Repaso Examen - Testing

Cultivo

Imagina que trabajas en la programación de un gestor de cultivos y se ha implementado un programa para llevar a cabo la contabilidad de la producción de cada uno de los cultivos de los que se disponen. Nos facilitan una versión inicial en la que es necesario implementar los test para comprobar que lo desarrollado hasta el momento funciona correctamente.

Requisitos del sistema

El sistema debe ser capaz de:

- El programa debe de ser capaz de crear objetos con sus atributos.
- Permitir a los usuarios que interactúen los objetos mediante una gestión de cultivos y mostrar quien los trabaja, qué maquinarias intervienen y calcular los ingresos de los cultivos y productos desprendidos de él.

Clases

1. Las clases **Cultivo, Empleado**, **Maquinaria** y **Producto** representan los objetos con los que se puede trabajar para administrar el sistema. Cada clase implementa los constructores, getters y funciones propias de la clase. Se observa que los atributos de la clase tienen una especificación del valor de los datos.

Tests Unitarios

Para comprobar que el sistema funciona correctamente, debes realizar los siguientes tests:

- Tests unitarios: Debes crear tests unitarios para cada una de las clases y métodos, asegurándote de que funcionan correctamente. Es decir, debéis crear tests para comprobar los setters y getters y cada uno de los métodos como se especifica a continuación.

CultivoTest

- *testCalcularRendimientoTotal*
- *testCalcularIngresos*
- *testNombreCultivoNoPuedeSerVacio* **usando el constructor**
- *testAreaDebeSerPositivo* **usando el constructor**

EmpleadoTest

- *testCalculaSalarioAnual*
- *testDarAumento*
- *testNombreEmpleadoNoVacio* **usando el constructor**
- *testSalarioDebeSerPositivo* **usando el constructor**

MaquinariaTest

- *testCalcularTiempoArado*
- *testAnyadirHorasTrabajo*
- *testTipoMaquinariaNoVacio* **usando el constructor**
- *testHorasUsoNoNegativo* **usando el constructor**

ProductoTest

- *testCalcularPrecioConDescuento*
- *testNombreProductoNoVacio* **usando el constructor**
- *testPrecioDebeSerPositivo* **usando el constructor**

Tests Funcionales

ExceptionTest

- *testCultivoConstructorException* **usando el constructor**
- *testEmpleadoConstructorException* **usando el constructor**
- *testProductoConstructorException* **usando el constructor**
- *testMaquinariaConstructorException* **usando el constructor**
- *testMaquinariaHorasNegativasException* **usando el constructor**
- *testMaquinariaHectareasNegativasException* **usando el constructor y calcularTiempoArado**
- *testEmpleadoAumentoNegativoException* **usando el constructor y darAumento**
- *testProductoDescuentoInvalidoPorAbajoException* **usando el constructor y calcularPrecioConDescuento**
- *testProductoDescuentoInvalidoPorArribaException* **usando el constructor y calcularPrecioConDescuento**

Tests Integración

IntegrationTest

Crear un objeto de cada tipo y comprobar mediante asserts que los getters y setters funcionan correctamente.

Java Doc

1. Realiza la documentación asociada a las clases:
    a. Cultivo
    b. Empleado
    c. Maquinaria
    d. Producto
2. La documentación debe incluir al menos:
    a. Una descripción de cada una de las funciones
    b. @param
    c. @return (si retorna)
    d. @see y @link
    e. @deprecated
        i. Haremos que una función sea obsoleta creando una nueva con otro nombre y mismo código y enlazaremos una con la otra con @link
    f. @version a nivel de clase
    g. ejecutar el comando javadoc sobre el directorio de trabajo para obtener la documentación actualizada en formato html. [Ejemplo en la última diapositiva del contenido teórico.]