**Tesztesetek életciklusa**

**(előkészület = test fixture=?, kiemelés, hasznosság, annotációk)**

valamiféle felkészülés, előkészülés szükséges lehet a teszteset lefuttatásához, pl. példányosítás és valamilyen állapotba hozás ->

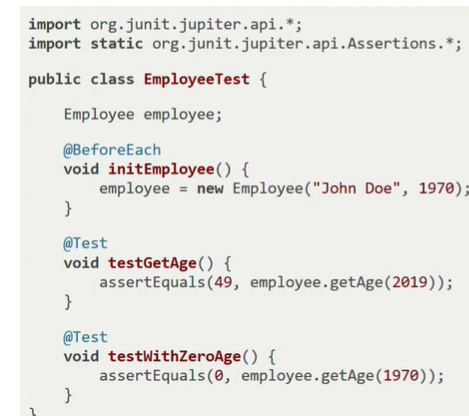
ez az előkészület a **test fixture,** amikor **az adott környezetet a megfelelő állapotba hozzuk**

a test fixture **kiemelhetjük külön metódusba** -> hasznos, mert **elkülönül az esettől és újra fel lehet használni** több esetre is -> erre vannak **annotációk**

**@BeforeEach, @AfterEach ->** mindegyik előtt és után

**@BeforeAll, @AfterAll ->** egyszer fut le az adott tesztosztály betöltődésekor még a tesztek előtt

beforeAll és AfterAll csak egyszer fut le, ezért a metódusa static kell legyen!

**Test fixture példa**

**Test fixture külön interfészben**

fixtureok külön interfészben, mint default metódusok

****

**Izoláltság**

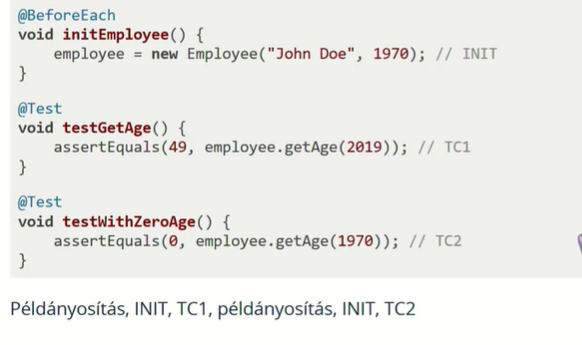
**(=függetlenség, függéssel a problémák, állapotátmenet, teszt osztály attribútumának állapotátmenetét a jUnit hogy oldja meg, statikus atrik)**

íráskor törekedni kell az esetek izoláltságára, **függetlenségére**, hogy ne legyenek egymásra hatással

ha függenének egymástól, az **esetek futtatásának sorrendje sem lehetne véletlen**, ill. megnehezítené a hibakeresést is ez

**ne lehessen állapotátmenet az esetek között;** állapotátmenete csak a tesz osztályban felvett attribútumnak lehet, mert ezt a fajta állapotátmenetet a jUnit megoldja -> **minden teszt eset előtt újrapéldányosítja az adott teszt osztályt**, újra felveszik az atrik a kezdőértéket(újra inicializálódnak)

**statikus attribútum** ha van az eredeti osztályban, az megtartaná az állapotát teszt esetek között, **nem szerencsés**



**Metódusok futtatásának sorrendje**

**(uaz de nem tudható, miért nem baj, integrációsnál kellhet, @TestMethodOrderer és MethodOrderer példányok ->Alphanumeric, OrderAnnotation, Random)**

mindig ugyanaz, de nem magától értetődő, nem lehet rá építeni, de az izoláltság betartásával nem is lenne rá szükség

integrációsnál lehet szükség egy pontos sorrendre, ezt a **@TestMethodOrder** annotációval tudjuk megadni és egy **MethodOrderer** implementációval kell paraméterezni; vannak előre elkészítettek:

MethodOrderer.Alphanumeric – abc sorrendben

MethodOrderer.OrderAnnotation – tehetünk a teszt esetekre @Order annotációt, amit egész számokkal paraméterezhetünk

MethodOrderer.Random – véletlenszerű; jó módszer az izoláltság vizsgálatára

**Disabled**

**(esetre vagy egész teszt osztályra, @Disabled, kikommentelés helyett, csak ideiglenes használat))**

bizonyos teszt esetek kikapcsolására, ne lehessen lefuttatni

az adott osztályra vagy teszt metódusra kell rátenni a @Disabled annotációt

opcionálisan megadható üzenet

inkább ezt használjuk, mintsem kikommenteljük a teszt esetet

inkább ne, vagy csak ideiglenesen használjuk!

**Conditional test execution**

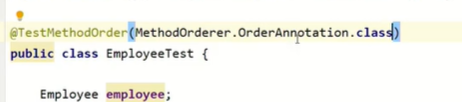
**(feltételhez kötve, @Disabledxxx(paraméter); feltételre példák)**

a kikapcsolást/végrehajtást feltételhez is tudjuk kötni

pl: OP rendszer, JVM verziószám, op rendszerben beállított környezeti változó, Java környezeti változó

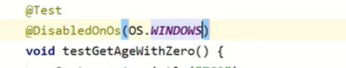
@Disabled\*xxx annotációk, paraméterrel, pl: @DisabledOnOs(WINDOWS)

**Tesztesetek életciklusa a gyakorlatban (gyakorlat)**

****

****

****

****

**Kérdések**

*Mi az a test fixture?*

*Miyen annotácókat használunk arra, hogy a text fixture kiemeljük külön metódusba? Mikor futnak le ezek a metódusok?*

*Hogy rakjuk a test fixturet külön interfészbe?*

*Miért törekszünk a tesztek izoláltságára?*

*Hogy oldja meg a jUnit azt, hogy ne legyen állapotátmenet a tesztesetek között? Minek lehet még így is állapotátmenete?*

*Milyen sorrendben futnak le a tesztek alapesetben?*

*Mikor lehet szükség megfelelő sorrend szeirnit futtatásra?*

*Milyen annotációt haszálunk a sorrendi futtatáshoz és milyen implementációkkal paraméterezhetjük?*

*Milyen annotációval tudunk tesztet vagy egész teszt osztályt kikapcsolni?*

*Hogyan tudjuk feltételhez kötni a végrehajtást? Milyen implementációi vannak?*

*Hogyan tudjuk személyre szabni a tesztesetke neveit?*