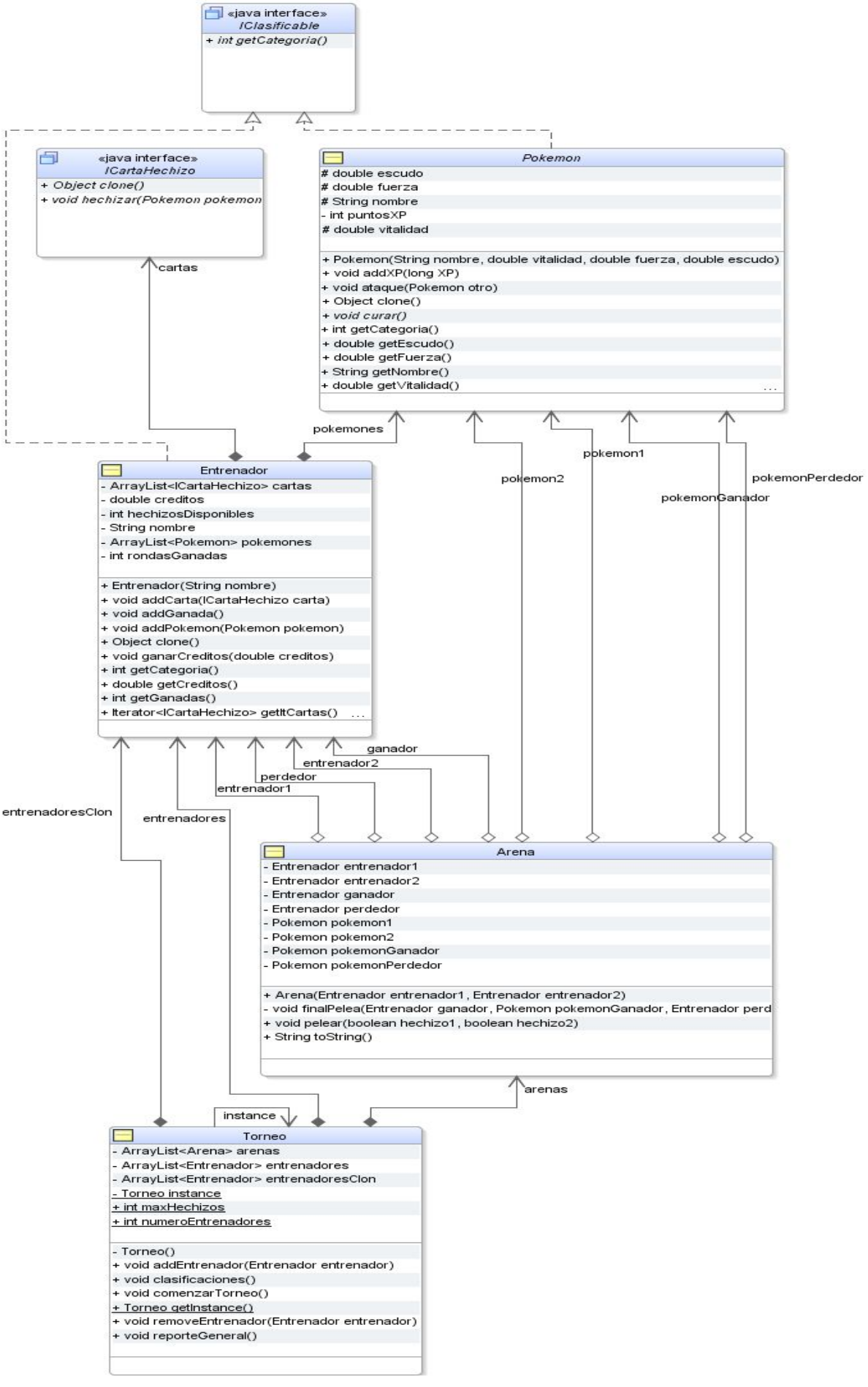
