Nombres: Carnés:

Luis Alejandro Contreras Paredez 9989-22-473

Eduardo José Leonardo Acuña 9989-23-2719

# Investigar sobre los tipos de Pokémon y como utilizar herencias para crear las clases necesarias, para poder tener al menos 5 tipos de Pokémon distintos

Índice:

[Introducción 3](#_Toc174974271)

[Tipos de Pokémon 4](#_Toc174974272)

[Modelado de Tipos de Pokémon mediante Herencia 5](#_Toc174974273)

[Subclases para Diferentes Tipos de Pokémon 5](#_Toc174974274)

[Ventajas de Utilizar Herencia 5](#_Toc174974275)

[Relaciones y Dependencias 6](#_Toc174974276)

[Aplicaciones Prácticas y Consideraciones 6](#_Toc174974277)

[Conclusiones 8](#_Toc174974278)

[Referencias 9](#_Toc174974279)

# Introducción

La franquicia Pokémon, creada por Satoshi Tajiri y desarrollada por Game Freak, es un fenómeno cultural que ha trascendido generaciones desde su lanzamiento en 1996. Los videojuegos, series de televisión, películas y cartas coleccionables han convertido a Pokémon en una parte integral de la cultura popular. En el corazón de esta franquicia están los Pokémon, criaturas con habilidades especiales que los jugadores pueden capturar, entrenar y utilizar en batallas. Una característica fundamental de los Pokémon es su clasificación en diferentes tipos, como Fuego, Agua, Planta, entre otros, que determinan sus fortalezas y debilidades.

En el contexto de la programación orientada a objetos (OOP), los tipos de Pokémon pueden modelarse eficazmente utilizando el concepto de herencia. La herencia permite crear nuevas clases basadas en clases existentes, facilitando la reutilización de código y la creación de jerarquías lógicas. Esta investigación tiene como objetivo explorar cómo se puede utilizar la herencia para modelar diferentes tipos de Pokémon en un entorno de desarrollo de software, proporcionando una estructura robusta y extensible para la creación de sistemas que simulen las características y dinámicas del universo Pokémon.

# Tipos de Pokémon

En el universo Pokémon, cada Pokémon se clasifica en uno o más tipos, que determinan las habilidades que posee y su rendimiento en combate. Estos tipos son esenciales para la estrategia del juego, ya que las interacciones entre los diferentes tipos (por ejemplo, Fuego es fuerte contra Planta, pero débil contra Agua) forman la base de la mecánica de las batallas. A continuación, se detallan algunos de los tipos más comunes:

* **Fuego:** Estos Pokémon generalmente tienen habilidades relacionadas con el calor, el fuego y las llamas. Son efectivos contra Pokémon de tipo Planta, Hielo, Bicho y Acero, pero son débiles frente a tipos Agua, Roca y Tierra.
* **Agua:** Los Pokémon de tipo Agua habitan en entornos acuáticos y usan ataques basados en el agua. Son fuertes contra Fuego, Roca y Tierra, pero débiles frente a Eléctrico y Planta.
* **Planta:** Pokémon que están conectados con la vegetación y la naturaleza. Utilizan ataques que emulan el crecimiento de plantas y la energía solar. Son efectivos contra Agua, Roca y Tierra, pero tienen desventajas frente a Fuego, Bicho, Volador y Veneno.
* **Eléctrico:** Estos Pokémon controlan la electricidad y pueden utilizarla en ataques poderosos. Son fuertes contra Pokémon de tipo Agua y Volador, pero débiles frente a Tierra.
* **Hielo:** Los Pokémon de tipo Hielo utilizan el frío extremo y el hielo en sus ataques. Son efectivos contra Dragón, Volador, Planta y Tierra, pero vulnerables a Fuego, Lucha, Roca y Acero.

Además de estos, existen otros tipos como Volador, Tierra, Psíquico, Veneno, entre otros, cada uno con sus propias características únicas. Las interacciones entre estos tipos crean una compleja red de fortalezas y debilidades que los jugadores deben dominar para tener éxito en las batallas Pokémon.

**Programación Orientada a Objetos (OOP) y Herencia**

La programación orientada a objetos es un paradigma de programación que organiza el software en "objetos", que son instancias de "clases". Una clase define las propiedades y comportamientos que comparten los objetos que pertenecen a ella. OOP facilita la modelación de sistemas complejos al permitir que los desarrolladores trabajen con conceptos abstractos de manera modular y organizada.

