

Email Address: t.levente1598@gmail.com
Név: Tóth Levente
Date Started: 06/28/2021 2:57:02 PM
Date Completed: 06/28/2021 3:14:40 PM
Attempt: 1

Question Topic		
Num	Question	
	Respondent's Answer	Correct Answer
JUnit		
Group Earned Points: 11 of 11 (100%)		

1 Melyik tanácsot nem kell betartani az alábbiak közül?

- | | |
|---|---|
| <p><input type="radio"/> Mindig használjuk a megfelelő assert metódust!</p> <p><input checked="" type="radio"/> Mindig ellenőrizzünk le mindent, lehetőleg egy kollekció minden elemére írunk assertet!</p> <p><input type="radio"/> A tesztesetek legyenek függetlenek!</p> <p><input type="radio"/> Törekedjünk a minél nagyobb tesztlefedettségre!</p> | <p><input type="radio"/> Mindig használjuk a megfelelő assert metódust!</p> <p><input checked="" type="radio"/> Mindig ellenőrizzünk le mindent, lehetőleg egy kollekció minden elemére írunk assertet!</p> <p><input type="radio"/> A tesztesetek legyenek függetlenek!</p> <p><input type="radio"/> Törekedjünk a minél nagyobb tesztlefedettségre!</p> |
|---|---|

✓ Earned Points: 1 of 1

2 Mi a különbség a unit teszt és az integrációs teszt között?

- | | |
|--|--|
| <p><input type="radio"/> Az integrációs teszt csak az alkalmazásba beintegrált third-party-libraryk ellenőrzésére szolgál, a unit teszt pedig a saját alkalmazásunk tesztelésére.</p> <p><input type="radio"/> Integrációs tesztnél elengedhetetlen a mockolás, míg unit tesztnél nem szükséges.</p> <p><input type="radio"/> Az integrációs tesztek sokkal gyorsabbak mint a unit tesztek.</p> <p><input checked="" type="radio"/> A unit teszt a program kis egységeit teszteli, az integrációs teszt pedig különböző osztályok együttműködését.</p> | <p><input type="radio"/> Az integrációs teszt csak az alkalmazásba beintegrált third-party-libraryk ellenőrzésére szolgál, a unit teszt pedig a saját alkalmazásunk tesztelésére.</p> <p><input type="radio"/> Integrációs tesztnél elengedhetetlen a mockolás, míg unit tesztnél nem szükséges.</p> <p><input type="radio"/> Az integrációs tesztek sokkal gyorsabbak mint a unit tesztek.</p> <p><input checked="" type="radio"/> A unit teszt a program kis egységeit teszteli, az integrációs teszt pedig különböző osztályok együttműködését.</p> |
|--|--|

✓ Earned Points: 1 of 1

3

Melyik állítás igaz az alábbiak közül?

- ☐ () A `@RepeatedTest` annotációnak egy paramétere van.
- ☐ () A `@RepeatedTest` annotációval ellátott metódusokban nem tudjuk nyomkövetni hogy hányadik ismétlésnél tartunk.
- ☒ (X) A `@RepeatedTest` annotációval ellátott metódusoknál a teszt adatokat egy tömbök tömbjében adhatjuk meg.
- ☐ () A `@RepeatedTest` annotáció paraméter nélkül, önmagában is használható.

- ☐ () A `@RepeatedTest` annotációnak egy paramétere van.
- ☐ () A `@RepeatedTest` annotációval ellátott metódusokban nem tudjuk nyomkövetni hogy hányadik ismétlésnél tartunk.
- ☒ (X) A `@RepeatedTest` annotációval ellátott metódusoknál a teszt adatokat egy tömbök tömbjében adhatjuk meg.
- ☐ () A `@RepeatedTest` annotáció paraméter nélkül, önmagában is használható.



Earned Points: 1 of 1

4

Melyik állítás hamis a JUnittal kapcsolatban?

- ☐ () A JUnit tesztesetek felépítésének egyik módja a Given-When-Then struktúra.
- ☒ (X) A JUnit bármelyik verziója csak Maven projektekben használható.
- ☐ () A jelenleg legfrissebb verzió a JUnit 5.
- ☐ () A tesztesetek egyszerű metódusok melyeket `@Test` annotációval kell ellátnunk.

- ☐ () A JUnit tesztesetek felépítésének egyik módja a Given-When-Then struktúra.
- ☒ (X) A JUnit bármelyik verziója csak Maven projektekben használható.
- ☐ () A jelenleg legfrissebb verzió a JUnit 5.
- ☐ () A tesztesetek egyszerű metódusok melyeket `@Test` annotációval kell ellátnunk.



