16 вопрос. **TDD. 3 закона TDD.**

Модульное тестирование, или юнит-тестирование (англ. *unit testing*) — процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы, наборы из одного или более программных модулей вместе с соответствующими управляющими данными, процедурами использования и обработки.

Идея состоит в том, чтобы писать тесты для каждой нетривиальной функции или метода. Это позволяет достаточно быстро проверить, не привело ли очередное изменение кода к регрессии, то есть к появлению ошибок в уже оттестированных местах программы, а также облегчает обнаружение и устранение таких ошибок.

**TDD** (Разработка через тестирование, англ. test-driven development) — техника разработки программного обеспечения, которая основывается на повторении очень коротких циклов разработки: сначала пишется модульный тест, покрывающий желаемое изменение, затем пишется код, который позволит пройти модульный тест, и под конец проводится рефакторинг нового кода к соответствующим стандартам.

**Разрабатывая функцию вы будете проходить два этапа**: **проектирование интерфейса и реализацию.** На стадии написания теста вам нужно понять, каким будет интерфейс вашей функции, что она будет получать на вход и что она будет возвращать. Каким будет формат входных и выходных данных. Спроектировав интерфейс и написав отказной тест, вы напишите простейшую реализацию, ограничения которой уже заданы в тесте. Это путь к простым и читаемым решениям. Вам остаётся только описать функцию в документации проекта.

**Три закона TDD.**

**1. Новый рабочий код пишется только после того, как будет написан модульный тест, который не проходит.**

**2. Вы пишете ровно такой объем кода модульного теста, какой необходим для того, чтобы этот тест не проходил (если код теста не компилируется, считается, что он не проходит).**

**3. Вы пишете ровно такой объем рабочего кода, какой необходим для прохождения модульного теста, который в данный момент не проходит.**

Эти три закона заставляют вас использовать рабочий цикл продолжительностью около 30 секунд. **Сначала вы пишете маленькую часть модульного теста. За эти считанные секунды вы упоминаете в коде имя класса или функции, которые еще не были написаны; естественно, модульный тест не компилируется. Следовательно, далее вы должны написать рабочий код, с которым тест откомпилируется. Но писать больше кода нельзя, поэтому вы переходите к написанию дополнительного кода модульного теста.**

**Цикл прокручивается снова и снова. Добавляем небольшой фрагмент в тестовый код. Добавляем небольшой фрагмент в рабочий код. Два кодовых потока растут одновременно, превращаясь во взаимодополняющие компоненты.** Соответствие между тестами и рабочим кодом напоминает соответствие между антителом и антигеном.