Universidad del Valle de Guatemala
Nathalie Valle 251290

Análisis

Requisitos funcionales

1. **Dos entrenadores participan en una batalla de 4 rondas.**
   Esto para poder simular un enfrentamiento competitivo que se define por estrategia y consistencia, no solo por suerte o una sola jugada.
2. **Cada uno elige 4 Pokémon distintos al inicio.**
   Esto para que permita variedad estratégica y que pueda evitar repeticiones. Esto obliga al jugador a usar todos sus recursos, no depender de un solo Pokémon.
3. **Cada Pokémon tiene nombre, tipo, ataque, defesa y una habilidad especial.**
   Estos atributos son esenciales para representar sus capacidades y permiten cálculos de combate, diferenciando uno de otro.
4. **Las habilidades afectan ataque, defensa o hacen daño directo al rival y tienen una probabilidad de activación.**
   Esto añade un elemento táctico y de incertidumbre al juego, similar a los videojuegos originales: no siempre puedes depender de tu habilidad.
5. **En cada ronda, cada entrenador selecciona un Pokémon no repetido.**
   Esto fuerza a rotar ente los 4 Pokémon y utilizar toda la estrategia del equipo, no solo el más fuerte.
6. **El entrenador decide si atacar o usar su habilidad especial.**
   Este brinda decisiones tácticas al jugador, dándole control sobre su jugada (no es todo automático).
7. **Se considera la ventaja/desventaja de tipo (+20, -10 o 0).**
   Esto imita el sistema clásico de Pokémon, agregando lógica y profundidad al sistema de combate.
8. **Si la habilidad se activa, su efecto se aplica en esa ronda y la siguiente.**
   Esto añade persistencia táctica: esto es una buena habilidad no solo impacta una vez, sino que afecta también la estrategia futura.
9. **El ganador de la ronda es quien tenga mayor ataque total.**
   Esto define la condición clara y objetiva de victoria ene cada ronda, basada en los cálculos definidos.
10. **Se empata si el ataque total es igual.**
    Se considera la posibilidad lógica de igualdad en fuerza, manteniendo el equilibrio y evitando decisiones arbitrarias.
11. **El entrenador con más rondas ganadas gana la batalla.**
    Esto promueve la consistencia y la estrategia a largo plazo. No gana el que haga una buena jugada aislada, sino el que lo haga mejor durante todo el enfrentamiento.

Clases

Clases	Propósito
Entrenador	Gestiona el equipo de Pokémon de un jugador, mantiene el registro de cuáles han sido usados y permite seleccionar un Pokémon disponible para cada ronda. Facilita la interacción entre el jugador y sus criaturas, centralizando las decisiones de combate.

Universidad del Valle de Guatemala
Nathalie Valle 251290

Pokémon	Modela a un Pokémon con sus atributos esenciales: nombre, tipo elemental, ataque, defensa y una habilidad especial. Incluye la lógica para calcular su ataque total en combate, considerando afecto de tipo y la activación de habilidades especiales que pueden persistir.
Habilidad	Representa una habilidad especial con un nombre, tipo de efecto (ataque, defensa o daño directo), valor de impacto y probabilidad de activación. Encapsula la mecánica de activación y la aplicación de sus efectos en el combate, manteniendo esa lógica independiente del Pokémon.
Ronda	Encapsula la ejecución de una sola ronda de combate, donde dos Pokémon se enfrentan. Controla el uso y activación de habilidades, calcula el ataque total de cada contendiente y determina el ganador o empate. Permite registrar el resultado para llevar un historial detallado.
Batalla	Administra el flujo completo del enfrentamiento, coordinando las cuatro rondas entre los entrenadores. Lleva el conteo de rondas ganadas, almacena cada instancia de ronda para historial y decide el ganador final del duelo, manteniendo el control general del combate.
Juego	Orquesta la inicialización del sistema: crea los entrenadores, asigna sus Pokémon y lanza la batalla. Centraliza la configuración y preparación antes de la ejecución del combate manteniendo el main limpio y enfocado.
Main	Punto de entrada del programa, responsable únicamente de iniciar la ejecución llamando a Juego. Mantiene la separación de responsabilidades y facilita la mantenibilidad del código.