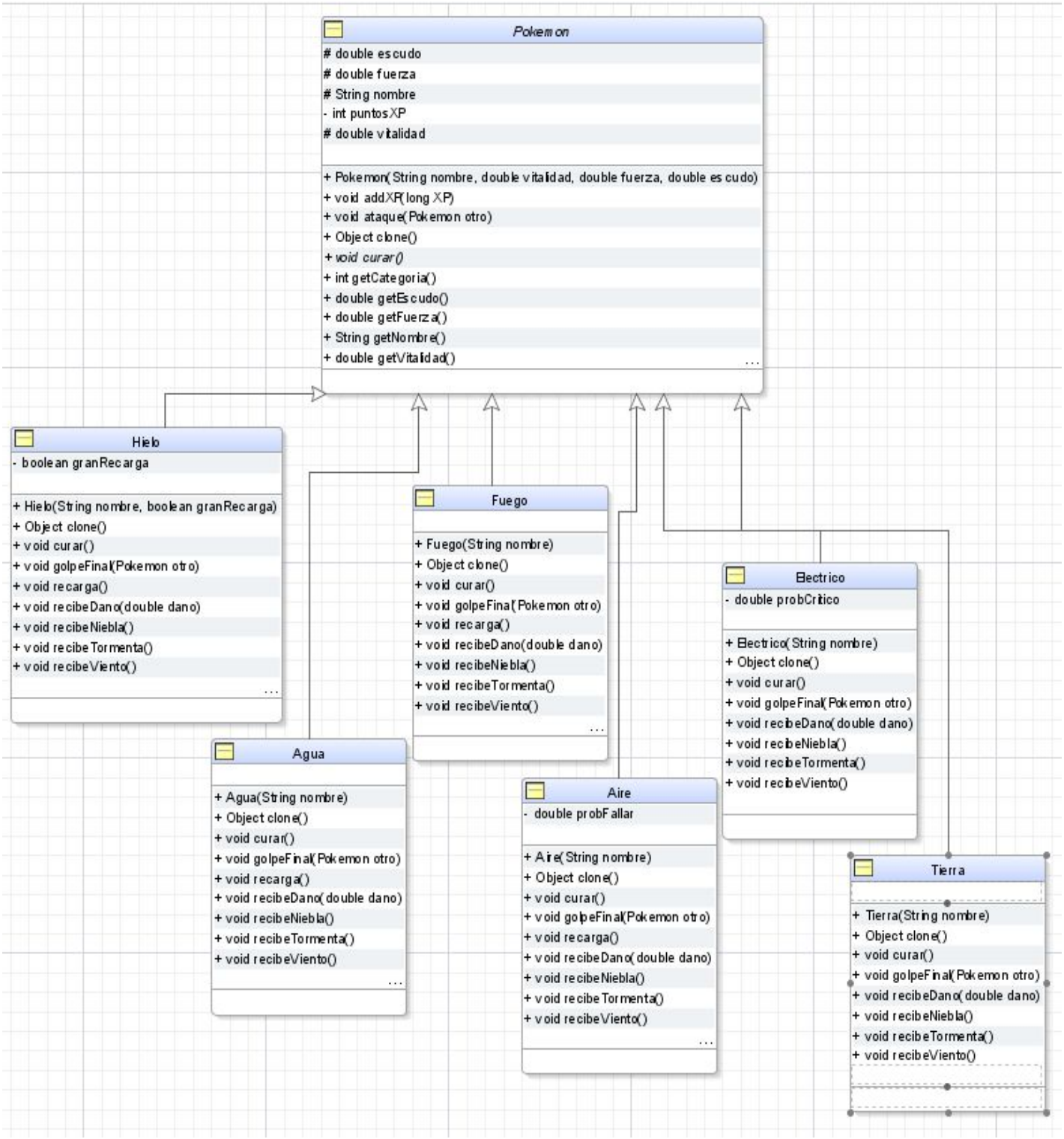
Trabajo Final
Gimnasio Pokémon