Earned Points: 1 of 1

5

Adott az alábbi tesztesztály:

```
class LifecycleTest {
```

```
    @BeforeAll
```

```
    void init() {
```

```
        System.out.println("Init");
```

```
    }
```

```
    @BeforeEach
```

```
    void initEach() {
```

```
        System.out.println("Hello");
```

```
    }
```

```
    @Test
```

```
    void test() {
```

```
        System.out.println("My");
```

```
    }
```

```
    @Test
```

```
    void test2() {
```

```
        System.out.println("World");
```

```
    }
```

```
}
```

Milyen sorrendben kerülnek a szavak a konzolra (egymástól sortöréssel elválasztva)?

☐ () Init Hello My World

☐ () Init Hello My Hello World

☐ () Init Hello My Init Hello World

☒ (X) Nem támaszkodhatunk a tesztesetek lefutási sorrendjére.

☐ () Init Hello My World

☐ () Init Hello My Hello World

☐ () Init Hello My Init Hello World

☒ (X) Nem támaszkodhatunk a tesztesetek lefutási sorrendjére.


Earned Points: 1 of 1

6

Mi az a unit teszt?

☐ () A tesztelők írják. Olyan teszt, amivel az alkalmazásunk funkcióit tudják tesztelni.

☐ () A unit teszt egy Javás fogalom, a Java állományainkat tudjuk vele tesztelni.

☒ (X) Az adott programozási nyelv legkisebb egységeit tudjuk vele tesztelni.

☐ () Az alkalmazásunk együttműködését teszteli más third-party library-kkel.

☐ () A tesztelők írják. Olyan teszt, amivel az alkalmazásunk funkcióit tudják tesztelni.

☐ () A unit teszt egy Javás fogalom, a Java állományainkat tudjuk vele tesztelni.

☒ (X) Az adott programozási nyelv legkisebb egységeit tudjuk vele tesztelni.

☐ () Az alkalmazásunk együttműködését teszteli más third-party library-kkel.


Earned Points: 1 of 1

7

Adott a következő tesztosztály. Feltételezhetjük, hogy a Movie osztály kész. Két attribútuma String és int típusú, ezekhez generáltunk konstruktort, illetve getter és setter metódusokat.

```
class MovieTest {

    Movie movie = new Movie("Titanic", 120);

    @Test
    void setTitleTest(){
        movie.setTitle("Star Wars");

        assertEquals("Star Wars", movie.getTitle());
    }

    @Test
    void getTitleTest(){
        assertEquals("Titanic", movie.getTile());
    }
}
```

Melyik igaz a tesztosztályra?

- ☐ () Nem fordul le a kód.
- ☐ () A setTitleTest() metódus elbukik.
- ☐ () A getTitleTest() metódus elbukik.
- ☒ (X) Minden teszt sikeres.

- ☐ () Nem fordul le a kód.
- ☐ () A setTitleTest() metódus elbukik.
- ☐ () A getTitleTest() metódus elbukik.
- ☒ (X) Minden teszt sikeres.



Earned Points: 1 of 1

8

Az alábbiak közül melyik metódushívás felel meg a JUnit követelményeknek? (Az actual változó az aktuális értéket, míg az expected változó az elvárt értéket tárolja)

- ☐ () assertEquals(expected)
- ☐ () assertEquals(actual, expected)
- ☒ (X) assertEquals(expected, actual)
- ☐ () assertEquals(actual)

- ☐ () assertEquals(expected)
- ☐ () assertEquals(actual, expected)
- ☒ (X) assertEquals(expected, actual)
- ☐ () assertEquals(actual)



Earned Points: 1 of 1

9

Melyik állítás hamis az alábbiak közül?

- ☒ (X) A unit tesztelés egyedüli hátránya, hogy nem refactoring tűrő.
- ☐ () A unit tesztek a fejlesztő írja.
- ☐ () A unit tesztelés elősegíti a fejlesztést, mert előbb fény derül a hibákra.
- ☐ () Egy módszer a Test Driven Development, mikor a tesztek írók meg előbb.

- ☒ (X) A unit tesztelés egyedüli hátránya, hogy nem refactoring tűrő.
- ☐ () A unit tesztek a fejlesztő írja.
- ☐ () A unit tesztelés elősegíti a fejlesztést, mert előbb fény derül a hibákra.
- ☐ () Egy módszer a Test Driven Development, mikor a tesztek írók meg előbb.




