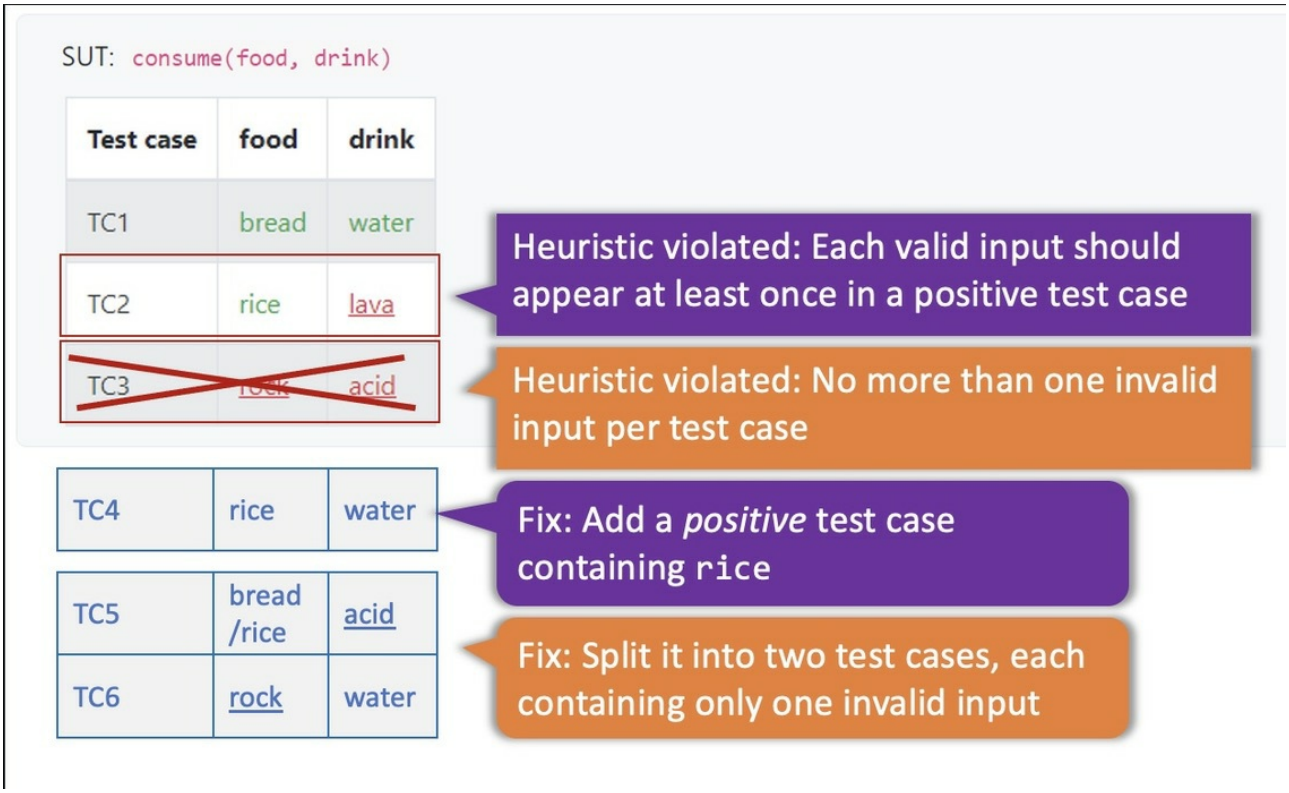
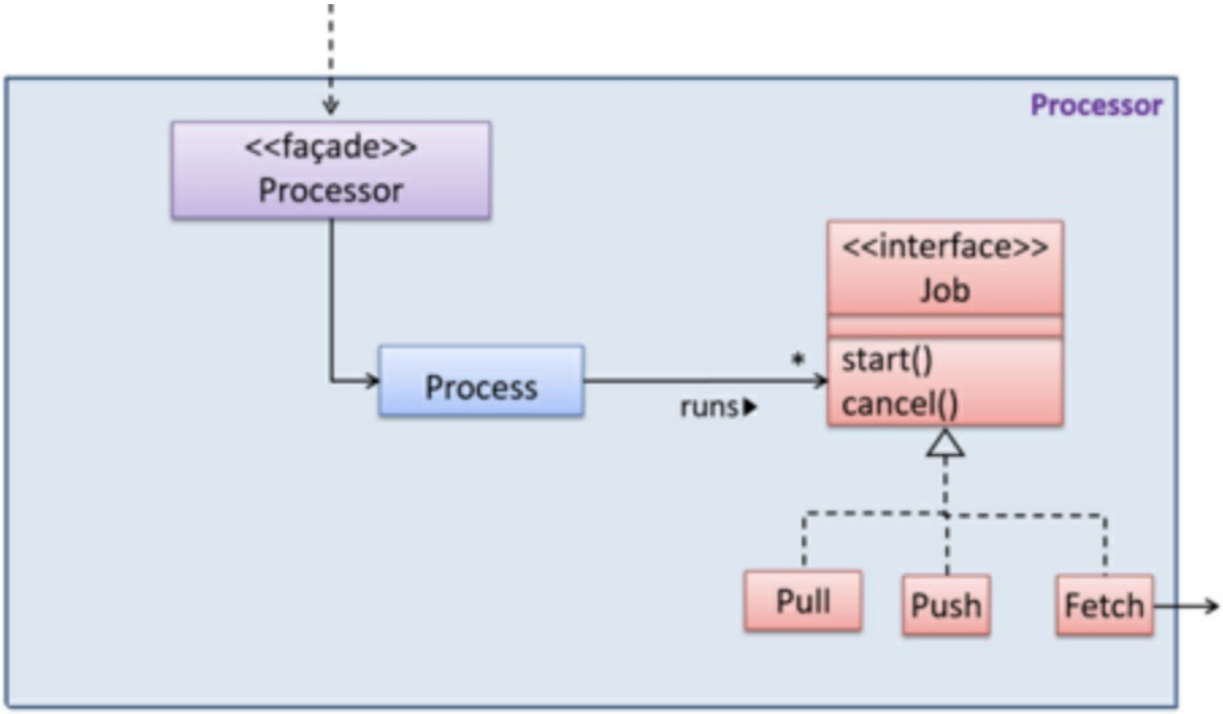
Week 11: Test cases heuristics, design patterns