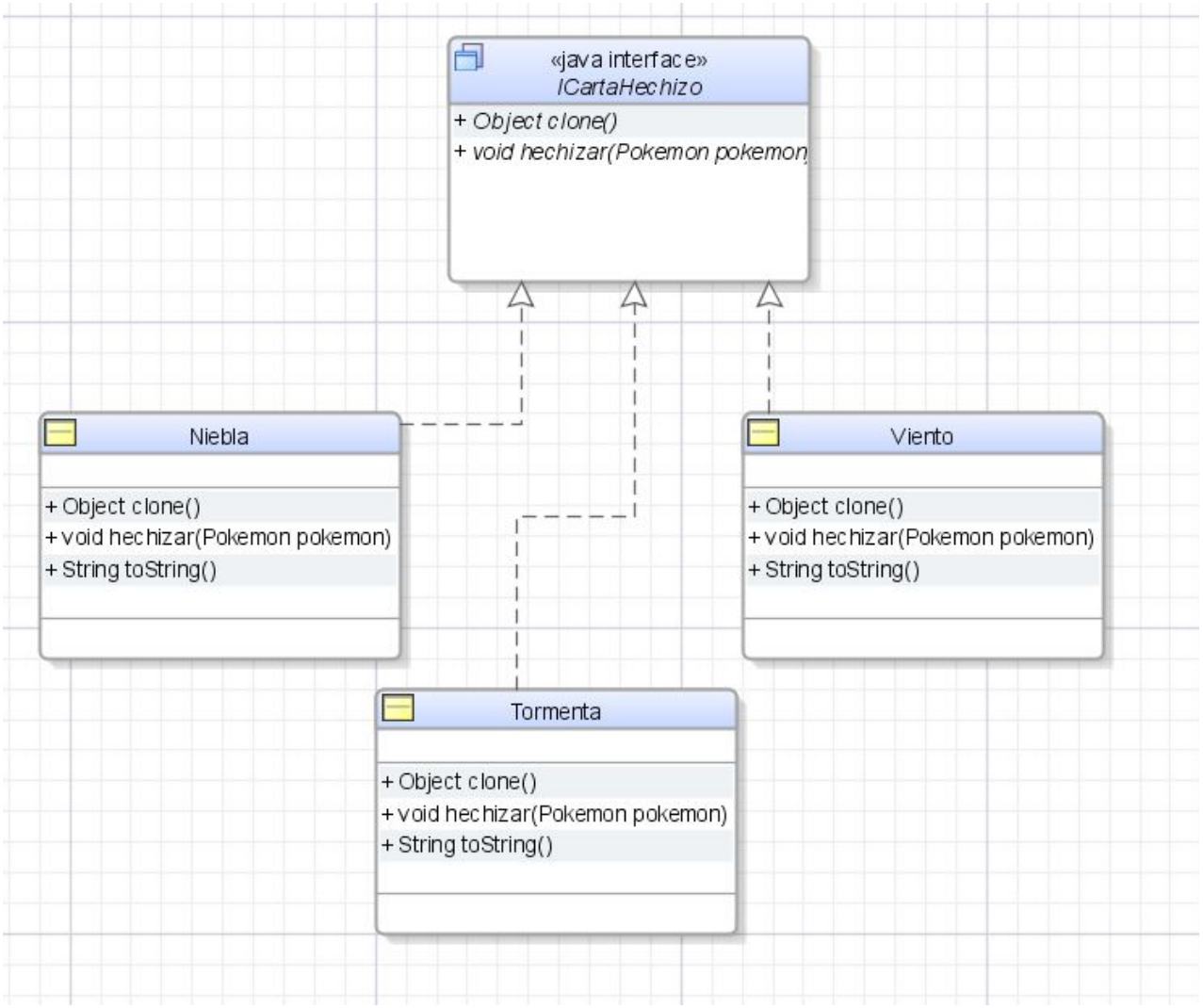
Grupo: 3
Integrantes: Fidelibus, Gabriel
Izurieta, Luciano
Casamayou, Ignacio
Fecha de entrega: 03/05/2020