Earned Points: 1 of 1

10 Melyik állítás igaz az alábbiak közül?

- ☐ Kivételt nem tesztlünk.
- ☒ Kivétel dobásának tényét az `assertThrows` metódussal tesztlünk.
- ☐ Kivétel dobásának tényét az `assertEquals` metódussal tesztlünk, ahol az `expected` paraméter a kivétel osztály.
- ☐ Kivételek üzeneteit nem tudjuk tesztelni

- ☐ Kivételt nem tesztlünk.
- ☒ Kivétel dobásának tényét az `assertThrows` metódussal tesztlünk.
- ☐ Kivétel dobásának tényét az `assertEquals` metódussal tesztlünk, ahol az `expected` paraméter a kivétel osztály.
- ☐ Kivételek üzeneteit nem tudjuk tesztelni

 Earned Points: 1 of 1

11 Melyik állítás hamis az alábbiak közül?

- ☐ Az `assertThrows()` metódus második paramétere egy lambda kifejezés
- ☒ Az `assertThrows()` metódusnak nincs visszatérési érték
- ☐ A teszteset, amiben van `assertThrows()` metódus, elbukik, ha a tesztelt metódus nem dobott kivételt
- ☐ Az `assertThrows()` metódussal akár saját kivétel dobását is tesztelhetjük

- ☐ Az `assertThrows()` metódus második paramétere egy lambda kifejezés
- ☒ Az `assertThrows()` metódusnak nincs visszatérési érték
- ☐ A teszteset, amiben van `assertThrows()` metódus, elbukik, ha a tesztelt metódus nem dobott kivételt
- ☐ Az `assertThrows()` metódussal akár saját kivétel dobását is tesztelhetjük

 Earned Points: 1 of 1


AssertJ

Group Earned Points: 4 of 5 (80%)

12 Az alábbiak közül melyik helyes AssertJ metódus hívás ?

- ☐ `assertThat("John Doe", name).startsWith("John");`
- ☐ `assertEquals(name).startsWith("John").endsWith("Doe");`
- ☐ `assertThat(name, "John Doe").endsWith("Doe");`
- ☒ `assertThat(name).startsWith("John");`

- ☐ `assertThat("John Doe", name).startsWith("John");`
- ☐ `assertEquals(name).startsWith("John").endsWith("Doe");`
- ☐ `assertThat(name, "John Doe").endsWith("Doe");`
- ☒ `assertThat(name).startsWith("John");`

 Earned Points: 1 of 1

13

Mi az a tuple?

- ☐ () Egy Java adatszerkezet a Collection interfészből származik és a Tuple interface reprezentálja.
- ☒ (X) Ha több paraméter alapján végzek extractingot akkor az AssertJ úgynevezett tuple-be gyűjti az eredményt.
- ☐ () A tuple egy tesztelési fogalom, ha két teszt ugyanazt ellenőrzi, akkor ők egy tuple.
- ☐ () A tuple() egy metódus, amivel egy változó típusát tudjuk ellenőrizni.

- ☐ () Egy Java adatszerkezet a Collection interfészből származik és a Tuple interface reprezentálja.
- ☒ (X) Ha több paraméter alapján végzek extractingot akkor az AssertJ úgynevezett tuple-be gyűjti az eredményt.
- ☐ () A tuple egy tesztelési fogalom, ha két teszt ugyanazt ellenőrzi, akkor ők egy tuple.
- ☐ () A tuple() egy metódus, amivel egy változó típusát tudjuk ellenőrizni.



Earned Points: 1 of 1

14

Mi az a soft assert?

- ☐ () Olyan tesztet, amit a Maven nem futtat le csak mi tudjuk kézzel futtatni.
- ☐ () Olyan tesztet, ami egy előellenőrzést végez a valódi assert előtt.
- ☒ (X) Olyan tesztet, ami hibás eredmény esetén nem bukik el egyből, csak a teszt lefutása után kiírja az eredményt.
- ☐ () Olyan tesztet, ami csak a lényegi eltéréseket nézi, például egy kis-nagybetű nem egyezés miatt nem bukik el

- ☐ () Olyan tesztet, amit a Maven nem futtat le csak mi tudjuk kézzel futtatni.
- ☐ () Olyan tesztet, ami egy előellenőrzést végez a valódi assert előtt.
- ☒ (X) Olyan tesztet, ami hibás eredmény esetén nem bukik el egyből, csak a teszt lefutása után kiírja az eredményt.
- ☐ () Olyan tesztet, ami csak a lényegi eltéréseket nézi, például egy kis-nagybetű nem egyezés miatt nem bukik el



Earned Points: 1 of 1

15

Mi az az AssertJ?

