# Naming Convention

We follow *XUnit Patterns* (<http://xunitpatterns.com/Test%20Double.html>) naming convention for objects used for testing.

Test Double

When the movie industry wants to film something that is potentially risky or dangerous for the leading actor to carry out, they hire a "stunt double" to take the place of the actor in the scene. The stunt double is a highly trained individual who is capable of meeting the specific requirements of the scene. They may not be able to act, but they know how to fall from great heights, crash a car, or whatever the scene calls for. How closely the stunt double needs to resemble the actor depends on the nature of the scene.

For testing purposes, we can replace the real dependency with **our** equivalent of the "stunt double": the ***Test Double****.*



**Test Stub**We use a ***Test Stub***to replace a real component on which the code depends on, so that the test can get canned responses from external components. It usually returns hard-coded values.

**Test Spy**We can use a more capable version of a ***Test Stub***, the ***Test Spy***. Like a ***Test Stub***, the *Test Spy* may need to provide values to the code in response to method calls but the *Test Spy* also captures those calls  and saves them for **later** verification by the test. So in many ways the ***Test Spy*** is "just a" ***Test Stub*** with some recording capability.

**Mock Object**We can use a ***Mock Object*** to verify the calls made by the code being tested. Typically, the *Mock Object* also includes the functionality of a ***Test Stub*** in that it must return values to the code but the emphasis is on the verification of the **external** calls (it may also check the order of the calls). Therefore, a ***Mock Object*** is lot more than just a ***Test Stub*** plus **assertions**; it is used a fundamentally different way. One disadvantage of using mocks is that they couple the tests to the implementations details.

**Fake Object**Typically, it implements the same functionality as the real **dependency** but in a much simpler way. While a ***Fake Object*** is typically built specifically for testing, it is not used as an **observation point** by the test.

**Dummy Object**Some method signatures of the **code** may require objects as parameters. If neither the test nor the [**code**](http://xunitpatterns.com/SUT.html) care about these objects, we may choose to pass in a ***Dummy Object***which may be as simple as a null object reference, an instance of the Object class.

# Code

* All database components are implemented using interfaces to allow the creation of Test Doubles.
* Some methods that should be private are defined as protected so that they can be tested with JUnit.