Mock objects

* Cheap placeholder object
* Predictable and allow method to be tested in the current class
* Useful because otherwise an object may depend on other objects or parts of the system that are hard to control or predict
  + Real object may be complex, or take a lot of time/computing power to set up
  + Object hasn’t been coded yet
  + Object is non-deterministic
  + Has or is a user interface
  + Has behaviour that is hard to trigger (ex: network error)
    - Or a time specific behaviour (like an alarm that triggers at midnight)
* Implement
  + Use an interface to describe the object
  + Implement the interface for production code
  + Implement the interface in a mock object for unit testing

Test suites

* Allow grouping of multiple test classes to run as a single batch
* Run specific subset of unit tests from multiple test classes
* Use annotations to run a test suite

Test driven development

* Successfully written test will fail because there is no production code

Diagram

Description automatically generated

* See the code from the user’s perspective not the developer’s
* Helps avoiding writing a class that is hard to test
* EX: Functionality and GUI rolled into one, is hard to test, therefore change the design => a simpler design

Invariants

* Assertions about classes/objects that should not change
* Should always prioritise testing these
* EX:
  + In a bank account credits and debits should always match balance
  + Measurements in different units, need to match after conversion
  + When implementing an array 0 <= array index < length-1
* Need to ensure an object can never be viewed in an inconsistent state