Q&As



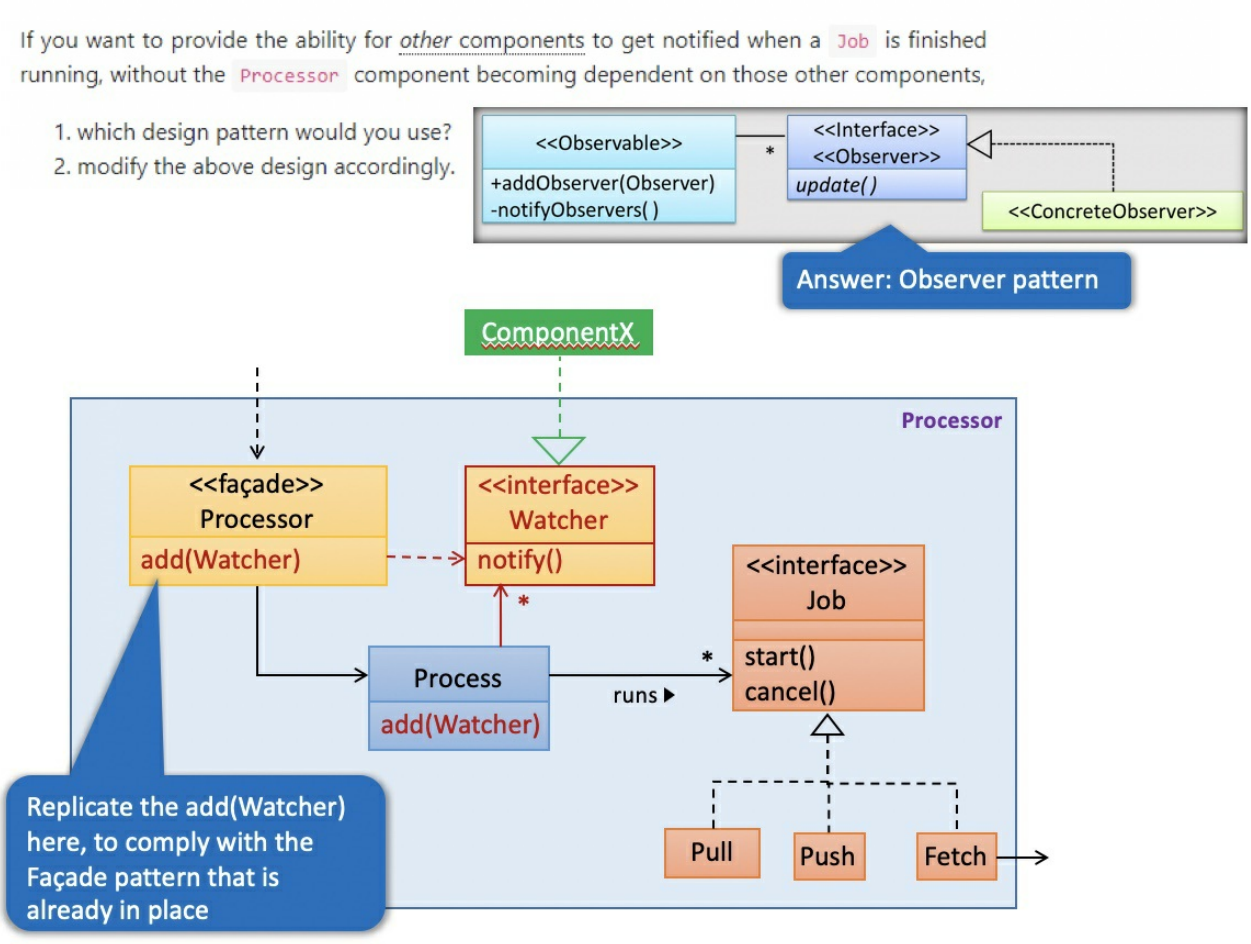
Q: Should test cases be in any particular order? For example, in the screenshot, it seems like it would be 'better' to slot TC4 right after TC1 as doing so we would have tested that SUT works for rice?

A: The ordering of test cases might have an effect. In this module, we do not consider that aspect. In fact, there is a general guidance that one should not depend on tests running in a particular order as some hidden dependencies between test cases can hide bugs. In fact, some testing frameworks run tests in random order by default, to prevent such dependencies.



Q: In the screenshot, is it necessary to have the label <<facade>>, and if so, should there also be label <<command>>?

A: It is optional to add such <<role>> to elements of the diagram, to provide more information about a special role played by such an element.



Q Can the Job class(es) play the role of <<Observable>>?