Atributos y métodos

Clase	Atributos	Propósito	Métodos	Propósito
Entrenador	• Nombre: String • Equipo: List<Pokémon> • pokemonUsados: List<Pokémon>	• Identifica al entrenador. • Lista de los 4 Pokémon que posee. • Registro de Pokémon ya usados en rondas anteriores.	• elegirPokemon(): Pokemon • registraUsoPokemon (pookemon: Pokemon): void	• Permite seleccionar un Pokémon disponible para la ronda. • Añade n Pokémon a la lista de usados.
Pokémon	• Nombre: String • tipo: String • ataque: int • defensa: int • habilidad: Habilidad • habilidadActiva: boolean	• Habilidad: referencia a su habilidad especial. • HabilidadActiva: Indica si la habilida está activa y su efecto persiste.	• calcularAtaqueTotal (oponente: Pokemon, usarHabilidad: boolean): int • activarHabilidad(): void	• Calcula el ataque total considerando tipo y habilidad. • Activa la habilidad si se cumple la probabilidad.
Habilidad	• Nombre: String • tipoEfecto: String • valor: int	• tipooEfecto: (ataque, defensa, daño)	• seActiva(): boolean	• Genera un número aleatorio y determina si la

	• probabilidad: int	• valor: magnitud del efecto. • Probabilidad: (0-100) porcentaje para que se active la habilidad.		habilidad se activa.
Ronda	• Pokemon1: Pokemon • Pokemon2: Pokemon • habilidadesUsada1: boolean • habilidadesUsada2: boolean • habilidadActivada1: boolean • habilidadActivada2: boolean • ganador: Enntrenador	• Pokémon del entrenador 1 • Pokémon del entrenador 2 • Indica si entrenador 1 decidió usar habilidad • Indica si entrenador 2 decidió usar habilidad • Si la habilidad de Pokémon 1 se activó • Si la habilidad de Pokémon 2 se activó • Entrenador que ganó la ronda (null si empate)	• Resolver(): void • getGanador(): Entrenadorr • mostrarResumen(): void	• Ejecuta la lógica de la ronda: activa habilidades, calcula ataque total y determina ganador. • Devuelve el entrenador ganador o null si hubo emapate. • Muestra detalles de la ronda (Pokémon usados, habilidades, resultado.)
Batalla	• entrenador1: Entrenador • entrenador2: Entrenador • rondasGanadas1: int • rondasGanadas2: int • rondas: List<Ronda>	• rondas : historial de las rondas jugadas.	• iniciarBatalla() : void — • jugarRonda(): void	• Controla la secuencia de las 4 rondas. • Crea una instancia de Ronda, la resuelve y actualiza resultados.
Juego	Ninguno	Ninguno	• Iniciar(): void	• Crea entrenadores, asigna Pokémon y

				comienza la batalla.
Main	Ninguno	Ninguno	• Main(String[] args): void • Juego.iniciar()	• Punto de entrada que llama a Juego.iniciar().

Todos los atributos de las clases se definen como private para aplicar el principio de encapsulamiento, evitando acceso directo desde otras clases. Los métodos públicos (public) permiten acceder o modificar estos atributos de manera controlada, principalmente a través de *getters* y *setters*, cuando sea necesario.

Getter y Setters

Todas las clases que contienen atributos privados implementan métodos de acceso (getters) y modificación (setters) para cumplir con el principio de encapsulamiento. Esto permite controlar y proteger el acceso a los datos internos de cada clase, facilitando además la mantenibilidad y extensibilidad del sistema.

En donde se aplican son las siguientes:

- Entrenador: para acceder a nombre, equipo, pokemonUsados.
- Pokemon: para acceder a nombre, tipo, ataque, defensa, habilidad, habilidadActiva.
- Habilidad: para todos sus atributos.
- Ronda: para consultar el ganador y otros estados internos.
- Batalla: para consultar resultados y rondas jugadas.

Universidad del Valle de Guate
Nathalie Valle 251290

Diseño