**Assert**

**(mire jó;, statikus import, null, equas, notequals, assertSame, assertTrue)**

vele tudjuk megvizsgálni, hogy az **az elvárt eredmény megegyezik-e a megkapott aktuális eredménnyel**

metódusok ezekre az ellenőrzésekre; statikus metódusok; importálás után elég csak a nevét használni:

**Assert metódusok**

assertNull(), aseertNotNull()

a paraméterként átadott referencia null vagy nem null-e?

assertEquals() és assertnotEquals()

használható pirmitívre és objektumra egyaránt; ref típus esetén az equals() metódust hívja

assertSame(), assertNotSame()

ha szigorúan a referencia egyenlőséget akarjuk vizsgálni (=) objektumoknál, nem pedig az adott osztályban deklarált equals()-t

assertTrue(), assertFalse()

paraméterként boolean értéket kifejező kifejezést adhatunk át; nem javasolt, a true vagy false nem írja ki az elvárt eredményt is

fail() – ha explicit el akarjuk buktatni a teszt esetet; ha olyan catch ágba kerül a teszt metódus pl. ami nem nyerő?

**AssertEquals lebegőpontos számok esetén**

harmadik paramétert is meg lehet adni

delta paraméter = hibahatár; számábrázolási pontatlanságok miatt

assertEquals(1.0, 1.0, 0.005);

**Assert metódusok több elem esetén**

kollekció vagy tömbök összehasonlítására

assertArrayEquals töbökre

assertIterableEquals Iterable -> kolelkciókra

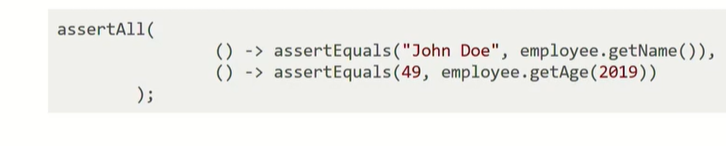
assertLinesMatch List<String> - aktuális érték, regulás érték, nem pontos egyezőség?

**Egyszerre több Assert**

**(cél: egy metóduson belül több assert úgy, hogy mind ki legyen értékelve elbukás esetén is; AssertAll())**

lehetőség van egyszerre több assertet is megadni

AssertAll() metódusnak paraméterül átadunk két assertEquals() hívást lambda kifejezéssel; mindkettő meg lesz hívva és ki lesz értékelve, ha pl. mindkettő elbukik mindkettő meg fog jelenni a logban is



**Egymásba ágyazhatóság**

két párhuzamosan futó részre bontja az assertet (egymástól függetlenek); egyik a nevet, másik az életkort vizsgálja; ha a név null, már nem megy tovább, de megvizsgálja?

**Assert kollekciókon**

**(korlátozott; streamek használata -> list of és map**

korlátozott jUniton belül, bonyolultakat nem nagyon lehet vele anélkül, hogy ne konvertálnánk, mert az equals() implementáció stb. bekavar; inkább streamekkel átkonvertálunk pl. csak a String nevekre és csak azt vizsgáljuk

****

**Assert**

**(korlátozott; 3rd party library kiterjesztések; Hamcrest, AssertJ, speciálisak)**

jUnitnál korlátozott(pl kollekcióknál); ezért vannak libraryk, amik kiterjesztik

általános célú library: Hamcrest(oldschool, de sok régi projektben ott van; egy időben a jUnitnak függősége is volt)

modernebb: Assertj

Speciális célúak:

XMLUnit(XML összehasonlítás struktúrális egyezőség alapján);

JSONassert

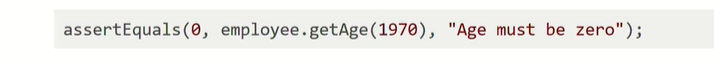
DbUnit – integrációs teszteknél adatbázis tartalmak összehasonlítására

**Alternatív üzenet**

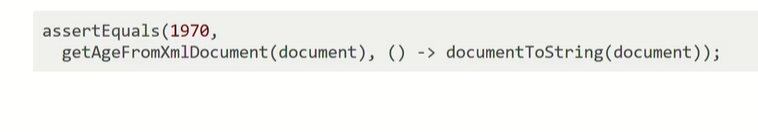
**(saját üzi beépített, helyet, assertEquals() 3. paramétereként**

Lehetőség ne a jUnit beépített, hanem saját üzikre, ha az assert elbukik

AssertEquals() 3. paramétereként tudjuk megadni



Lehetőség arra is, hogy a hibaüzit kizárólag akkor értékelje ki, ha a teszt elbukik; pl. egy hatalmas xmlt akarnánk kiírni a konzolra akkor, ha a teszt elbukik, ami nagyon erőforrás igényes; 3. paraméterként egy lambdakifejezéssel



**Assume**

Ha ezek elbuknak, a teszteset végrehajtása leáll, de nem fog a jUnit hibát jelenteni; bizonyos eseteket csak akkor érdemes lefuttatni, ha valamilyen feltételek fennállnak

Asserthez hasonló metódusokkal rendelkezik

import static org.junit.jupiter.api.Assumptions.\*;

**Az Assert metódus használata a gyakorlatban (gyakorlat)**

1. EmployeeTest, init John Doe, testGetAge(), AssertNotnull()
2. equals és hash implement, másik expected employee és assertSame
3. fail() egyik esetbe
4. assertEquals() delta
5. assertAll lamdával, getAge és getName; mindne hibát kiír, míg külön asert soroknál mi van?
6. EmployeeService osztály listEmployees() és listEmployeeNames() metódussal -> return list.of.. , EmployeeServiceTest(); névre csekk streammel és üzivel
7. assumeTrue() - > employeeService! null

**Kérdések**

*Mire használjuk az Assert statikus metódusait?*

*Hogy vizsgálunk nullra?*

*Mit hív meg az assertEquals() – assertNotEquals() és milyen típusokra használható?*

*Mit vizsgál az assertSame() és assertNotSame?*

*Miért nem célszerű az assertTrue() és assertFalse() használata?*

*Hogyan buktatjuk el explicit módon a tesztesetet és mikor használjuk?*

*Milyen paramétert használunk assertEquals()hoz lebegő pontos számok esetén?*

*Milyen metódusokat ismersz kollekciók és tömbök összehasonlítására?*

*Hogy ágyazunk egybe több assertet az assertAll()-al és ez mit fog eredményezni?*

*Miért korlátozott jUniton belül a kollekciókon végzett assert? Mit használunk ennek kiküszöbölésére?*

*Milyen 3rd party librarykat ismersz, amik kiterjesztik az Assertet?*

*Hogyan építünk be alternatív üzenetet jUnitban assertre?*

*Mire jó az Assume és milyne metódusai vannak?*