Practica 2 - Testing

One Test

Imagina que trabajas en la programación de un juego basado en los personajes de One Piece. Para ello, debes desarrollar un pequeño sistema que gestione los datos de los personajes que se piden y permita controlar que no haya fallos.

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 batalla.

Clases

1. Las clases **Luffy** y **Zoro**, que representan a los personajes de One Piece. Ambas implementan la interfaz Personaje, que define métodos comunes como **getNombre, getPoder y recibirDanio**. Además, cada personaje tiene un método atacar, que permite atacar a otro personaje y reducir su poder.

La clase **Luffy** debe tener los siguientes métodos:

- El constructor Luffy(String nombre, int poder) *que recibe el nombre y el poder de ataque de Luffy*.
- El método getNombre() *que devuelve el nombre de Luffy.*
- El método getPoder() *que devuelve el poder de ataque de Luffy.*
- El método atacar(Personaje enemigo) *que realiza un ataque a otro personaje. Recibe como parámetro un objeto Personaje y reduce su poder de acuerdo al poder de ataque de Luffy.*

La clase **Zoro** debe tener los siguientes métodos, similares a los de la clase Luffy:

- El constructor Zoro(String nombre, int poder).
- El método getNombre().
- El método getPoder().
- El método atacar(Personaje enemigo).

2. La clase **Barco**, que tiene un nombre y una capacidad. El método **agregarTripulante()** intenta agregar un tripulante al barco. Si la capacidad es menor a 10, se incrementa en uno y retorna true, de lo contrario, retorna false.

La clase **Barco** debe tener los siguientes métodos:

- El constructor Barco(String nombre, int capacidad) que recibe el nombre del barco y su capacidad máxima de tripulantes.
- El método getNombre() que devuelve el nombre del barco.
- El método getCapacidad() que devuelve la capacidad actual de tripulantes en el barco.

- El método agregarTripulante() que intenta agregar un tripulante al barco. Este método debe cumplir con las siguientes condiciones:
- Si la capacidad actual es menor a 10, se puede agregar un tripulante y la capacidad debe incrementarse en uno.
- Si la capacidad actual es igual o mayor a 10, no se puede agregar más tripulantes.

3. La clase **Fruta**, que tiene un nombre y un indicador booleano para determinar si es deliciosa. El método **puedoComer()** verifica si la fruta es deliciosa y si no es la "Fruta del Diablo". Si ambas condiciones se cumplen, devuelve true, de lo contrario, devuelve false.

La clase **Fruta** debe tener los siguientes métodos:

- El constructor Fruta(String nombre, boolean esDeliciosa) que recibe el nombre de la fruta y un indicador de si es deliciosa o no.
- El método getNombre() que devuelve el nombre de la fruta.
- El método esDeliciosa() que devuelve true si la fruta es deliciosa o false en caso contrario.
- El método puedoComer() que verifica si la fruta es deliciosa y si no es la "Fruta del Diablo". Debe devolver true si ambas condiciones se cumplen, y false en caso contrario.

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.

Tests de Integración

Batalla Test

En la clase **TestBatalla**, utilizamos los casos de prueba de para probar el funcionamiento de los métodos de ataque.

1. El caso de prueba **testAtaqueLuffy()** verifica que el método atacar() de Luffy reduzca correctamente el poder de ataque del personaje enemigo. Debes crear un objeto Luffy con nombre "Monkey D. Luffy" y poder de ataque 100, y

un objeto Zoro con nombre "Roronoa Zoro" y poder de ataque 80. Verifica que, al atacar a Zoro, el poder de Zoro se reduzca a -20 🪦.

2. El caso de prueba **testAtaqueZoro()** verifica que el método atacar() de Zoro reduzca correctamente el poder de ataque del personaje enemigo. Utiliza los mismos objetos de Luffy y Zoro creados en el caso de prueba anterior. Verifica que, al atacar a Luffy, el poder de Luffy se reduzca a 20 🏳️.

Barco Test

En las pruebas, verificamos que el método **agregarTripulante()** se comporte correctamente.

1. El caso de prueba **testAgregarTripulante()** verifica que se pueda agregar un tripulante al barco cuando la capacidad actual es menor a 10. Debes utilizar el barco con nombre "Thousand Sunny" y capacidad inicial de 5. Verifica que el método agregarTripulante() devuelva true y que la capacidad actual se haya incrementado en uno.
2. El caso de prueba **testAgregarTripulanteMaximo()** verifica que no se pueda agregar un tripulante adicional cuando la capacidad actual es igual a 10 (capacidad máxima). Utiliza el barco con nombre "Going Merry" y capacidad inicial de 10. Verifica que el método agregarTripulante() devuelva false y que la capacidad actual se mantenga en 10.

Fruta Test

Verificamos que el método **puedoComer()** funcione correctamente en diferentes escenarios.

1. El caso de prueba **testPuedoComerFrutaDeliciosa()** verifica que el método puedoComer() devuelva true cuando la fruta es deliciosa y no es la "Fruta del Diablo". Debes crear una fruta con nombre "Manzana" y que sea deliciosa. Verifica que el método puedoComer() devuelva true.
2. El caso de prueba **testPuedoComerFrutaDelDiablo()** verifica que el método puedoComer() devuelva false cuando la fruta es la "Fruta del Diablo". Crea una fruta con nombre "Fruta del Diablo" y que sea deliciosa. Verifica que el método puedoComer() devuelva false.
3. El caso de prueba **testPuedoComerFrutaNoDeliciosa()** verifica que el método puedoComer() devuelva false cuando la fruta no es deliciosa, independientemente de si es la "Fruta del Diablo" o no. Crea una fruta con nombre "Durian" y que no sea deliciosa. Verifica que el método puedoComer() devuelva false.