TA Guide for SED Ex2 - Mock Objects

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=932b138b-e0b1-4476-83d7-af3001217d1d

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.

Check that they are writing some mock object tests, with a checking / Expectations block — mocking the Sensor and the Memory card — passing them in to the Camera as dependencies.

They should still be doing TDD and each test should be named as a fact about the behaviour.

They may see unexpected test failures if they:

- call the mocks directly
- trigger an action on the camera **before** setting relevant expectations instead of afterwards

They should not need any assertThat / assertEquals style checks. The presence of these normally means they are revealing internal state that they should not / need not be.

When it comes to the last two behaviours, they do not need to create any threads. They should not have tests that take any real time to run. Think about a sequence of events, rather than a long-running activity.

It's good to use ignoring() or allowing() to specify expectations that can happen but don't need to be verified.

Example solution for the exercise:

https://imperial.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=88b923ad-9624-49c2-9159-ad5f00c98694

Marking scheme:

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

+ 1 Mocks of Sensor and MemoryCard created and passed in to the Camera constructor

+ 2 Mock object tests written for each behaviour

+ 1 Correct use of will(returnValue(xxx))

+ 2 Implementation is correct and simple. Sensor & MemoryCard fields are private and final.

+ 1 Default power state is consistent (e.g. power is off to start with) and tests reflect that.

+ 2 Avoids duplication: Mocks and camera extracted as fields in the test. Common test setup (e.g. "switch on camera") factored out into a separate method

+ 1 Each test should be named as a fact about the behaviour.