Funcionamiento general del programa

Diagramas UML




Pokemon
double escudo
double fuerza
String nombre
- int puntosXP
double vitalidad
+ Pokemon(String nombre, double vitalidad, double fuerza, double escudo)
+ void addXP(long XP)
+ void ataque(Pokemon otro)
+ Object clone()
+ void curar()
+ int getCategoria()
+ double getEscudo()
+ double getFuerza()
+ String getNombre()
+ double getVitalidad()
Hielo
- boolean granRecarga
+ Hielo(String nombre, boolean granRecarga)
+ Object clone()
+ void curar()
+ void golpeFinal(Pokemon otro)
+ void recarga()
+ void recibeDano(double dano)
+ void recibeNiebla()
+ void recibeTormenta()
+ void recibeViento()
Fuego
+ Fuego(String nombre)
+ Object clone()
+ void curar()
+ void golpeFinal(Pokemon otro)
+ void recarga()
+ void recibeDano(double dano)
+ void recibeNiebla()
+ void recibeTormenta()
+ void recibeViento()
Electrico
- double probCritico
+ Electrico(String nombre)
+ Object clone()
+ void curar()
+ void golpeFinal(Pokemon otro)
+ void recibeDano(double dano)
+ void recibeNiebla()
+ void recibeTormenta()
+ void recibeViento()
Agua
+ Agua(String nombre)
+ Object clone()
+ void curar()
+ void golpeFinal(Pokemon otro)
+ void recarga()
+ void recibeDano(double dano)
+ void recibeNiebla()
+ void recibeTormenta()
+ void recibeViento()
Aire
- double probFallar
+ Aire(String nombre)
+ Object clone()
+ void curar()
+ void golpeFinal(Pokemon otro)
+ void recarga()
+ void recibeDano(double dano)
+ void recibeNiebla()
+ void recibeTormenta()
+ void recibeViento()
Tierra
+ Tierra(String nombre)
+ Object clone()
+ void curar()
+ void golpeFinal(Pokemon otro)
+ void recibeDano(double dano)
+ void recibeNiebla()
+ void recibeTormenta()
+ void recibeViento()


«java interface»
ICartaHechizo
+ Object clone()
+ void hechizar(Pokemon pokemon)
Niebla
+ Object clone()
+ void hechizar(Pokemon pokemon)
+ String toString()
Viento
+ Object clone()
+ void hechizar(Pokemon pokemon)
+ String toString()
Tormenta
+ Object clone()
+ void hechizar(Pokemon pokemon)
+ String toString()

Clases (17 en total)

POKEMONES (7)

●Clase padre Pokémon

●Agua

●Aire

●Eléctrico

●Fuego

●Hielo

●Tierra

TIPOS DE CARTAS (3)

●Niebla

●Viento

●Tormenta

INTERFACES (2)

●ICartaHechizo

●IClasificable

EXCEPCIONES (2)

●LimiteHechizosException

●NoTieneCartasException

●CloneNotSupportedException

OTRAS (3)

●Entrenador

●Arena

●Torneo

Clase Entrenador

Clase que implementa las interfaces IClasificable y Cloneable. Contiene conjuntos de Pokemones y cartas en forma de ArrayList, los cuales pueden ser listados a través de los métodos mostrarPokemones() y mostrarCartas() respectivamente.

Al momento de ingresar a la arena el entrenador tiene la obligación, para continuar en el torneo, de tener pokemones disponibles. De ser así al momento de ser elegido para enfrentarse a otro entrenador se seleccionara aleatoriamente un Pokémon para la pelea. En caso contrario el entrenador es eliminado. También es aleatorio el uso de cartas de hechizo. Si el entrenador usa una carta esta se selecciona mediante el método sacarCartaRandom(), el cual puede llegar a retornar una carta de tipo ICartaHechizo o lanzar dos tipos de excepciones: LimiteHechizosException (cuando ya se alcanzó el límite de cartas utilizables por el entrenador en el torneo) y NoTieneCartasException (cuando el entrenador no tiene ninguna carta disponible). Ambas excepciones se extienden de la excepción Exception.

La clonación del entrenador es profunda y no siempre se puede realizar, debido a que hay pokemones que no son clonables. Al ganar en la arena, el entrenador sumará créditos y su

pokemon experiencia, mientras que al perder, el entrenador se elimina de la lista de entrenadores del torneo.

Clase Pokemón

Clase abstracta que implementa las interfaces IClasificable y Cloneable, de la cual se extienden los diferentes tipos de Pokemones. En un principio se planteó la posibilidad de implementar el patrón Decorator para diferenciar los tipos de Pokemones e ir agregando funcionalidades a la clase principal, pero luego de analizarlo con más detenimiento simplemente usando herencia no se daba una explotación de clases que justifique su uso. Se encarga de gestionar los atributos, de la estructura del ataque y de la implementación del golpe inicial.

Cada instancia de la clase tiene un estado inicial en común, y este se va a ir modificando a medida que peleen en la arena. Los atributos que representan a los Pokemones son: nombre, vitalidad, fuerza y escudo.

