**Mockito**

**Külső függőségek**

**(továbbhívás más osztályba -> integrációs; függőség nélkül tesztelés -> csereosztály=test double; fajták(dummy, fake, stubs, mock; mock -> milyen válaszokat adjon/rögzít, megfelelő meti a megfelelő paraméterrel)**

**a tesztelt osztály gyakran rendelkezik külső függőségekkel** = továbbhív más osztályokat; ilyenkor ez már integrációs teszt

**SUT = system unnder test**

hogyan lehet az osztályt mégis magában tesztelni a függőségei nélkül? **-> lecseréljük a függőségét egy tesztelésre előre előkészített példánnyal** = csereosztályok, v. **test double**

*test double fajtái:*

dummy – **semmit nem csinál,** metódusai üresek, csak azért van, hogy leforduljon az adott teszteset esetében az adott osztály

fake – már működő implementáció, de nem éles; **hasonlóan működik, mint az eredeti, de sok korláttal**; pl. ez az osztály a memóriába ment, míg az eredeti adatbázisba

stubs – **előkészített válaszokkal rendelkezik,** megmondjuk neki, hogyha meghívják valamely metódusát, mit adjon vissza; általánosságban nem használható csak az adott teszt esetre

**mock** – amellett, hogy **megmondhatjuk** neki, **milyen válaszokat adjon**, **rögzíti** is a rajta elvégzett **hívásokat**; a tesztelendő osztály hívásait; erre tudunk teszteket írni, hogy a mock **megfelelő metódusai lettek- e meghívva, a megfelelő paraméterekkel**

**Unit tesztek csoportosítása**

**(állapot- és viselkedés alapú -> külső kapcsolatot megfelelően veszi-e igénybe)**

Állapot alapú: a **tesztelendő osztály egy metódusát meghívja majd megnézi, hogy amit visszaadott eredmény helyes-e** (=beadott bementre megfelelő kimenetet adja-e vissza)

Viselkedés alapú: azt vizsgálja, hogy **egy külső kapcsolattal rendelkező osztály a kapcsolattát megfelelőképpen veszi-e igénybe**; adott inputra a kapcsolódó osztályban a megfelelő metódusokat hívja-e meg a megfelelő paraméterekkel

**MOCKITO**

**(mock objektum, definiálás, ellenőrzés, test double), default visszatérési értékek, üzleti logika tesztelése és perzisztens réteg hívás / service->repository)**

**Keretrendszer, amely mock objektumok programozott előállítására szolgál**

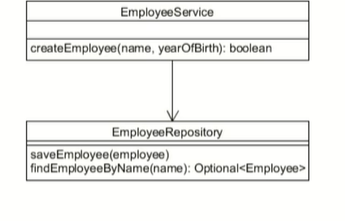
úgy néznek ki, mint az eredti példány, de **definiálható**, hogy **milyen metódushívásra mit adjon vissza** és **ellenőrizhető, hogy a megfelelő metódus meg lett-e hívva és hogy milyen paraméterrel** = **test** **double**

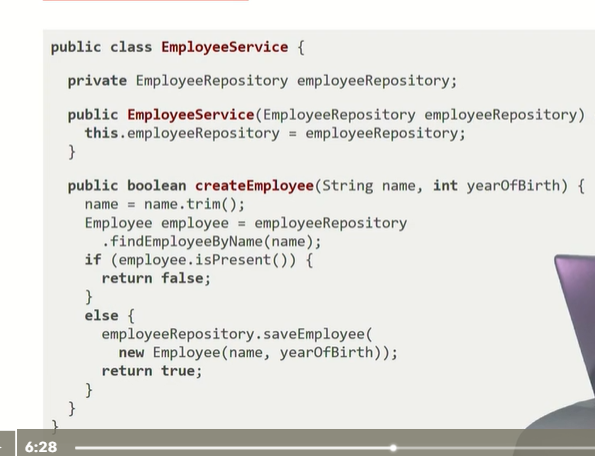
ha **alapból** generálunk egy mock objektumot, akkor ez olyan metódusokat tartalmaz, ami objektum esetén **nullt**, **boolean** esetén **falset**, primitív típusok esetén pedig **0** értéket ad vissza

