Has completado este test el *07/06/2022, 14:14*
Tu calificación es 75.00%

CORRECTO

Las realización de pruebas unitarias junto con una refactorización constante a medida que crecen los proyectos de software garantizan el crecimiento sostenible de los proyectos.

- ✓ ***Verdadero***
- Falso

CORRECTO

Un caso de prueba que se ejecuta en una prueba unitaria es código que nada tiene que ver con el uso real que tendrá el software en los entornos de producción a los cuales sera sometido posteriormente

- Verdadero, únicamente sirven para operaciones sencillas
- ✓ ***Falso, el código de pruebas permiten garantizar la corrección de la aplicación ante posibles escenarios en los entornos de producción***
- Verdadero, el código de pruebas no se compila en la aplicación final
- Falso, el código de prueba define nuevas funcionalidades y métodos de la solución

---

INCORRECTO

Relaciona los conceptos con su correspondiente descripción

✓***Objetos que permiten controlar y "espiar" el uso que se hace del propio objeto***	✓***Spy***
✗*Objeto equivalente al objeto real que simula correcta y completamente los comportamientos, se usa para simplificar las pruebas*	✗*Test Double* **(Fake)**
✗*Objeto que reemplaza a uno equivalente de la aplicación real y cuyo comportamiento podemos configurar a nuestra conveniencia*	✗*Mock* **(Test Double)**

✓ ***Objetos que devuelven valores establecidos necesarios para las pruebas***	✓ ***Stub***
✗ *Objetos que devuelven valores establecidos necesarios para las pruebas y ademas tenemos control del comportamiento que realizan*	✗ *Fake* **(Mock)**
✓ ***Objeto que necesitamos para ejecutar un test aunque no hará nada, o bien no afectara a la prueba***	✓ ***Dummy***

CORRECTO

Si tenemos un caso de prueba donde el componente, método o código que estamos probando cuenta con una dependencia a una base de datos MySql desde donde se trae informacion de una consulta de las transacciones vendidas en el ultimo mes y deseamos que nuestro caso de prueba no dependa de dicha conexión a la base de datos a fin de simplificar la prueba unitaria. Podemos hacer uso de....

Stub, un objeto que devolverá valores predeterminados

✓ ***Fake, un objeto que falsea una conexión a una base de datos en memoria para que el caso de prueba continúe***

Spy, un objeto que supervisara los argumentos que se pasan a la prueba unitaria

CORRECTO

Los Dobles de Prueba (Test Doubles) solo pueden ser creados mediante frameworks como JUnit y Mockito

Falso, no es posible crear Test Doubles con Mockito

Verdadero, Mockito facilita la creación de Test Doubles

✓ ***Falso, podemos crear clases que emulen los comportamientos de los diferentes Test Doubles sin necesidad del framework***

Verdadero, solo se pueden crear con la etiqueta @Mock de Mockito

CORRECTO

Usando Mockito, si queremos que una clase determinada se comporte como un Mock en nuestro caso de prueba debemos usar la etiqueta:

✓ ***@Mock***

@Before

@Rule

@Test

CORRECTO

El método estático When() permite...

Ejemplo:

```
when(dataService.getListOfNumbers()).thenReturn(new int[] { 1, 2, 3, 4, 5 })
```

- ✓ ***Definir un valor de retorno para un comportamiento especifico***

  Cambiar el comportamiento de un método determinado

  Definir el tipo de datos de retorno para un comportamiento especifico

  Cambiar el comportamiento del estado de un objeto determinado

---

INCORRECTO

Del siguiente fragmento de código

```
1. @Test
2. public void testAddTwo(){
3.     when(dependency.addTwo(1)).thenReturn(5);
4.     assertEquals(5, dependency.addTwo(1));
5.     assertEquals(null, dependency.addTwo(27));
6. }
```

El caso de prueba resultara en:

  AssertionError

- ✗ *NullPointerException*

  Test Passed