

## Refactorización del código del proyecto Pokemon :

## • Renombrado

- Renombrado de respuesta a multiplicador para mas expresividad

```

@eloymartos
fun efectividad(otro:Tipo):Double{
    var respuesta = 0.0
    if (supereficaces[otro.tipo]?.contains(this.tipo) == true) respuesta = 2.0 else{
        if (listanomuy[otro.tipo]?.contains(this.tipo) == true) respuesta = 0.5 else respuesta = 1.0
    }
    return respuesta
}

```

- 

```

fun efectividad(otro:Tipo):Double{
    var multiplicador = 0.0
    if (supereficaces[otro.tipo]?.contains(this.tipo) == true) multiplicador = 2.0 else{
        if (listanomuy[otro.tipo]?.contains(this.tipo) == true) multiplicador = 0.5 else multiplicador = 1.0
    }
    return multiplicador
}

```

- 

- Renombrado de danio a potencia en ataque porque nos daba toc y porque es mas expresivo, danio suena mas a representar el daño definitivo que hace el ataque

```

@eloymartos
class Ataque(nombre: String, tipo:Tipo, danio:Int) {

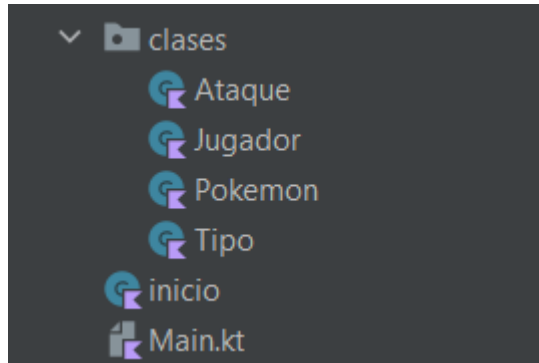
    var nombre : String
    var tipo : Tipo
    var potencia : Int

    @eloymartos *
    init {
        this.nombre = nombre
        this.tipo = tipo
        this.potencia = danio
    }
}

```

-

- Sustitucion de bloques
  - -
- Campos encapsulados
  - Ya lo están
- Movimieto de clases



- 
- Extraer interfaz
  - -
- Cambios de parámetros
  - Efectividad ahora recibe directamente el ataque y saca por su cuenta el tipo, para que la comprension en recibir ataque sea mejor

```
fun efectividad(otro:Tipo):Double{
    var multiplicador = 0.0
    if (supereficaces[otro.tipo]?.contains(this.tipo) == true) multiplicador = 2.0 else{
        if (listanomuy[otro.tipo]?.contains(this.tipo) == true) multiplicador = 0.5 else multiplicador = 1.0
    }
    return multiplicador
}
```

```
fun efectividad(ataque: Ataque):Double{
    var multiplicador = 0.0
    if (supereficaces[ataque.tipo.tipo]?.contains(this.tipo) == true) multiplicador = 2.0 else{
        if (listanomuy[ataque.tipo.tipo]?.contains(this.tipo) == true) multiplicador = 0.5 else multiplicador = 1.0
    }
    return multiplicador
}
```

```
fun recibir_ataque(ataque: Ataque) {
    vida -= (ataque.potencia*tipo.efectividad(ataque))
}
```

- 
- Mover a otro nivel
  - -
- Borrado seguro
  - Borrado de fuerza de la clase pokemon, no se usa en ningun momento

```
class Pokemon(nombre:String, lore:String, tipo:Tipo, fuerza:Int, vida:Int) {
    var nombre : String
    var lore : String
    var tipo : Tipo
    var fuerza : Int
    var vida : Double
}
```

```
class Pokemon(nombre:String, lore:String, tipo:Tipo, vida:Int) {
    var nombre : String
    var lore : String
    var tipo : Tipo
    var vida : Double
}
```



## Refactorizacion de codigo COMPLETO de pokemon

- Renombrado

```
/**
 * ### Funcion [seleccionarPokemonAutomatico]
 *
 * @return Funcion la cual selecciona el equipo de los 6 Pokemons de forma automatica, usada por
 */
// José Manuel Luque González
fun seleccionarPokemonAutomatico(lista : Array<Pokemon>, entrenador: Entrenador, numero : Int){
    println("RIVAL")
    var opcion :Int

}

/**
 * ### Funcion [seleccionarPokemonIA]
 *
 * @return Funcion la cual selecciona el equipo de los 6 Pokemons de forma automatica, usada p
 */
// José Manuel Luque González *
fun seleccionarPokemonIA(lista : Array<Pokemon>, entrenador: Entrenador, numero : Int){
    println("RIVAL")

}

112 /**
113  * ### Funcion [turno_automatico]
114  *
115  * @return Es el turno que usa la IA, es automatico entero
116  */
117 // José Manuel Luque González +1
118 fun turno_automatico // (entrenador: Entrenador, numero: Int, rival:Entrenador){
119     turnoAutomatico
120     when((1 ≤ .. ≤ 2).random()){
121         1->{

}

/**
 * ### Funcion [turnoIA]
 *
 * @return Es el turno que usa la IA, es automatico entero
 */
// José Manuel Luque González +1 *
fun turnoIA(entrenador: Entrenador, numero: Int, rival:Entrenador){
    println("turno del jugador $numero!")
    Thread.sleep( millis: 1000)
```

