GUÍA DE EJERCICIOS

DESARROLLO WEB FULL STACK CON JAVA

Nº 9

ING. LUISINA DE PAULA

Guía de Ejercicios Nº 9: Programación Orientada a Objetos - Parte 3

1) Clases abstractas

a. Crear una clase abstracta llamada consola, la cual tenga los atributos: codigo_consola, nombre, empresaDesarrollo y año de lanzamiento. Al mismo tiempo, crear un método abstracto "cargarJuego" que indique un mensaje por pantalla que diga "Cargando juego. Por favor espere".
A partir de la clase abstracta creada crear las siguientes subclases hijas:

- ✓ **Nintendo64:** La cual tiene un atributo propio norma y otro para determinar si lee cartuchos piratas. Al mismo tiempo, implementa un método propio "leerCartucho" el cual recibirá el nombre de un juego como parámetro e indicará un mensaje por pantalla indicando esta situación. Por ejemplo: "Leyendo cartucho Banjo Kazooie".
- ✓ **PlayStation2:** La cual tiene los atributos propios norma, chipeadaONo y tamañoMemoryCard. Al mismo tiempo, implementa un método propio "leerDvd" el cual recibirá el nombre de un juego como parámetro e indicará un mensaje por pantalla indicando esta situación. Por ejemplo: "Leyendo DVD Fifa 2009". Por otro lado, también implementa un método propio llamado "grabarEnMemory" el cual debe informar al usuario que se ha guardado correctamente un juego en la memory card.
- ✓ **Xbox One:** La cual tiene una serie de atributos propios para manejar: Si está conectada a internet o no, si la calidad 4K está activada y si se permite la descarga automática de actualizaciones. Al mismo tiempo implementa un método propio "leerJuegoDigital" el cual recibirá el nombre de un juego como parámetro e indicará un mensaje por pantalla indicando esta situación. Por ejemplo: "Leyendo GTA V desde tienda".

Una vez desarrolladas cada una de las clases, se solicita desde la clase Main, crear una instancia de cada una de ellas y llamar a sus correspondientes métodos. *Tener en cuenta que todas las consolas, al heredar de una clase abstracta, deben implementar el/los método/s de su clase madre utilizando sobreescritura de métodos.*

2) Clases Abstractas + Interfaces

Ejercicio Nº 1

Un fanático de Pókemon desea implementar para el modelado de un videojuego los diferentes ataques de cada una de estas criaturas. Para ello, cuenta con una clase abstracta llamada Pokemon, la cual posee los atributos: num_pokedex, nombrePokemon, pesoPokemon, sexo, temporadaQueAparece y tipo, e implementa métodos para los ataques comunes que suele tener la mayoría, entre ellos se encuentran: atacarPlacaje(), atacarArañazo() y atacarMordisco(). Sin embargo, este fanático también desarrolló una serie de interfaces para contemplar los ataques de Pókemons de cierto tipo:

- ✓ **IElectrico:** con los métodos atacarImpactrueno(), atacarPunioTrueno(), atacarRayo(), atacarRayoCarga().
- ✓ **IPlanta:** con los métodos atacarParalizar(), atacarDrenaje(), atacarHojaAfilada(), atacarLatigoCepa().
- ✓ **IFuego:** con los métodos atacarPunioFuego(), atacarAscuas(), atacarLanzallamas().
- ✓ **IAgua:** con los métodos atacarHidrobomba(), atacarPistolaAgua(), atacarBurbuja(), atacarHidropulso().

A partir de estas interfaces, el Pokefanático desea crear las clases que manejen a los personajes principales del videojuego, los cuales son los pokemons starters de la primera temporada (Charmander, Bulbasaur y Squirtle) y Pikachu; para ello tener en cuenta que: Charmander es de tipo fuego, Bulbasaur es de tipo planta, Squirtle es de tipo agua y Pikachu de tipo eléctrico.

Una vez implementadas la clase abstracta e interfaces, sobrescribir los métodos correspondientes para adaptarlos a cada Pókemon mostrando un mensaje en pantalla que indique qué pókemon es y qué ataque está realizando, por ejemplo: “Soy Charmander y estoy atacando con Ascuas” o “Soy Pikachu y estoy atacando con placaje”. Luego de realizar lo mencionado, crear las instancias necesarias y llamar a cada uno de los métodos de cada Pokemon.