1. Что такое процесс тестирования? Для чего?

**Модульное тестирование (unit testing, юнит-тестирование)** – процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы.

**Назначение** ► Улучшают качество архитектуры приложения. ► Стимулируют написание простых методов. ► Упрощают интеграцию кода. ► Помогают документированию кода. ► Минимизируют зависимости в системе. ► Отладка и рефакторинг (скорость, повторный запуск)

1. Что такое TestNG и Junit?

**Unit** и **TestNG**, несомненно, являются **двумя** наиболее популярными фреймворками для модульного тестирования (юнит-тестирования) в экосистеме Java. Фреймворки модульного тестирования в Java

► JUnit (3, 4, 5) https://junit.org/junit5/ ♣ набор расширений – таких как JMock, HtmlUnit и т.д. ♣ Портирован на другие языки: PHP, C#, Python, Delphi, Perl, C++, JavaScript и т.д.

► TestNG https://testng.org/doc/ ♣ широкие возможностями по созданию тестов и проверок, управлению выполнением тестов ♣ Не имеет аналогов вне Java

1. Что такео юнит тестирование и какие его основные признаки?

**Unit** — модульные тесты, применяемые в различных слоях приложения, тестирующие наименьшую делимую логику приложения: например, класс, но чаще всего — метод. Эти тесты обычно стараются по максимуму изолировать от внешней логики, то есть создать иллюзию того, что остальная часть приложения работает в стандартном режиме.

Ваши тесты должны:

* Быть достоверными
* Не зависеть от окружения, на котором они выполняются
* Легко поддерживаться
* Легко читаться и быть простыми для понимания (даже новый разработчик должен понять **что именно** тестируется)
* Соблюдать единую конвенцию именования
* Запускаться регулярно в автоматическом режиме

4. Что таое тест-кейс?

Тест-кейс — это профессиональная документация тестировщика, последовательность действий направленная на проверку какого-либо функционала, описывающая как придти к фактическому результату.

Набор тест-кейсов называют тест-комплектом. Иногда тест-набор путают с тест-планом. Тест-план описывает какие работы, как и когда должны быть проведены в рамках тестирования продукта, а так же что необходимо для их выполнения.

Тест-кейс — это профессиональная документация тестировщика, последовательность действий направленная на проверку какого-либо функционала, описывающая как придти к фактическому результату.

Набор тест-кейсов называют тест-комплектом. Иногда тест-набор путают с тест-планом. Тест-план описывает какие работы, как и когда должны быть проведены в рамках тестирования продукта, а так же что необходимо для их выполнения.

5. Назначение @Test аннтоции?

**Это самая важная аннотация в TestNG, в которой находится основная логика теста**. Все автоматизируемые функции находятся в методе с аннотацией @Test. Она имеет различные атрибуты, с помощью которых может быть настроен запуск метода.

6. Что такое test suit? Как его создать?

**Test** Suite– это группа тест-кейсов (**test** case), объединенных по некоторому принципу, и предназначенных для проверки определенной части программы. Для формирования **test** suite, нужно чтобы файлы тест-кейсов находились в одном каталоге с собравшим их **test** suite.  С помощью Test Suite он получает универсальный инструмент тестирования — все задачи, которые у него есть он может решать в одном инструменте с единой методологией и единым подходом, не вдаваясь в детали реализации. Тест сьют - это связка тест кейсов в один набор. Эта фича реализована и в JUnit 4 и в TestNG, но сделано это по-разному.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | <suite name="TestSuite">  <test name="AutoQA">  <classes>  <class name="automationFramework.TestNG" />  </classes>  </test>  </suite> |

7. Что обеспечивает Assert class?

Assert — это специальная конструкция, позволяющая проверять предположения о значениях произвольных данных в произвольном месте программы.

Эта конструкция может автоматически сигнализировать при обнаружении некорректных данных, что обычно приводит к аварийному завершению программы с указанием места обнаружения некорректных данных.

8. Поясните назначение @BeforeSuite / @AfterSuite @BeforeTest / AfterTest @BeforeClass / @AfterClass @BeforeMethod / @AfterMethod?

1. @BeforeSuite – указывает, что данный метод будет запускаться перед любым методом тестового класса.

2. @BeforeGroups – аннотирует методы, которые будут выполняться перед первым методом в любой из указанных групп.

3. @BeforeClass – указывает, что метод будет выполнен до всех тестовых методов тестового класса.

4. @BeforeTest – аннотированный метод будет запускаться до всех тестовых методов.

5. @AfterTest – аннотированный метод будет запущен после всех тестовых методов, принадлежащих классам внутри тега <test>.

6. @BeforeMethod – аннотированный метод будет выполняться перед каждым тестовым методом.

7. @AfterMethod – аннотированный метод будет запускаться после каждого тестового метода.

8. @AfterClass – аннотированный метод будет запущен после всех тестовых методов в текущем классе.

9. @AfterGroups – аннотируется методы, которые будут выполняться после всех методом в любом из указанных групп.

10. @AfterSuite – указывает, что данный метод, будет запускаться после всех методов тестового класса.

9. Как и для чего используют параметризацию в тестировании?

**Параметризованные** **тесты** - это практика написания **тестов**, когда один и тот же **тест** выполняется на разных наборах входных данных. **Параметризация** означает многократный запуск одного и того же метода .

10.Что такео TDD и DDT

* **TDD** — ну, это все знают, сначала пишем тесты, а потом остальной код.

**TDD** — это методология разработки ПО, которая основывается на повторении коротких циклов разработки: изначально пишется тест, покрывающий желаемое изменение, затем пишется программный код, который реализует желаемое поведение системы и позволит пройти написанный тест. Затем проводится рефакторинг написанного кода с постоянной проверкой прохождения тестов.

Разработка через тестирование (TDD) — это подход к разработке программного обеспечения, при котором тестовые примеры разрабатываются для определения и проверки того, что будет делать код. … Полная форма TDD — это разработка через тестирование. Простая концепция TDD заключается в написании и исправлении неудачных тестов перед написанием нового кода (до разработки).

**Data** **Driven** **Testing** (DDT) – подход к созданию/архитектуре автоматизированных тестов (юнит, интеграционных, чаще всего применимо к backend тестированию), при котором тест умеет принимать набор входных параметров, и эталонный результат или эталонное состояние, с которым он должен сравнить результат, полученный в ходе прогонки входных параметров.