JUNIT 5

Date 27/08/19

- has Saparate APIs for different concerns.

  i.e to run, to write tests

- composed of

  - Junit plateform
  - Junit jupiter
  - Junit vintage (for Junit 4)

- Dependencies required for Junit ,tests

  - Junit API (For compilation)
  - Jnit Engine (to RUN)

- Default setting

  → Test file name should have "test" in it.

  → because by default gradle plugin & console

Date

pick test classes only ends with "test".

→ Phases of TEST

1) Arrange - set the state

2) Act - call the function/ action

3) Assert - check expected outcome with actual outcome

4) Annhilation - reset the state

→ Annotations

1) @ BeforeEach - executes the method before each "test".

2) @ AfterEach - executes method after each "test".

Date

3) @BeforeAll - Before All the tests executed.

4) @AfterAll - It will execute after all the tests executed.

→ When we set Lifecycle to "per_method" mode @BeforeAll will be executed even before the constructor

then constructor
then beforeEach
then test
then afterEach

this lifecycle will repeat
And atlast AfterAll will be executed.

Date

→ When we set Lifecycle to "per_class." mode.

Constructor will be executed before all the tests.
after that @BeforeAll will be called
then BeforeEach
then test
then afterEach
. . . -
And at last AfterAll.

→ Assertion libraries (extend)
→ AssertJ
→ Hemcrest

→ Assertequals Syntax

assertEquals (Expectedvalue, actualvalue, (optional)string when assertion fails).

Date

→ AssertAll Syntax.

```
assertAll (heading: "message
    for error",
() → assertNotNull (info),
() → assertEquals (ex, act)
() → assertEquals (exp, ac)
);
```

→ Assert Throws checks whether the exception is thrown or not. If not it will fail.

Syntax:

assertThrows (RuntimeException.class, method);

We can specify exception or exception type.

Date

→ Assert timeout

- checks if the given expression executed before the given timeout is exceeded.

Syntax : assertTimeout
(Duration.ofMillis(4)
, expression);

→ DisabledAnnotation

- for methods / Classes.
- to ignore its execution.

→ Assumptions - abort the test when failed.

Date

→ Parameterization test

→ annotation : @ Parameterized
test

→ need to add a ~~dependency~~
dependency

- groupid : org.junit.jupiter
Artifact id : junit-jupiter-
params
version : 5.0.1

→ atleast 1 source must be
provided.

∴ SOURCE ANNOTATIONS.

1) @ValueSource
- allows to define array
of String, int, long
double
- used with methods
having single parameter

2) @EnumSource
- runs test with value of enum
- used with single parameter methods.

3) @MethodSource
- specify names of methods that provide data for test
- used for single parameter test.
- returns stream of parameter type / primitive types.

- also used with multiple parameter methods.
- there it returns
→ Stream, iterable, iterator,
- must be static

Date

4) @CsvSource -

- comma separated String literals.
- takes parameter → delimiter.

5) @CsvFileSource
- takes csv files
- parameters needed.
  - encoding
  - Line Separator
  - Delimiter

6) @ArgumentsSource.

- specify your own sources

Date

Junit 4	Junit 5
1) need to write access specifiers - public.	1) dont need to specify access specifiers.
2) Annotations used: - Before - After - BeforeClass - After class - @Ignore	2) Annotation used. - BeforeEach - AfterEach - BeforeAll - AfterAll - @Disabled
3) used @Category annotation to choose and run selected tests	3) we use @Tags("String")
4) assertion package is under class → org.junit.Assert	4) assertion methods now belongs to org.junit.jupiter.api.assertion.

Date

5) assert Equals
("message", expected
value, actual value)

5) assert Equ
(exp value,
actual val,
"message");