**Teszt lefedettség**

**(mérőszám ami megmondja, utasításszinten is lehet, 70-80%, white boksz tesztelés, manuális tesztelés, fejlesztőeszközi támogatás)**

egy **mérőszám**, ami megmondja, hogy az adott alkalmazás tesztelésekor az **alkalmazás kódsorainak hány százaléka futott le** = hány százaléka lett letesztelve

meg lehet adni **utasításszinten** is (a végrehajtott utasítások között mennyi futott le, amikor a teszteket futtattuk), és lehetséges ágak szerint is

**70-80%os lefedettség már jónak** mondható; automatikus tesztekkel le van fedve

e felett már getter setterek, generált metódus tesztelés is kéne ami nem éri meg

**white box** testing módszer =fejlesztői; **megnézzük annak a belsejét, amit tesztelünk(=forráskód**); milyen utik elágazások stb vannak benne? ebből kiindulva kezdjük el írni a teszteket és folyamatosan növeljük a lefedettséget azzal, hogy rárakunk teszteket olyan sorokra is, amikre addig nem volt

akár **manuális tesztelésnél is használható**; beállítjuk a tesztlefedettség eszközt, hogy mérjen, majd akár kattintgatjuk manuálisan a felületet, és nézzük mire futott le teszt és mire nem, és tudjuk segíteni a tesztelőket, hol mi maradt ki

**fejlesztőeszközök támogatása -** >rögzítik mely sorok futnak le a tesztesetek futása során

**Jacoco**

**Maven függőség, bíte kód manipuláció és jacoco.exe állomány, olvasható megjelenítéshez mvn jacoco:report)**

eszköz, egyik **leggyakoribb**

Maven project esetén **plugin felvétele pom.xmlbe**

**byte kód manipulációt végez** -> az adott **classokat módosítja** úgy, hogy önmaguk **rögzítsék le** egy fileba, hogy **milyen utik futottak le** -> a futás során létrejön egy **jacoco.exe** bináris állomány, ami tárolja ezt; ezt **olvasható formában való megjelenítéshez** (pl. html formátumban) a **jacoco maven pluginjának a riport goalját kell meghívni** (mvn jacoco:report)



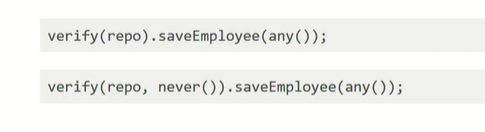
**A teszt lefedettség mérése (gyakorlat)**

legegyszerűbb a fejlesztőeszközből végezni; IDEA - > jobb klikk testnél a packagere és Run with coverage

eredmények és összefoglaló ablak; ha belemegyünk az osztályba, látszik, mely sorok vannak lefedve(zöld)

parancssoros mérés, vagy bekötés conituous integration rendszerbe, vagy ha hisztorikusan akarjuk látni, akkor érdemes pl. sonar cube-ot használni (alkalmazás kódminőség mérésére, de ez egy meglévő mérő eszköz eredményét használja fel , ez a jacoco!)

jacoco -> pom.xml felvétel pluginként; futtatunk egy packaget -> első lépésként lefut a jacoco prepare agent goalja, ami megváltoztatja az osztályokat a fentiek szerint; teszt esetek futása után létrejön a targetben a jacoco.exe bináris állomány, ha ezt elemezni szeretnénkjacoco: report goal futtatása (lehet ideából is, hogy?) -> site könyvtár a targetben és megjelenik egy html file (index.html); beleklikkelgetett a layersekbe még a htmlen belül

****