TA Guide for SED Ex3 - Reuse and Extensibility

To prepare, watch the lecture video linked below and download the exercise spec from CATE:

https://imperial.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=49332064-d9ec-4391-b9da-af370076920c

During the lab (Tuesday 4-6pm in Level 2 computer labs) you should:

Walk around the room, checking how people are doing. Look over their shoulder. Don't be afraid to talk to them. Do not just stand at the side and wait for people to put their hand up.

First of all the students should copy and paste the Fibonacci code to make two Triangle Number sequences. They should copy and paste the tests too, just adjusting the numbers. F5 (copy class) is your friend in IntelliJ.

A common mistake is to try to use a recursive formula for the triangle numbers, rather than just n * (n+1) / 2

If they're not sure how to get started, I normally suggest doing the Template Method Pattern first.

It's a point of design whether to factor out the throw-exception-for-negative-index code into the common class, or keep it with the specialisations. Either way is fine but they should know why they did what they did.

People often need to think more carefully about naming different components. Encourage them to think about the role that each object plays - but not to use pattern names or "interface" in their type names.

Once they have the two patterns in place with good code, think about duplication in the tests. Think about how many times each part gets tested (e.g. typically in TMP the superclass is getting tested twice).

There is no need to remove duplication between the two different parts of the exercise.

Example solution for the exercise:

https://imperial.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=80139e65-8a2e-4ef6-bc01-ad5f00c9866a

Marking scheme:

+ 10 if they get 3/3 on LabTS. This means everything compiles, tests pass and coverage is > 80%

+ 2 Correct Template Method Pattern (abstract superclass, abstract hook method, two subclasses overriding)

+ 2 Correct Strategy Pattern (interface TermCalculator (or similar) with 2 implementations - composition)

+ 1 for good clean code
+ 1 for all good names

So both patterns correct with great code, but no attention to removing duplication in the tests gives 16.

Strategy Pattern: + 2 for using a mock or a fake to test the NumberSequence class in isolation (with simple tests for the individual strategies)

Template Method Pattern: + 2 for using a test-only subclass to test NumberSequence logic in isolation (with short tests for the individual subclass logic)