Junit 5 Notes

Introduction:

Junit 5 is most popular testing framework for java applications. In Java 8 very notable changes has happened like lambda expression. Junit5 aims to adapt java8 style of coding and several other features as well, that's why java8 requires to create and executes test in Junit5.

Junit 5 is composed of several different modules as follows:

> *JUnit 5=JUnit Platform + JUnit Jupiter +JUnit Vintage*
>
> *JUnit Jupiter: It includes new programming extension and models for writing test*
> *JUnit Platform: To be able to launch JUnit tests, IDEs, build tools or plugins need to include and extend platform APIs*
> *JUnit Vintage: It is to provide support for JUnit3 and JUnit4 as backward compatibility*

JUnit Annotations:

`@BeforeEach`	*The annotated method will be run before each test method in the test class.*
`@AfterEach`	*The annotated method will be run after each test method in the test class.*
`@BeforeAll`	*The annotated method will be run before all test methods in the test class. This method must be static.*
`@AfterAll`	*The annotated method will be run after all test methods in the test class. This method must be static.*
`@Test`	*It is used to mark a method as junit test*
`@DisplayName`	*Used to provide any custom display name for a test class or test method*
`@Disable`	*It is used to disable or ignore a test class or method from test suite.*
`@Nested`	*Used to create nested test classes*

`AfterEach():`

```
@AfterEach
public void cleanUpEach(){
    System.out.println("After Each cleanUp() each method called");
}
```

BeforeAll():

```
@BeforeAll
public static void init(){
    System.out.println("BeforeAll init() method called");
}
```

BeforeEach():

```
@BeforeEach
public void initEach(){
    System.out.println("BeforeEach initEach() method called");
}
```

RepeatedTest():

```
@RepeatedTest(5)
void addNumber(TestInfo testInfo) {
    Calculator calculator = new Calculator();
    Assertions.assertEquals(2, calculator.add(1, 1), "1 + 1 should equal 2");
}
```

Disabled():

```
@Disabled
public class AppTest {

    @Test
    void testOnDev()
    {
        System.setProperty("ENV", "DEV");
        Assumptions.assumeFalse("DEV".equals(System.getProperty("ENV")));
    }

    @Test
    void testOnProd()
    {
        System.setProperty("ENV", "PROD");
        Assumptions.assumeFalse("DEV".equals(System.getProperty("ENV")));
    }
}
```

JUnit 5 Expected Exception:

To test methods which throws exception, we use `assertThrows()` method from `org.junit.jupiter.Assertions` class.

```
class Assert_ThrowExceptionTest {
    @Test
    void giveVoteThenReturnNull() {
        Assertions.assertThrows(InvalidAgeException.class,()->{new Assert_ThrowException().giveVote( age: 15);
        } , message: "It should check the voter");
    }

    @Test
    public void whenExceptionThrown_thenAssertionSucceeds() {
        Exception exception = assertThrows(NumberFormatException.class, () -> {
            Integer.parseInt( s: "1A");
        });

        String expectedMessage = "For input string";
        String actualMessage = exception.getMessage();

        assertTrue(actualMessage.contains(expectedMessage));
    }
}
```

JUnit 5 Assertions Examples:

JUnit 5 assertions help in validating the expected output with actual output of a test case. To keep it simple, all methods assertions are static methods in `org.junit.jupiter.Assertions` class.

Assertions.assertEquals() and Assertions.assertNotEquals()
Assertions.assertArrayEquals()
Assertions.assertIterableEquals()
Assertions.assertLinesMatch()
Assertions.assertNotNull() and Assertions.assertNull()
Assertions.assertNotSame() and Assertions.assertSame()
Assertions.assertTimeout() and Assertions.assertTimeoutPreemptively()
Assertions.assertTrue() and Assertions.assertFalse()
Assertions.assertThrows()
Assertions.fail()

Testing:

- *Unit Testing:* The Smallest testable part of an application, called unit testing and independently tested for proper operation.
- *Integration Testing:* Individual software modules are combined and tested as a group, it takes place after unit testing.

Testing tools:

1. ***JUnit***
2. *Mockito*
3. *Hamcrast*
4. *TestNG*
5. *Power Mock*
6. *Spock*

Black Box Testing

White Box Testing

Black Box Testing *is a software testing method in which the internal structure/ design/ implementation of the item being tested is not known to the tester*

White Box Testing *is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.*