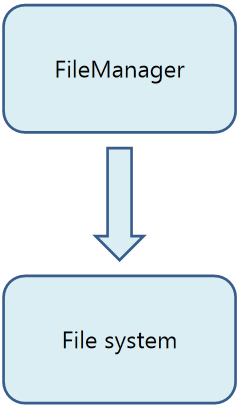
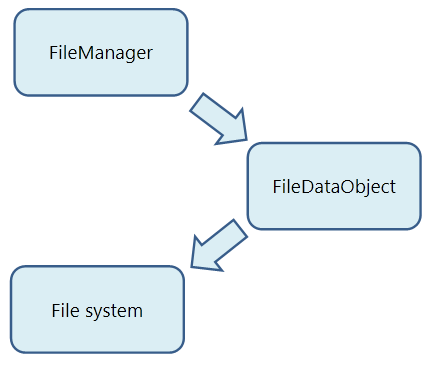
Внешняя зависимость – объект в системе, с которым взаимодействует тестируемый код, и который невозможно контролировать. ( Например, файловая система, потоки, память, службы и т. д. )

**Проблема:** Тест для FileManager’a будет зависить от файловой системы, сответственно будет является интеграционным, так как мы будет тестировать не только FileManager, но и файловую систему (наличие файла, правильность формата файла и т.д.)

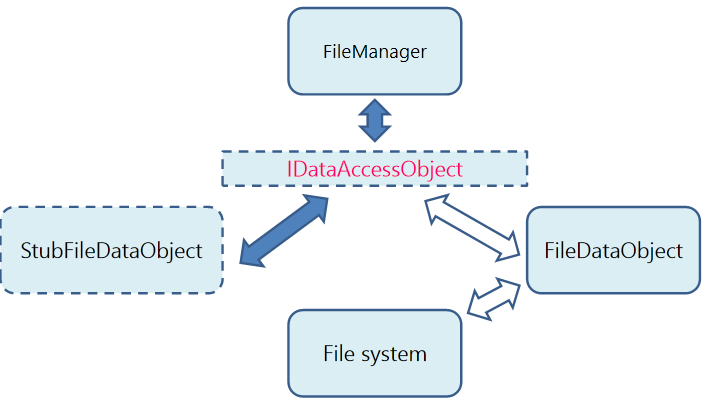


Введение косвенного слоя, для избежания прямой зависимости от файловой системы. Код, который работает с файловой системой разделяется на FileDataObject, который в будущем будет заменен с stub объектом в тесте.



**Создание stub объекта.**

Новый интерфейс позволит сделать объектную модель, чтобы абстрагироваться от операций класса FileDataObject, и создать тест с заглушкой.



Stub-объект (заглушка) - это управляемая замена существующих зависимостей в системе. Stub-объект позволяет тестировать код без использования внешних зависимостей.

Как правило, внешняя зависимость появляется в следующих случаях:

1. При создании объекта явно указывается класс.

2. Зависимость от аппаратных/программных платформ.

3. Зависимость от представления или реализации объекта.

4. Зависимость от алгоритмов.

Сильная зависимость порождает следующие проблемы. Систему сложно:

1. Поддерживать

2. Расширять

3. Понимать

4. Тестировать

Dependency Injection – паттерн описывающий технику внедрения внешней зависимости программному компоненту.

Преимущества использования Dependency Injection:

1. Разделение конфигурирования и использования объектов

2. Уменьшается связь между объектами. Конкретные объекты проще заменить.

3. Увеличение мобильности модулей

4. Систему проще сопровождать и тестировать.

Способы внедрения зависимости:

1. Внедрение через интерфейс.

2. Внедрение через свойство. Используется если зависимость имеет опциональный характер.

3. Внедрение через конструктор. Проблемно использовать, если для правильной работы тестируемого класса требуется несколько Sub-объектов. В этом случае приходится создавать или множество конструкторов, или же один конструктор с множеством параметров.

Способы создания экземпляров зависимостей:

1. Через контейнеры.

2. С помощью Ninject.

Inversion of Control (IoC) – абстрактный принцип, описывающий способы написания слабосвязанного кода. Dependency Injection – один из способов реализации данного принципа.

DI container – набор объектов, позволяющая упростить и автоматизировать процесс написания кода с использованием принципа Inversion of Control.

Для решения проблем инкапсуляции членов, созданных исключительно в целях тестирования можно использовать директивы препроцессора #if, #elif, #endif а так же атрибут InternalsVisibleToAttribute.

Как правило, типы и элементы с модификатором доступа internal доступны только в сборке, в которой они определены. Атрибут InternalsVisibleToAttribute делает их также видимыми для типов в указанной сборке, которая называется "дружественная сборка".