Mocks

Заглушка - это небольшой фрагмент кода, который заменяет другой компонент во время тестирования. Преимущество использования заглушки заключается в том, что она возвращает согласованные результаты, что упрощает написание теста. И вы можете запускать тесты, даже если другие компоненты еще не работают.

Когда нам нужно использовать double?

- Низкая скорость выполнения тестов с реальными объектами (если это, например, работа с базой, файлами, почтовым сервером и т.п.)
- Когда есть необходимость запуска тестов независимо от окружения (например, на машине у любого разработчика)
- Система, в которой работает код, не дает возможности (или дает, но это сложно делать) запустить код с определенным входным набором данных.
- Нет возможности проверить, что SUT отработал правильно, например, он меняет не свое состояние, а состояние внешней системы. И там эту проверку сделать сложно.

Согласно Жерару Месарошу, существует 5 видов тестовых двойников:

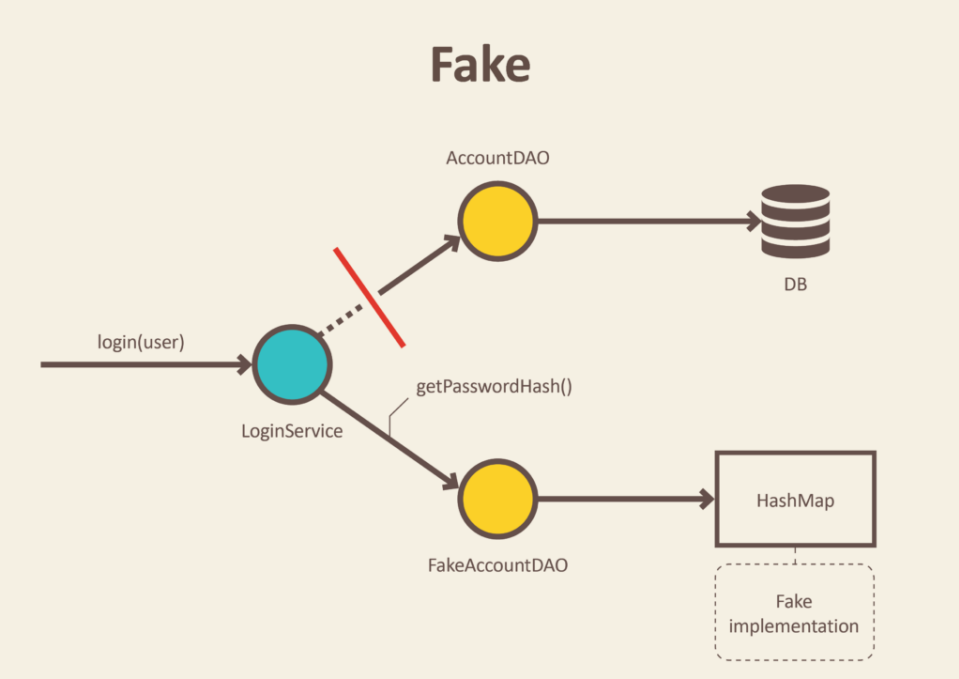
- Пустышка (dummy)
- Стаб (stub)
- Шпион (spy)
- Мок (mock)
- Фейк (fake)

1) **Fake (имитация)**

Имитация (или поддельные объекты) — это объекты, имеющие рабочие реализации, но не такие, как у настоящих рабочих объектов. Обычно они идут коротким путём и имеют упрощённую версию реального объекта.

В качестве примера может быть реализация в оперативной памяти объектов доступа к данным (Data Access Object) или репозиторий (Repository). Реализация поддельных объектов не будет привлекать базу данных, но будет использовать простую коллекцию для хранения данных. Это позволяет нам выполнять интеграционный тест сервисов без участия базы данных и выполнения тем самым трудоёмких запросов.

Основные примеры — эмулятор базы данных (fake database) или фальшивый web-сервис.