Uno de los conceptos clave en OOP es la **herencia**, que permite crear nuevas clases que heredan las propiedades y métodos de otras clases. Las clases derivadas (subclases) pueden reutilizar el código de la clase base (superclase) y agregar o modificar funcionalidades según sea necesario. En el contexto de los Pokémon, la herencia se puede utilizar para crear una jerarquía de clases que refleje la clasificación de los Pokémon en diferentes tipos.

# Modelado de Tipos de Pokémon mediante Herencia

La clase Pokemon servirá como la clase base para todos los tipos de Pokémon. Esta clase incluirá los atributos y métodos que son comunes a todos los Pokémon, independientemente de su tipo.

En esta clase se puede definir varios atributos como (nombre, nivel, salud, ataque, defensa, tipo) y métodos (atacar, recibir\_daño, subir\_nivel) que son esenciales para todos los Pokémon.

## Subclases para Diferentes Tipos de Pokémon

A partir de la clase Pokemon, se pueden crear subclases para cada tipo de Pokémon. Estas subclases heredan los atributos y métodos de Pokemon, pero también pueden definir atributos y métodos adicionales que son específicos para cada tipo.

Por ejemplo, al crear las subclases Fuego y Agua heredan de Pokemon y agregan métodos específicos (ataque\_especial\_fuego, ataque\_especial\_agua) que son únicos para estos tipos de Pokémon.

# Ventajas de Utilizar Herencia

La herencia en OOP ofrece varias ventajas importantes:

* **Reutilización de Código:** Permite evitar la duplicación de código al reutilizar atributos y métodos definidos en la clase base, lo que facilita el mantenimiento y la expansión del sistema.
* **Extensibilidad:** Facilita la adición de nuevos tipos de Pokémon al sistema, simplemente creando nuevas subclases que heredan de Pokémon. Esto es especialmente útil en sistemas que pueden necesitar escalar o adaptarse en el futuro.
* **Organización Lógica:** La herencia ayuda a organizar el código de manera lógica, siguiendo una estructura jerárquica que es fácil de entender, mantener y expandir.

# Relaciones y Dependencias

Además de la herencia, la programación orientada a objetos permite modelar relaciones y dependencias entre diferentes clases. En un sistema más complejo, los Pokémon interactúan con otros objetos, como Entrenador, Pokedex, y Piedras Evolutivas.

Por ejemplo, un Entrenador puede tener un equipo de Pokémon (List<Pokemon>) y una Pokedex que registra todos los Pokémon que ha capturado, Entrenador tiene una relación de agregación con Pokedex y una asociación con los Pokémon que captura.

# Aplicaciones Prácticas y Consideraciones

La implementación de un sistema de Pokémon utilizando herencia es aplicable no solo en videojuegos y simulaciones, sino en cualquier aplicación que requiera modelar jerarquías y relaciones complejas entre objetos. Sin embargo, es importante tener en cuenta algunos principios prácticos:

* **Polimorfismo:** En OOP, el polimorfismo permite que objetos de diferentes clases puedan ser tratados como objetos de una clase común. Esto es útil para trabajar con colecciones de objetos de diferentes subclases, como en una lista de Pokémon de distintos tipos.
* **Encapsulamiento:** Mantener los atributos de la clase privados y proporcionar métodos públicos para acceder a ellos ayuda a proteger la integridad de los datos y facilita el mantenimiento.
* **Manejo de Excepciones:** Es crucial implementar un manejo adecuado de errores y excepciones que puedan surgir durante la ejecución del programa.

Imagen de la pantalla de un video juego

Descripción generada automáticamente con confianza baja

Una captura de pantalla de un celular con letras

Descripción generada automáticamente con confianza media

# Conclusiones

La herencia en la programación orientada a objetos es una herramienta poderosa que permite modelar de manera eficiente y organizada las complejidades del universo Pokémon. Al utilizar una clase base Pokemon y extenderla para crear subclases que representen diferentes tipos de Pokémon, se puede crear un sistema robusto, flexible y fácil de mantener.

Este enfoque no solo mejora la organización del código, sino que también facilita la expansión y el mantenimiento del sistema, permitiendo agregar nuevas funcionalidades y tipos de Pokémon de manera sencilla. La herencia, junto con otros conceptos de OOP como el polimorfismo y el encapsulamiento, proporciona una base sólida para el desarrollo de aplicaciones complejas que capturan la esencia y la estrategia del mundo Pokémon.

# Referencias

* Tajiri, S., & Game Freak. (1996). *Pokémon Red and Blue*. Nintendo.
* Bulbapedia. (2024). *Type*. Retrieved from https://bulbapedia.bulbagarden.net/wiki/Type