La forma de ataque está implementada mediante el patrón Template. Este está formado por un algoritmo general con tres pasos: golpe inicial, recarga y golpe final.

Está garantizado, mediante las pre condiciones, que el ataque siempre será hacia una variable de tipo Pokemon (o de sus subclases) no nula.

Todas las subclases que se extienden de la clase Pokémon son clases concretas.

Método public void ataque(Pokemon otro)

- ❏ golpeInicial: Primer paso del algoritmo general. Se emplea un golpe inicial que es genérico para todos los Pokemones. Este está implementado en la clase abstracta Pokemon, y por lo tanto, ninguna subclase necesita sobreescribirlo. Este golpe consiste en dañar al Pokémon enemigo por los puntos de fuerza del Pokémon atacante, y luego este atributo será reducido a la mitad.

- ❏ recarga: Segundo paso del algoritmo general. En la clase padre está implementado el método genérico para los tipos de pokemones que no recargan, indicándose mediante una salida por pantalla. Los que sí lo hacen sobrescriben este método mediante un hook.

Lista de tipos de Pokemones con sus respectivas recargas:

- ★ Aire: Su fuerza se regenera 125% y su escudo 100%
- ★ Agua: Su fuerza se regenera 10% y su vitalidad 10%
- ★ Fuego: Su fuerza se regenera 10% y su vitalidad 10%
- ★ Hielo*: Su fuerza se establece en 400 puntos si posee el atributo granRecarga.En caso contrario su fuerza y su vitalidad se regeneran en 10%.
- ★ Eléctrico: No recarga.
- ★ Tierra: No recarga.

*Hielo: El atributo granRecarga es un boolean que se envía como parámetro al constructor de este tipo de Pokemon.

❑ golpeFinal: Tercer y último paso del algoritmo general. Este método está declarado como abstracto, por lo cual cada subclase tiene la obligación de implementarlo. Todos los tipos de Pokemones realizan golpes finales diferentes, donde algunos tienen probabilidades de fallar o probabilidad de un golpe crítico (daño aumentado proporcional a su fuerza).

Lista de tipos de Pokemones con sus respectivos golpes finales:

★ Aire*: Tiene una probabilidad de fallar de 10%. Si no lo hace, dañará al enemigo con el total de la fuerza del Pokémon.

★ Agua: Daña al enemigo con el total de su fuerza y luego este atributo se reduce a la mitad. Idéntico al golpeInicial.

★ Fuego: Daña al enemigo con el 125% de su fuerza actual, y luego este atributo se vuelve nulo.

★ Hielo: Daña al enemigo con el 90% de su fuerza. Si el Pokémon tiene la habilidad de granRecarga, el atributo fuerza vuelve a su valor base (100 puntos).

★ Eléctrico*: Tiene una probabilidad de crítico del 25%. Si lo hace dañará al enemigo con el doble de su fuerza actual, y sinó lo hará con el valor de su fuerza actual.

★ Tierra: Daña al enemigo con el 300% de su fuerza.

*Aire: La probabilidad de fallar (10%) está dada por un atributo privado double probFallar.

*Eléctrico: La probabilidad de golpe crítico (25%) está dada por un atributo privado double probCritico.

Al realizar un Pokémon el golpe inicial o golpe final, el Pokémon que recibe el ataque asimilará ese daño de diferentes maneras, dependiendo de su tipo. Está funcionalidad está determinada por el método abstracto recibeDano (double dano), que sobreescribe cada subclase.

Lista de tipos de Pokemones con sus respectivos recibeDano.

★ Aire: Tiene una probabilidad de esquivar de 20%. Si lo hace no recibirá daño, sinó se le descontará la cantidad total del daño al escudo, y cuando este sea 0, comenzará a decrementar su vitalidad.

★ Agua: Se le descontará la cantidad total del daño al escudo, y cuando este sea 0, comenzará a decrementar su vitalidad.

★ Fuego: Su escudo y su vitalidad bajarán en proporción a la mitad del daño recibido.

★ Hielo : Su escudo se decrementará en proporción de un 75% del daño recibido, y en un 25% la vitalidad.

★ Eléctrico: Su escudo se decrementará en proporción de un 10% del daño recibido, y en un 90% la vitalidad.

★ Tierra: Su escudo se decrementará en proporción de un 80% del daño recibido, y en un 20% la vitalidad.

