**Rodzaje testów:**

Passed, NotPassed, Ignore

@Ignore(„powód”)

@Test

Ignore można również dawać nad klasą.

@Test(except=”ArithmeticException.class”) – gdy spodziewany się, że będzie wyjątek. Nie używamy wtedy asercji.

Assert.true(message,actual,excepted)

Fajnie jest dawać message.

**TESTY POWIĄZANE**

@RunWith(Suite.class) – to dajemy przed klasą, która uruchamia jakiś zestaw testów

Uruchamianie testów w jakiś sposób

@Suite.SuiteClasses({Pierwsza.class,Druga.class})

Te obie rzeczy przed klasą. I będzie uruchomiony zestaw testów

A więc mamy: przypadek testowy (metoda), zestaw testów (klasa) oraz testy powiązane.

W klasie testującej można użyć

@BeforeClass – metoda wykonywana przed rozpoczęciem testów

@Before – metoda wykonywana przed każdym testem

@AfterClass - metoda wykonywana po zakończeniu testów

@After - metoda wykonywana po każdym teście

**Zaślepiania otoczenia**

Mockito

Mockito.when(…).thenReturn(…)

Mockito.mock(…), Mockito.spy(….)

When using **mock objects**, the default behavior of the method when not stub is **do nothing**. Simple means, if its a void method, then it will do nothing when you call the method or if its a method with a return then it may return null, empty or the default value.

While in spy objects, of course, since it is a real method, when you are not stubbing the method, then it will call the real method behavior. If you want to change and mock the method, then you need to stub it.

Zamkowanie to przygotowanie klasy tak, aby można było podmieniać implementacje metod.

**Testy parametryzowane**

Nad klasą z testami należy napisać

@RunWith(Parametrized.class)

Metodę dostarczającą dane do testu parametryzowanego oznaczamy @Parametrized.Parameters

Ctrl-Shift-J – zamiast wielu wierszy będzie jedna linia

**Junit 5 (Jupiter)**

Możemy tutaj stosować wyrażenia lambda

@BeforeEach, @AfterEach, …

Testy warunkowe (NOWOŚĆ)

Nie ma już Assert a jest Assertions

Wyjątki: teraz będzie assertThrows(wyjątek,lambda)

Testy wielokrotnie powtarzalne: @RepetedTest(liczba-razy)

Jak sprawdzić który to już raz? Dać w arg metody RepetitionInfo info i w ciele info.getCurrentRepetition()

Zamiast @Ignore jest @Disabled

assertAll() kolekcje asercji do sprawdzenia – najlepiej dawać jako wyrażenia lambda

nazwa testu - @DisplayName(„Nazwa”)

**@RepetedTest(value=10,name=”{displayName} {currentRepetition}/{totalRepetition}”)**

**@DisplayName(„Nazwa”)**