**leggyakoribb helyzet** a mock objektumok használatára az üzleti logika tesztelése, és **az üzleti logika hívja a perzisztens réteget** (service és repository Spring esetén; JEE- ben – DAO)

*Eredeti osztályok*

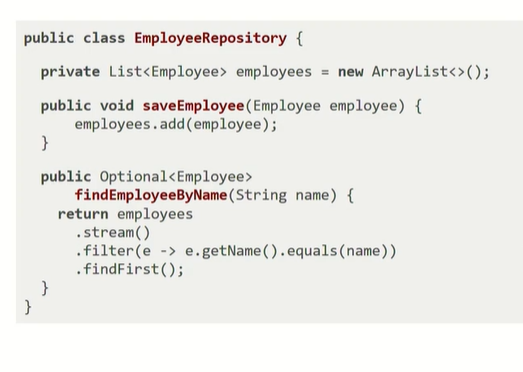
EmployeeService -> EmployeeRepository atrival(létrehozás konstruktor injectionnel); createEmployee metódus a servicben, ami hívja a repositoryban a findot és a savet



****

konstruktor injection – Employeerepository atri; a háttérben ide a mockito az eredeti repository egy leszármazottját fogja adni, amely metódusai első körben nem csinálnak semmit

a réteg két szabályt implementál (trim és if-else)



**Service Test mock objektummal**

**(SUT, mockolt EmployeeRepository teszt metóduson belül; default visszatérés, createEmployee=true)**

****

SUT -> EmployeeService osztály, aminek van egy külső referenciája, az EmployeeRepsitory, amit konstruktorban kap

a **Mockitonak** megmondjuk, hogy **hozzon létre az EmployeeRepostiroy alapján egy mock objektumot**; ennek először üres metódusai lesznek

utána ha meghívjuk a service createEmployee() metódusát, az tovább fog hívni az EmployeeRepositoryra, megnézi, hogy van-e olyan employee, de mivel az implementációja úgy fog kinézni, hogy return false (default implementációja a mocknak boolean típusra). ezért meg fogja hívni a save-t

arra ellenőrzünk rá, hogy a createEmployee metódus truet ad-e vissza és el tudja-e menteni az alkalmazottat

**Dependency Injection**

**további egyszerűsítés -> mockolt repository teszt metódusokon kívül; MockitoExtension átadása a teszteseteknek @ExtendWith annotációval; @Mock - > mockolandó osztály és @InjectMocks -> tesztelendő osztály -> példányosít és konstruktor injectionnel mockolt objektumot átadja)**

@Extendwith – valami paramétert ezzel automatikusan átadunk a tesztesetnek!

ugyanez a tesz eset, csak most @**ExtendWith** annotációval átadunk egy **MockitoExtensiont**; amely osztályt mockolni szeretnénk azt felvesszük attribútumként Mock annotációval

a tesztelendő osztály(EmployeeServicve) pedig az **@InjectMock** annotációval látjuk el;

így a **java példányosítani fog** egy EmployeeServicet **és paraméterként a létrehozott Mock** objektumot fogja neki átadni

* fel lesz töltve az EmployeeRepository értéke egy mockolt objektummal és egy EmployeeService úgy kerül példányosíytásra, hogy az előbbi lesz neki átadva

a tesztelendő metódustól azt várjuk, hogy truet adjon vissza, mert ő továbbhívott a mockolt employeeRepository findEmployeeByNAme metódusába ami defaultként falset fog visszaadni



**Visszatérési érték definiálása**

**(when – anyString – thenReturn – thenThrow ->exception)**

Hogyan lehet a default mock működést kicserélni? pl. a findEmployee ne false-al térjen vissza

**when()** metódusnak át kell adni a repo.findEmployeeName-t, annak paraméterül pedig **anyStringet** kell megadni, utána megadjuk a **thenReturnben**, hogyha meghívták a metódust bármilyen Stringgel, akkor azt az employeet adja vissza

a visszatérési érték false lesz és a teszteset elbukik? nem, ha **assertFalset** használunk

a mockolt objektumnak azt is meg lehet adni, hogyha **megívják bármilyen stringgel, dobjon exceptiont ->thenThrow()**



