**Paraméterezett tesztek**

Ha ugyanazt a tesztesetet különböző adatokkal akarjuk lefuttatni

a paramétereket megadhatjuk a kódban, de akár külső forrásból (.csv állományból is)

még csak experimental (kísérleti fázisban volt, van?)

használatához függőséket kell felvenni:



**Egyszerű értékekkel**

@ParameterizedTest és @ValueSource annotációk kellenek

ValueSource annotációban soroljuk fel a különböző érétkeket, tömb literállal

további annotációk a különböző paraméterekkel történő meghívásokra:

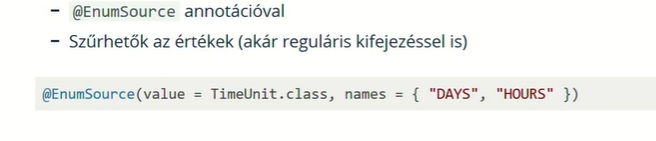
@NullSource @EmptySource, @NullAndEmptysource

tesztesetek nevében szereplni fog a paraméterérétk is

**Enum**

@Enumsource - enum értéke mint paraméter

szűrhetőek az értékek vagy explicit adjuk meg

****

**Methodsource**

lehetőség van továbbá a paraméterek nem explicit felsoorálására, hanem metódus állítsa őket elő

@Methodsource annotáció; hivatkozás egy statikus metódusra

@ParameterizedTest annotációval megadható, hogy az eset milyen néven jelenjen meg

visszaadhat Stream, Collection, Iterator, Iterable példányt

**Methodsource példa – FONTOS!**

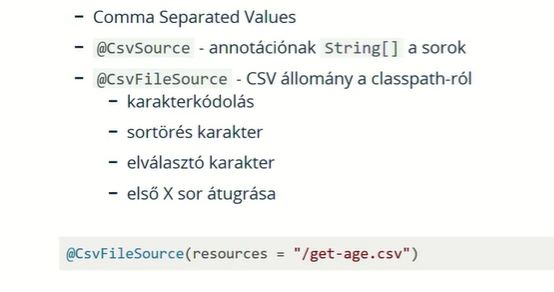


a createAges() statikus metódus egy Argument streamet ad majd vissza; ezekkel az értékekkel fogja meghívni a testGetAge() metódust

**CSV**

@CSVSource annotáció; ilyenkor a kódban át kell adni String tömbként az állományt

@CSVFileSource annotáció – ténylegesen fileból; sok mindent meg lehet neki adni

****

**Tempdir extension**

tesztelésnél problémát okozhat a fileműveletke tesztelése, ha az adott funkció filet ír ki és megvizsgáljuk hogy a kiírt tartalom megegyezik e az elvárttal, vagy mi hozunk létre egy példa input állományt, felolvastatjuk az adott funkcióval és ellenőrizzük az eredményt

az input és output állományt hol helyezzük el? ez már egy újabb komponenes, filerendszer bevonása, ez már inkább integrációs teszt

a jUnit @Tempdir annotációja a teszteset lefutása előtt létrehoz egy könyvtárat, az adott P rendszer TEMP könyvtárán belül; a teszteset lefutása után pedig ezt a könyvtárat letörli

Menet:

1.Tempdir típusú atri létrehozása @Tempdir annotációval

2. Futáskor a jUnit ezt érzékeli; létrehozza az ideiglenes könyvtárat; majd ez alapján létrehoz egy PAth objektumot és erzt értékül adja dependency injectionnel a tempdir változónak

3. Lefuttatja a tesztesetet majd a könyvtár letörlésre kerül



a metódus szöveges fileba kiírja az alkalmazott adatait; a tempDir könyvtáron belül egy john- doe.txt-re mutató referenciát hozunk létre majd átadjuk a write metódusnak és egy példányosított alkalmazottat is írásra

assertEqualssal összehasonlítjuk az elvárt eredményt a fileból kiolvasott tényleges adattal; ezzel vizsgáljuk, hogy az írási művelet helyesen lett-e implementálva