in test classes.

○ [p1.14] testing: types, test case design

? Which of these is **most** likely to be *glass-box* testing? ____

Why? ____

A1:

- ○ system testing
- ○ acceptance testing
- ○ alpha testing
- ○ beta testing
- ✓ unit testing

A2: Only unit testing is done by developers.

Examiner note: As glass-box testing requires the knowledge of code, it needs to be done by developers.

○ [p1.15] PM: statements about rcs

? Which statement is **incorrect**? ____

Why is it incorrect? ____

A1:

○ A Git repository can *pull* from one remote repository and *push* to a different remote repository, provided all repos involved have a shared history.

○ When you *push* code to your *fork*, any open PRs created from it get updated with the latest code.

○ GitHub is an online service that offers Git features and more.

○ It is possible to edit the commit message of a past commit.

✓ *Forking* creates a local copy of a repository.

A2: Cloning (not forking) a remote repo creates a local copy.

○ [p1.16] PM: statements about rcs

? Which statement is **incorrect**? ____

Why is it incorrect? ____

A1:

- ○ An architecture diagram can use emoji symbols to represent components of a system.
- ○ The *n-tier* architectural style is also known as the *layered* architectural style.
- ○ The MVC pattern is an example of the *Separation of Concerns* principle being applied.
- ○ GitHub Actions plays the role of a *continuous integration* tool.
- ✓ High-level *user stories* that cover bigger functionalities are called *legends*.

A2: They are called epics or themes.