Practica 1 - Testing

Los Testadores

Imagina que trabajas en la creación de un juego basado en los personajes de Marvel "Los Vengadores". Para ello, debes desarrollar un pequeño sistema que gestione los datos de los héroes disponibles y permita a los usuarios seleccionar a su personaje favorito.

Requisitos del sistema

El sistema debe ser capaz de:

- Almacenar los datos de los héroes disponibles, incluyendo su nombre, superpoderes y biografía.
- Permitir a los usuarios seleccionar a su héroe favorito, y mostrar su información completa.
- Lanzar una excepción personalizada si el héroe seleccionado no está disponible.

Clases

1. `Heroe`: Esta clase representará a los héroes de Marvel. Debe tener los atributos `nombre`, `superpoderes` y `biografia`. Además, debe contar con los métodos `getNombre()`, `getSuperpoderes()` y `getBiografia()` para obtener la información de cada héroe.
2. `GestorHeroes`: Esta clase será la encargada de almacenar los datos de los héroes disponibles. Debe tener un atributo `heroes` que sea una lista de objetos `Heroe`. Además, debe contar con los siguientes métodos:
    - `agregarHeroe(Heroe heroe)`: Agrega un héroe a la lista de héroes disponibles.
    - `buscarHeroe(String nombre)`: Busca un héroe por su nombre y devuelve su objeto `Heroe` correspondiente. Si no lo encuentra, lanza una excepción personalizada `HeroeNoEncontradoException`.
3. `Main`: Esta clase será la encargada de interactuar con el usuario. Debe mostrar un menú con los héroes disponibles y permitir al usuario seleccionar uno de ellos. Una vez seleccionado, debe mostrar toda la información del héroe elegido.

Tests

En primer lugar, tenéis que implementar en `GestorHeroes` la función `getHerores`, que devuelve una lista de Héroes. De esta forma solucionamos el error de la clase `Main`.

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 un Héroe y comprobar que las funciones setter y getter funcionan.
- Tests de integración: Debes crear un test de integración que compruebe que la clase `GestorHeroes` funciona correctamente al agregar y buscar héroes.
- Tests funcionales: Debes probar el programa completo con diferentes héroes y comprobar que se muestra la información correcta en cada caso. También debes probar que se lanza la excepción `HeroeNoEncontradoException` si se intenta buscar un héroe que no está disponible.

Espero que este enunciado te resulte ameno y te ayude a crear un ejercicio divertido y educativo para tus alumnos. ¡Que la fuerza (y los superpoderes) te acompañen!

Test de actualización de héroe existente

En este test, se crea un nuevo `GestorHeroes`, se agrega un héroe (Capitán América) y luego se actualiza su descripción.

El campo `descripcion`, no existe. Debéis crearlo en el objeto correspondiente y después verificar los cambios utilizando diferentes tipos de aserciones.

Test de eliminación de héroe existente

En este test, se crea un nuevo `GestorHeroes`, se agrega un héroe (Spiderman) y luego se elimina.

Debéis implementar la función `eliminarHeroe(String nombre)` en la clase `GestorHeroes`, que eliminará un héroe de la lista dado su nombre. La función `eliminarHeroe(String nombre)` trabajará sobre la lista de la clase `GestorHeroes`.

Se verifica que el héroe haya sido eliminado correctamente utilizando diferentes tipos de aserciones.

Test de agregar varios héroes

En este test, se crea un nuevo `GestorHeroes` y se agregan varios héroes. Se verifica que los héroes hayan sido agregados correctamente utilizando la función `buscarHeroe` de `GestorHeroes` utilizando diferentes tipos de aserciones comprobando sus atributos.

Buscar héroe por superpoder

En este test, se crea un nuevo `GestorHeroes` y se agregan varios héroes.

Debemos crear una nueva función en `GestorHeroes` que es `buscarHeroePorSuperpoder(String superpoder)`. Esta función devolverá una objeto Héroe que cuyo poder coincida entre alguno de los que tiene en su lista de superpoderes.

Se verifica que los héroes hayan sido agregados correctamente utilizando diferentes tipos de aserciones para comprobar sus superpoderes.

Actualizar la información de un héroe existente

En este test, se crea un nuevo `GestorHeroes` y se agregan varios héroes. Después nos centraremos concretamente en actualizar uno en concreto.

Se debe crear la función `actualizarHeroe(Heroe heroeActualizado)`. Esta función buscará un héroe en la lista que ya exista y lo sustituirá por completo, a diferencia del test de actualización de héroe existente en el que actualizamos algunos atributos y no todos. Esta función trabajará sobre la misma lista de la clase.

Se verifica que el nuevo Héroe ha sido actualizado correctamente en la lista de héroes y que tiene los nuevos atributos.

Listar héroes

En este test, se crea un nuevo `GestorHeroes` y se agregan varios héroes.

Creamos una función `listarHeroes()` que recorrerá la lista de héroes y mostrando una lista de los que contine de la siguiente manera: `"Iron Man, Spiderman, Thor, Viuda Negra"`

Se verifica que la lista de héroes devuelta es la misma que la esperada utilizando diferentes tipos de aserciones.

Buscar héroes por superpoder

En este test, se crea un nuevo `GestorHeroes` y se agregan varios héroes.

Creamos una función `buscarHeroesPorSuperpoder(String superpoderes)` que devuelve una lista de objetos `Heroe` con los héroes que tienen los superpoderes. Esta función devolverá una lista con objetos Héroe que tengan el/los superpoderes que se piden.

Se verifica que la lista de héroes con superpoder esperada es la misma que devuelve la función. Para ello utilizaremos diferentes tipos de aserciones.

En este caso, como los Héroes pueden tener más de un superpoder y los héroes pueden tener poderes en común, se deberá comprobar que que la lista de Héroes devuelta contenga héroes con superpoderes en común.