En todos los casos, en el momento en que el escudo sea nulo, el daño afectará en forma total a la vitalidad.

Otra funcionalidad que se suma a ésta clase es la de recibir una maldición por parte de una carta hechizo. La forma implementada es mediante el patrón Double Dispatch, donde se declaran 3 métodos abstractos: recibeNiebla, recibeViento y recibeTormenta. Estos métodos deberán ser implementados en cada tipo de Pokémon, para definir cómo les afectará el hechizo.

Al aplicar este patrón se decidió no realizar una interfaz sino que directamente se le agregaron los tres métodos a la clase Pokemon, ya que no hay otro tipo de objeto que pueda ser hechizado. De caso contrario se hubiese usado una interfaz IHechizable la cual implementaría Pokemon.

Cada clase es invulnerable a un hechizo específico.

Lista de tipos de Pokemones con sus respectivos recibeNiebla.

- ★ Aire: Se duplica la probabilidad de fallar.
- ★ Agua: La fuerza se reduce en un 80%
- ★ Fuego: No le afecta.
- ★ Hielo : La fuerza se reduce en un 30%
- ★ Eléctrico: Probabilidad de crítico se reduce un 60%
- ★ Tierra: No afecta.

Lista de tipos de Pokemones con sus respectivos recibeViento.

- ★ Aire: No afecta.
- ★ Agua: El escudo se reduce en un 15%
- ★ Fuego: El escudo se reduce en un 50%
- ★ Hielo : No afecta.
- ★ Eléctrico: La vitalidad se reduce en un 15%
- ★ Tierra: El escudo se reduce en un 33%

Lista de tipos de Pokemones con sus respectivos recibeTormenta.

- ★ Aire: La vitalidad se reduce en un 20%
- ★ Agua: No afecta
- ★ Fuego: La vitalidad se reduce en un 10%
- ★ Hielo : El escudo se reduce en un 25%
- ★ Eléctrico: No afecta.
- ★ Tierra: La vitalidad se reduce en un 15%

Cuando un hechizo no le afecta a un tipo de Pokémon en particular, se informa por pantalla en el método sobrescrito.

Cuando un hechizo es aplicado sobre un Pokémon, este quedará hechizado durante todo el Torneo.

Ésta clase también contiene un método abstracto curar, en el cual establece la salud del Pokémon en su valor base.

Con respecto a la clonación, la clase Pokémon realiza la sobrescritura del método clone() de la clase Object, para hacerlo público.

Tipos de Pokemones clonables y no clonables:

CLONABLE	NOCLONABLE
AGUA	AIRE
HIELO	
FUEGO	TIERRA
ELECTRICO	

En las clases que no son clonables se lanza una nueva excepción de tipo CloneNotSupportedException, donde se envía un mensaje informando que la clase no puede realizar la clonación.

Tabla de valores iniciales de los diferentes tipos de Pokemones:

Tipo	Vitalidad	Escudo	Fuerza
Aire	500	40	40
Agua	500	100	120
Electrico	600	50	80
Fuego	530	200	80
Hielo	500	120	100
Tierra	700	150	20

Interfaz ICartaHechizo

Interfaz que se encarga de definir el método para realizar el doble dispatch entre los tipos de pokemones y de cartas. Mediante su método void hechizar(Pokemon pokemon) se logra hechizar a un Pokémon. Dicho método es desarrollado en cada una de las clases que implementan esta clase (Niebla, Viento y Tormenta). La implementación funciona debido al polimorfismo, ya que por cada tipo de Pokémon se ejecutará un método distinto. Esta selección del método a ejecutar se da en tiempo de ejecución. A su vez, cada clase que implementa esta interfaz también implementa Cloneable, lo cual indica que todas las cartas son clonables.

Interfaz IClasificable

Interfaz que se encarga de hacer que tanto los Pokemon como los entrenadores sean clasificables. En el primer caso su categoría depende de sus puntos de experiencia, mientras que en el segundo es la suma de las categorías de todos sus Pokemon vivos. Cuando un Pokemon muere y es eliminado de la lista de Pokemones de un entrenador, la categoría de este último se reduce.