**Metódushívás ellenőrzése**

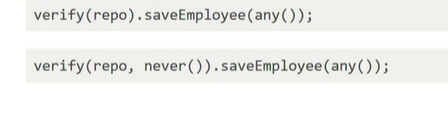
**(verify, any, never)**

**egy mock objektumon az adott metódus meghívásra került-e?**

**verify()** metódus

1. az adott metódus bármilyen paraméterrel meghívásra került-e
2. szabály -> biztos, hogy az adott metódus hívás közben nem szabad meghívni a saveEmployee() metódust -> paraméterként a verify()nak még adunk egy **never()t** ->

a teszt futása közben a repositorynak nem lehet meghívni a saveEmployee() metódusát; ha mégis, a tesztesetet elbukik

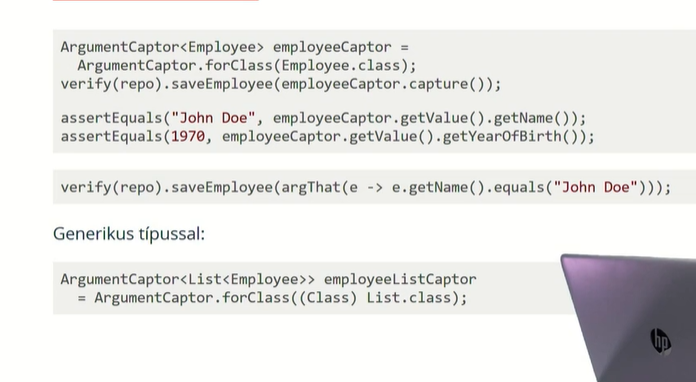
****

**Paraméter ellenőrzése**

**(verify -> argThat -> lamda)**

Azt is lehet ellenőirzni, hogy a **metódus milyen paraméterrel került meghívásra**

ehhez ArgumentCaptor-t kell használni; bonyolult megadási mód; Java8tól egyszerűbb -> a verify esetén az argumentumot az **argThat()** metódussal definiáljuk **és paraméterként egy lambda kifejezést adunk át**, amiben megadjunk, hogy milyen paraméterrel kell meghívni az adott metódust; a verify kifejezés azt írja le, hogy amikor meghivtuk a createEmployeet a Servicben, akkor neki tovább kell hívnia a repositoryban a SaveEmployeet, **olyan paraméterrel, amely paraméternek a neve meg kell, hogy egyezzen** John Doe-val -> azt vizsgáljuk, hogy megfelelőképpen hív-e tovább az adott service, a SUT



**POM.XML**

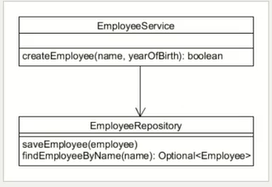
Mockito használatához test scope-al fel kell venni:

****

**A Mockito használata (gyakorlat)**

0.empapp ->layers

EmployeeService és EmployeeRepository atri (mentse csak listába, mint az emprep atrija); a save csak akkor hozza létre, ha a findemployee falset ad vissza; streammel működjön, name trim!

****

EmployeeServiceTest és testSaveemployee()

EmpService példányosítás, az emprepository atri példányt mint mock objektumot -> pom.xml update! a mock híváskor a savere semmit(void), a fidnra Optional emptyt fog visszaadni

!import static org.mockito.Mockito.*mock*;!

1.assertTrue -> create

2.verify -> saveemployee meghívásra került-e? (any mint paraméter)

3.else ág a savel menjen kommentbe -> erre már falset adjon a verify

4.mock objektum manuális példányosítása helyett &ExtendWith(..) és @Mock annotáció atrira és @Injectmocks Empservicere (ezzel konstruktorban a háttérben átadja neki a mockolt objektumot); majd újrafutttás

5.legyen már ilyen alkalmatott, magyarán a find truet, a save falset adjon vissza -> testSaveEmplAlreadyExists()

-> when (..then.return(optional.of..) majd .boolean result = create.., assertfalse!

verify -> never -> true

6.megfelelő paraméterrel lett-e meghívva? Java8 -> argthat(lamda – true; majd vegyük ki a trimet, hogy bukjon el és rakjunk spaceeket a createnél a név elé