**JUnit - Lifecycle of a Test Class**

JUnit framework provides following basic lifecycle annotations (all are method level annotations).

1. @BeforeClass:
   * This annotation must be used on public static void no-arg method.
   * The method is invoked before an instance of this test class is created and any tests are invoked.
   * This life cycle callback is not related to JVM class loading. JUnit framework calls this method explicitly before calling constructor/other methods.
   * This method is called only once, whereas, other instance lifecycle methods are called every time before calling each test method.
   * This annotation is useful for initializing static resources which would, otherwise, be expensive to create during each test invocation.
2. @AfterClass: Similar to @BeforeClass but is called at the very end of all test/other lifecycle methods. It is called only once. Useful for static resource clean up.
3. @Before:
   * It should be used on public void no-arg instance method.
   * It is invoked every time before each test method invocation.
   * Used to setup instance variables/resources which can be used during a test method execution. Useful to avoid code duplication and/or when there are many indirection of method calls starting from the target test method.
4. @After: Similar to @Before but runs after target test method execution. Useful for cleaning up instance resources.
5. @Test: public void methods to perform tests. This is where we perform one or more assertions by using static methods of org.junit.Assert. Assert methods throw org.junit.AssertionError on assertion failure. This exception or any other exception is reported as test failure. If no exceptions are thrown then the test will pass.

**How do you use test fixtures?**

Fixtures is a fixed state of a set of objects used as a baseline for running tests. The purpose of a test fixture is to ensure that there is a well known and fixed environment in which tests are run so that results are repeatable. It includes −

* setUp() method which runs before every test invocation.
* tearDown() method which runs after every test method.

**What is a test suite?**

* Test suite means bundle a few unit test cases and run it together. In JUnit, both @RunWith and @Suite annotation are used to run the suite test.