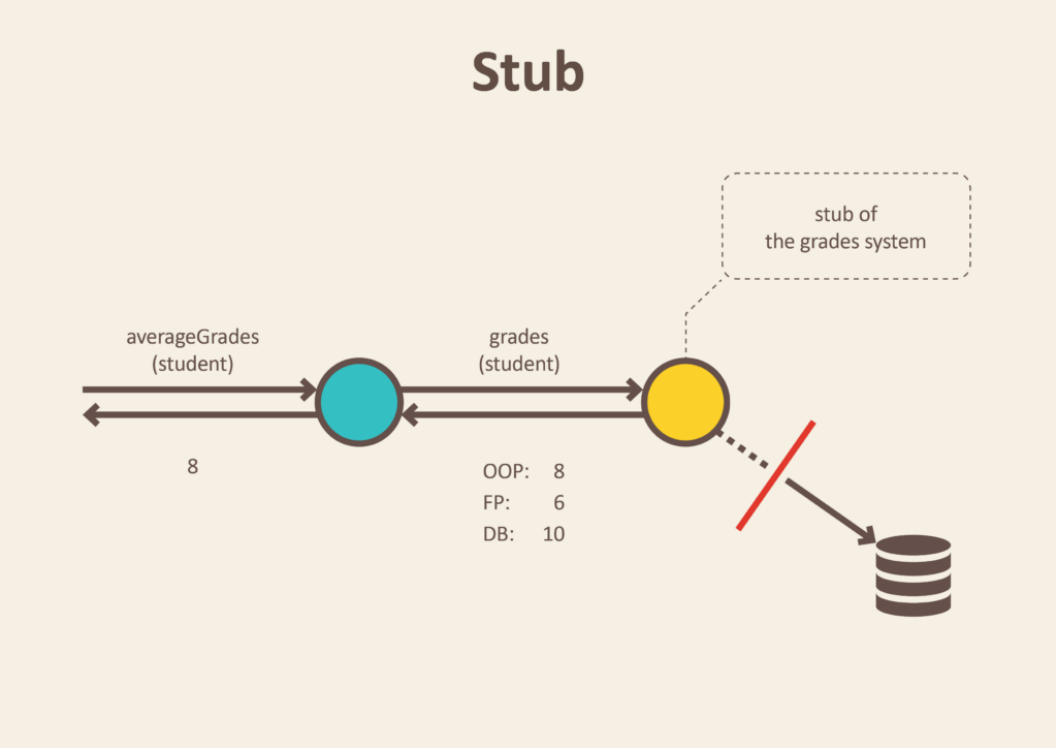


2) S**tub (заглушка)**

Заглушка— объект, содержащий предопределённые данные и использует их для ответа на вызовы во время тестов. Она используется, когда мы не можем или не хотим привлекать объекты, которые бы отвечали реальными данными или имели бы нежелательные побочные эффекты.

В качестве примера может быть объект, который должен получить некоторые данные из базы данных в качестве результата при вызове метода. Вместо реального объекта, мы вводим заглушку и определяем в ней, какие данные она должна вернуть.

Тестируемый объект использует чтение из конфигурационного файла? Тогда передаем ему заглушку `ConfigFileStub` возвращающую тестовые строки конфигурации без обращения к файловой системе.