Clase LimiteHechizosException y NoTieneCartasException

Se extienden de la clase Exception propia de Java. Sus constructores (públicos) reciben un mensaje de tipo String el cual incluye el nombre del entrenador que solicitó usar una carta pero alcanzó su límite máximo de cartas en el torneo o no tiene cartas en su mazo, respectivamente. Este mensaje puede obtenerse usando e.getMessage(), siendo e una instancia de la clase.

Clase CloneNotSupportedException

Clase propia de Java. Se usa con un mensaje de tipo String pasado por constructor que especifica cuál es el tipo de Pokémon que no se puede clonar. Este mensaje puede obtenerse usando e.getMessage(), siendo e una instancia de la clase.

Clase Arena

Se encarga de manejar la batalla entre dos entrenadores pasados por parámetro a su constructor.

Una vez creada la instancia de Arena se llama al método pelea(boolean Hechizo1, boolean Hechizo2), con los dos atributos pasados por parámetro indicando si el respectivo entrenador usa una carta o no. Este último método maneja la selección y uso de Pokémon y cartas aleatorias, el ciclo ataque-contraataque (quién ataca primero es aleatorio), la decisión del ganador y sus premios, y la eliminación del perdedor.

El ganador se decide por una fórmula que tiene en cuenta la vitalidad, la fuerza y el escudo de cada Pokémon al finalizar la pelea (Vitalidad + 2*Fuerza + 0.5*Escudo). En caso de empate gana el Entrenador 1.

Al ganar el entrenador gana la carta utilizada por su contrincante (si es que usó una), gana créditos de acuerdo a la categoría de este último (multiplicada por 0.4), y su Pokemon gana 3 puntos de experiencia, mientras el Pokemon del perdedor gana solo un punto.

Cabe destacar que al utilizar una carta esta es eliminada del mazo de su usuario, y que si un Pokémon llega a 0 o menos de vitalidad muere, independientemente de si ganó o no la pelea. Si un Pokémon gana la pelea y no muere, se cura restaurando su vitalidad original.

La clase también sobrescribe el método toString, el cual devuelve el ganador y perdedor de la pelea con sus respectivos Pokemones. Este método solo debe ser llamado luego de finalizar la pelea.

Clase Torneo

Implementa el patrón Singleton para tener una sola instancia de la clase accesible desde cualquier parte del programa.

Se encarga de la estructura general del torneo y del manejo de las listas de arenas y entrenadores. Esta última tiene un clon no profundo llamado entrenadoresClon utilizado para no perder la referencia a cada entrenador a medida que van siendo eliminados.

La clase tiene el método comenzarTorneo() el cual solo se ejecuta completamente si se agregaron la cantidad exacta de entrenadores definida por la constante maxEntrenadores. Su algoritmo tiene el objetivo de seleccionar dos entrenadores distintos al azar y crear una arena con ellos para hacerlos pelear, decidiendo si cada entrenador usará una carta de manera aleatoria. Si un entrenador seleccionado no tiene Pokemones es eliminado del torneo y se selecciona otro. El método termina cuando queda sólo un entrenador, el cual es mostrado por pantalla.

También existe el método reporteGeneral() el cual muestra el resultado de cada ronda a partir del toString implementado en la clase Arena, y el método clasificaciones() el cual imprime la categoría de cada entrenador que participó en el torneo.

Conclusiones

Durante el transcurso del proyecto se tuvieron que hacer varias decisiones en cuanto a la implementación de carácteristicas del programa que no estaban explícitamente requeridas por la consigna. Entre ellas:

-Se trató de balancear los atributos de los tipos de Pokémon agregados de manera que ninguno tenga ventaja.

-Se randomizaron varios aspectos como la selección de cartas, Pokemones y entrenadores para evitar que la prueba del sistema se vuelva tediosa con el uso de interacciones con el usuario por consola o por una interfaz visual.

-Se diseñó un torneo por selección aleatoria de batallas y eliminación directa en vez de uno por llave o por puntajes. Esto permite la posibilidad de que un entrenador peleé varias veces más que sus contrincantes, lo cual lo pondría en desventaja si sus Pokemones se debilitaría. Por esta razón se implementó la curación al finalizar cada pelea.

-Se implementó un sistema de créditos los cuales se otorgan al ganador de cada pelea. Si se expandiera el alcance del proyecto estos se podrían gastar para comprar Pokemones o cartas.