- ☐ () Egy third-party-library mellyel könnyebben tudunk asserteket írni
- ☒ (X) A JUnit 5-ös verziójának egy kiegészítő funkciója
- ☐ () A JUnit egy konkurenciája
- ☐ () A Hamcrestet nevezték át AssertJ-re a JUnit 5 megjelenésekor

- ☒ (X) Egy third-party-library mellyel könnyebben tudunk asserteket írni
- ☐ () A JUnit 5-ös verziójának egy kiegészítő funkciója
- ☐ () A JUnit egy konkurenciája
- ☐ () A Hamcrestet nevezték át AssertJ-re a JUnit 5 megjelenésekor



Earned Points: 0 of 1

16

Az alábbiak közül melyik helyes, ha Employee objektumok listáját akarom tesztelni?

(X) `assertThat(employees).hasSize(3).extracting(Employee::getName).contains("John Doe");`
 () `assertThat(employees).hasSize(3).extracting(name).contains("John Doe");`
 () `assertThat(employees).hasSize(3).extracting(getName).contains("John Doe");`
 () `assertThat(employees).hasSize(3).extracting(e.getName()).contains("John Doe");`

(X) `assertThat(employees).hasSize(3).extracting(Employee::getName).contains("John Doe");`
 () `assertThat(employees).hasSize(3).extracting(name).contains("John Doe");`
 () `assertThat(employees).hasSize(3).extracting(getName).contains("John Doe");`
 () `assertThat(employees).hasSize(3).extracting(e.getName()).contains("John Doe");`



Earned Points: 1 of 1

Mockito

Group Earned Points: 2 of 3 (67%)

17

Mire való a `verify()` metódus?

(X) Mockolásnál azt tudjuk vele ellenőrizni, hogy valóban meghívásra került-e egy metódus.
 () Mivel a unit teszt dokumentációként is funkcionál, ezt a metódust kell utoljára futtatnunk.
 () Az assertek után ezzel a metódussal tudjuk hivatalossá tenni a tesztesetet.
 () Az assertek előtti ellenőrzésre való.

(X) Mockolásnál azt tudjuk vele ellenőrizni, hogy valóban meghívásra került-e egy metódus.
 () Mivel a unit teszt dokumentációként is funkcionál, ezt a metódust kell utoljára futtatnunk.
 () Az assertek után ezzel a metódussal tudjuk hivatalossá tenni a tesztesetet.
 () Az assertek előtti ellenőrzésre való.



Earned Points: 1 of 1

18

Adott egy `MovieController` osztály mely függ egy `MovieService` nevű osztálytól, azaz van egy `MovieService` típusú attribútuma, melyet konstruktoron keresztül tudunk beállítani. A `MovieController` `findByTitle()` metódusa továbbhív a `service.findByTitle()` metódusába. Ekkor adott a következő teszt:

@Test

```
void testFindByTitle() {
```

```
    MovieService service = mock(MovieService.class);
```

```
    when(service.findByTitle(any())).thenReturn(new Movie("Titanic", 120, 6.5));
```

```
    MovieController controller = new MovieController(service);
```

```
    assertThat(controller.findByTitle("Titanic").getTitle()).isEqualTo("Titanic");
```

```
}
```

Mi történik?

() A teszt lefordul és a teszt sikeres.
 () A teszt lefordul, de a teszt elbukik.
 () A teszt lefordul és futási idejű hibát kapunk.
 (X) A teszt nem fordul le.

(X) A teszt lefordul és a teszt sikeres.
 () A teszt lefordul, de a teszt elbukik.
 () A teszt lefordul és futási idejű hibát kapunk.
 () A teszt nem fordul le.



Earned Points: 0 of 1

19

Mi az a test-double?

- ☐ () A három paraméteres assertEquals() hívjuk test-double-nek, amikor harmadik paraméternek megadhatunk egy hibahatárt lebegőpontos számok ellenőrzésekor.
- ☐ () Két egymástól függő teszteset.
- ☐ () Az AssertJ-ben a kétszer lefutó paraméterezett teszt.
- ☒ (X) Amikor két osztály függ egymástól, a függőség egy tesztelésre előkészített példánya.

- ☐ () A három paraméteres assertEquals() hívjuk test-double-nek, amikor harmadik paraméternek megadhatunk egy hibahatárt lebegőpontos számok ellenőrzésekor.
- ☐ () Két egymástól függő teszteset.
- ☐ () Az AssertJ-ben a kétszer lefutó paraméterezett teszt.
- ☒ (X) Amikor két osztály függ egymástól, a függőség egy tesztelésre előkészített példánya.



Earned Points: 1 of 1

Respondent Summary

Overall Time Used: 00:17:38
Score Percentile: 48th
Overall Result: Pass

Final Score: 89%