***JUnit Notes***

1. JUnit 5
   1. Core: Platform
      1. The arch of JUnit starts with platform
      2. It is the core of what comprises the lib of JUnit
      3. Involved in running the tests
      4. Way to call the test runner
      5. Provides execution context
   2. JUnit API - Jupiter
      1. Thing that we’re interacting with while writing code
      2. Tells the Platform –
         1. What is to run?
         2. How to run?
   3. Vintage
      1. Set of APIs which can be used when using older JUnit tests
      2. JUnit is not backward compatible out-of-the-box, New tests – Jupiter & Old tests – Vintage
   4. Extentions (3rd Party)
      1. Use your own APIs, & still use the underlying test engine i.e Platform to run your test.
   5. @Test annotation
      1. The IDE is talking to the platform and running those tests.
   6. Assertions
      1. Meaning – what is assumed as a fact is actually true
      2. Assert : expected = actual
   7. Test driven development
      1. Write the test case -> write actual code -> test will fail -> write code -> green test
   8. maven-surefire-plugin
      1. maven integration for all the tests, but it is standard in industry, and more powerful.
   9. Test life cycle
      1. The process in which the test instance is created, gets managed and gets destroyed
      2. Junit is managing the life cycle of this class
      3. Junit creates a new class instance for every test run, for every method run rather
      4. Note: we shouldn’t order the tests, or make them dependent on each other in any way
   10. Conditional Execution
       1. Is a way to execute tests only in certain situations/scenarios
       2. @EnabledOnOs(OS.LINUX)
          1. Stick on top of the testClass and that testClass will run only on that OS.
          2. But, if you run it on other OSs , it’ll be disabled, hence will not run, neither fail
       3. @EnabledOnJre(JRE.JAVA\_11)
          1. This comes handy because, we’ve certain language features that works only with certain JREs.
       4. @EnabledIf
       5. @EnabledIfSystemProperty
       6. @EnabledIfEnvironmentVariable
          1. These three, lets you control from outside what test we want to executed and during which time
          2. For eg, we’ve tests that we need to run only on certain nodes in our build system and not on certain nodes, we can set a system property and say, run only the dev test, prod test etc.
   11. @AssertAll
       1. When, all we’re doing in each test is assertions, then rather than doing assert statements all the way to many test methods, we can use assertAll once
       2. Way to run a bunch of assertions in just one statement
       3. If any of the assertions fail, mark it as failure, when we’re just running a bunch of test cases
   12. @Tag(“tagName”)
       1. We can tag the test methods or classes of same type with same tagName.
       2. There is a configuration to run only the tagged classes or methods inside the Configuration of IntelliJ or Eclipse
          1. We can select exclude or include tags from there
       3. Or we can also, configure it use Maven pom.xml file

<plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-surefire-plugin</artifactId>  
 <version>2.22.2</version>  
 <configuration>  
 <groups>Math</groups>

<excludedGroups>Circle</excludedGroups>  
 </configuration>  
</plugin>

* + - 1. Here, we’ve added the tagName – “Math” to the includeTag
      2. The excludedGroups is the excludeTag
  1. TestInfo & TestReporter
     1. TestInfo –
        1. Contains info about the test
        2. we can get metadata about the test
     2. TestReporter –
        1. lets you access the log/console where JUnit is reporting errors or anything and it lets you add to the reporting like, some messages to the final JUnit report
     3. Both of these are JAVA Interfaces that JUnit maintains, and can be accessed by Dependency Injection