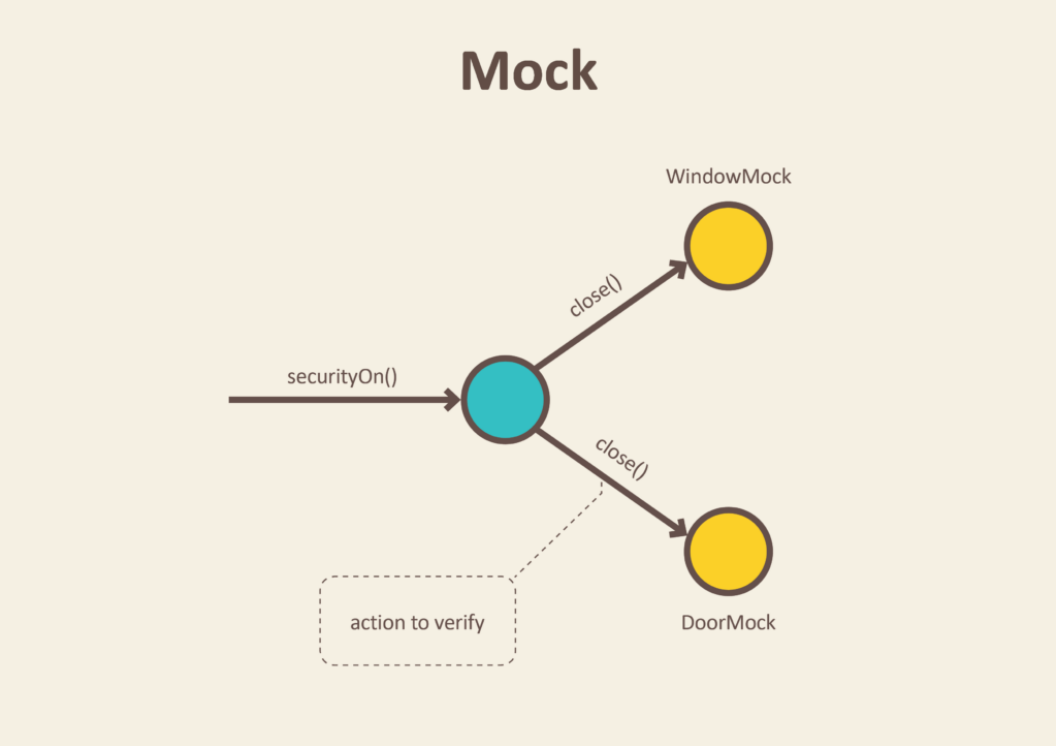


3) **Подставной объект (Mock)**

Подставные объекты регистрируют получаемые вызовы. В тестовом утверждении мы можем проверить, используя подставной объект, что все ожидаемые действия (иначе — вызовы) были выполнены.

Мы используем подставной объект, когда не хотим вовлекать реально работающий код или когда нет простого способа проверки, что код действительно был выполнен. Нет возвращаемого значения, а значит нет лёгкого пути проверить изменение состояния системы. В качестве примера может послужить функциональность, вызывающая сервис отправки электронной почты.Мы не хотим отправлять электронные письма каждый раз, когда запускаем тест. Кроме того, в тестах нелегко проверить, что письмо на самом деле было отправлено. Единственно, что мы можем сделать — это

проверить результаты функциональности, которые выполняется в нашем тесте. Другими словами — убедиться, что был вызван сервис отправки электронной почты.



4) **Spy** (**Шпион)**

Тестовый шпион - это объект, способный захватывать косвенный вывод и предоставлять косвенный ввод по мере необходимости. Косвенный результат - это то, что мы не можем наблюдать напрямую.

Разновидность заглушки, которая умеет протоколировать сделанные к ней обращения из тестируемой системы, чтобы проверить их правильность в конце теста. При этом фиксируется количество, состав и содержание параметров вызовов.

> Если существует необходимость проверки, что определённый метод тестируемого класса вызывался ровно 1 раз, то шпион - именно то, что нам нужно.

Мы можем добиться этого, расширив исходный класс и сохранив параметры функции в качестве аргументов класса.

5) **Dummy (Пустышка)**

Пустышки - это объекты, от которых зависит наш SUT, но они никогда не используются.

Dummy-Объект, который обычно передается в тестируемый класс в качестве параметра, но не имеет поведения: с ним ничего не происходит и никакие его методы не вызываются.

Примером dummy-объектов являются new object(), null, «Ignored String» и т.д.

Плюсы и минусы тестирования на заглушках:

Плюсы: Преимущество использования заглушки заключается в том, что она возвращает согласованные результаты, что упрощает написание теста. И вы можете запускать тесты, даже если другие компоненты еще не работают.

Минусы: Во-первых, моки надо создать. Надо устанавливать дополнительный софт, описывать логику, загружать нужные данные в макет ответа. Не то чтобы это сложно, но это требует усилий и времени. Может быть, кому-то будет проще сохранить результат вызова в файл и при тестировании обращаться не к заглушке, а к файлу. В небольших проектах с малым количеством тестов такой подход вполне оправдан.
Во-вторых, веб-сервис может поменяться. Например, программист написал рабочую логику для заглушки, а через какое-то время веб-сервис стал работать по-другому (а разработчика, как часто бывает, об этом никто не уведомил). О том, что в работе веб-сервиса что-то поменялось он узнал, когда выкатил обновление на рабочую базу.