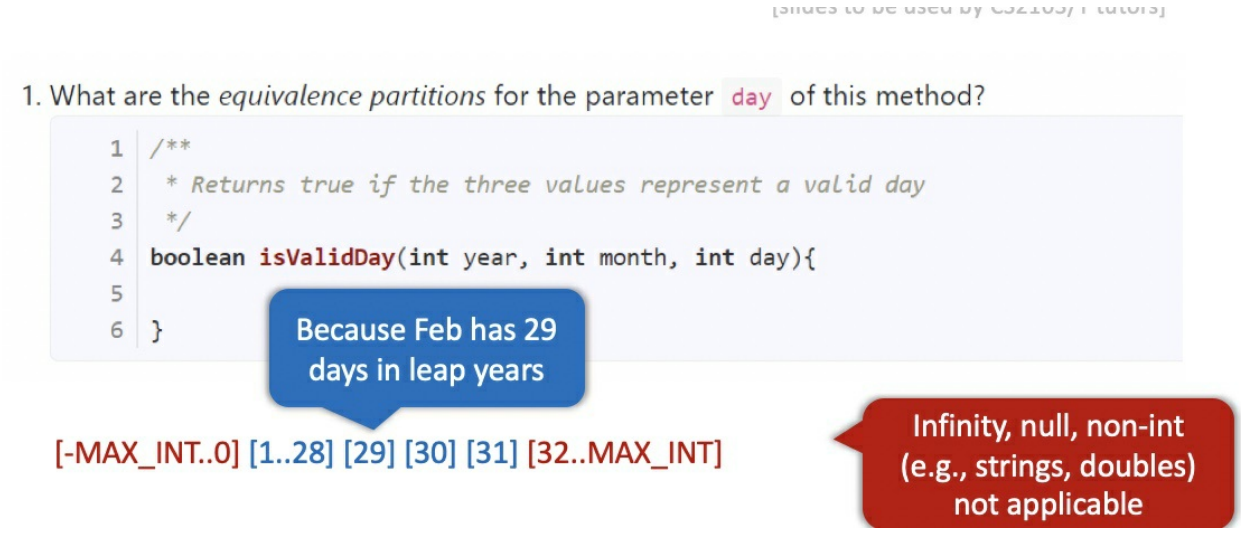
A: Possible but not suitable as managing <<Observer>> objects goes beyond the responsibility of the Job class. Process class seems to have oversight on managing Job objects and is better suited for managing Observer objects as well. In any case, the Job objects might not even exist at the start of the execution.

Q: Why don't we let the Process class be the facade?

A:

1. It already has other responsibilities
2. In the future, we might have to add more internal classes to the component that outsiders need to access. Accessing those through the Process class may not even make sense.

Noteworthy



- Remember we had a discussion on equivalence partitions? During the tutorial, some of you combine 29, 30, 31 as one EP (i.e. [29..31]).

EP are group of test inputs that are likely to be processed by the SUTs the same way. This also mean that they likely have the same logic/implementations handling these values.

To help indentify EP, you can try to think whether the logic/implementation handling these values are different. In this case, 29, 30, 31 would likely have separate logics handling them. Hence, they form EP in themselves.

Resources that I found useful for finals preparation

Good cheatsheet

- I used [this cheatsheet (https://github.com/AaronCQL/serious-collection/tree/master/CS2103T)](https://github.com/AaronCQL/serious-collection/tree/master/CS2103T) as a sanity check. I also used it as a quick reference during the finals in case I suddenly forgot anything (Note: This was not done by me, credits go to the creator).

Code quality

- Introduce assertion (https://refactoring.guru/introduce-assertion)
- Guard clauses (https://refactoring.guru/replace-nested-conditional-with-guard-clauses)

Design patterns

- Singleton (https://refactoring.guru/design-patterns/singleton)
- Facade (https://refactoring.guru/design-patterns/facade)
- Command (https://refactoring.guru/design-patterns/command)
- Observer (https://refactoring.guru/design-patterns/observer)

How I prepared for finals

1. Read through the entire module textbook one time, take note of things that I thought was noteworthy
2. Re-attempted all the weekly quizzes, take a screenshot of those that I still got wrong, then try to understand why I got them wrong. Revise those parts of the content. Look through these questions once again before exam.
3. Attempt past year papers under timed condition so I know my pace.
4. I read most of the forum discussion for past year papers questions.
5. I didn't make cheatsheet or notes. There was no time to look them up during the exam anyway.

Tips

Some tips from me here (https://docs.google.com/document/d/1yjzExU_AEERoxOOIlMCXdjr9uBxhYpaf8ltHMUE95zM/edit?usp=sharing)

- All the best for PED/PE/Finals, hope you achieve what you want for yourself! Otherwise, as cliche as it may sound, grades are not the only thing in life. Try your best :)
- Feel free to ask me questions if you have any doubts! :)