○

```
/**
 * ### Funcion [ataqueAutomatico]
 *
 * @return Funcion que usa la IA para atacar.
 */
@eloymartos
fun ataqueAutomatico // 📄():Ataque{
    return ataques.random()
}
```

○

```
/**
 * ### Funcion [ataqueIA]
 *
 * @return Funcion que usa la IA para atacar.
 */
@eloymartos
fun ataqueIA // 📄():Ataque{
    return ataques.random()
}
```

○

```
fun asignarAtaques(listamovimientos : Array<Ataque>){
    var contador = 0
    while (true){
        var i = listamovimientos.random()
        if(tipo.efectividad(i) == 1.0 || tipo == i.tipo){
            ataques.add(i)
            contador ++
            if(contador == 4) break
        }
    }
}
```

○

```
fun asignarAtaques(listamovimientos : Array<Ataque>){
    var contador = 0
    while (true){
        var nuevo = listamovimientos.random()
        if(tipo.efectividad(nuevo) == 1.0 || tipo == nuevo.tipo){
            ataques.add(nuevo)
            contador ++
            if(contador == 4) break
        }
    }
}
```

○

```
eloymartos  
  
fun sacarPokemon():Pokemon{  
    return equipo[pokemonEnCampo]!!  
}
```

○

```
66      /**  
67      * ### Función [pokemonActual]  
68      *  
69      * @return Cambia al pokemon activo en bat  
70      */  
eloymartos *  
71      fun pokemonActual():Pokemon{  
72          return equipo[pokemonEnCampo]!!  
73      }  
74
```

○

- Sustitucion de bloques
  - Funciones del main = combate1vs1, combatevsIA, turno, turnoia, seleccionarPokemon, seleccionarpokemonIA
- Campos Encapsulados
  - -
- MOVIMIENTO DE CLASES
  - Ya hecho en la refactorizacion anterior
- Extraer interfaz
  - -
- CAMBIO DE PARÁMETROS
  - -
- Mover a otro nivel
  - Clase tipo ahora es abstracta y ataque y pokemon heredan de ella

```
eloymartos +1 *  
↓ abstract class Tipo(cadena:String) {  
    eloymartos
```

○

```

class Ataque(nombre: String, tipo:String, danio:Int):Tipo( cadena: "" ) {

    var nombre : String
    var potencia : Int

    init {
        this.nombre = nombre
        this.tipo = tipo
        this.potencia = danio
    }
}

class Pokemon(nombre:String, lore:String, tipo:String, vida:Int):Tipo( cadena: "" ) {

    var nombre : String
    var lore : String
    var vida : Double
    var ataques : MutableList<Ataque> = mutableListOf<Ataque>()

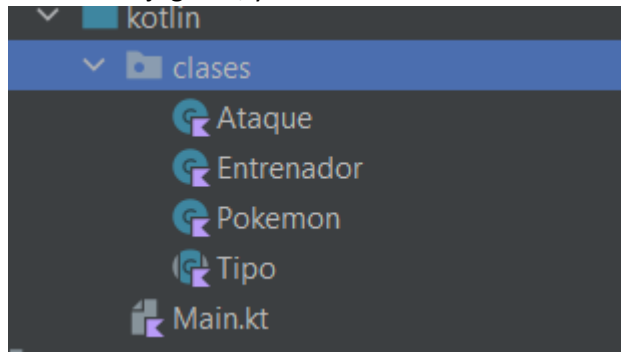
    init {
        this.nombre = nombre
        this.lore = lore
        this.tipo = tipo
        this.vida = vida.toDouble()
    }
}

```

- Las funciones en las que se accedía a este mediante atributos se han cambiado para su correcto funcionamiento

- BORRADO SEGURO**

- Borrado de jugador, ya no lo usamos



- Borrado de opciones de mas de seleccionarpokemonIA

```

fun seleccionarPokemonIA(lista : Array<Pokemon>, entrenador: Entrenador) {
    println("selección")
    var opcion : Int
    for (i in 0..12) {
        opcion = (1..12).random()
        when (opcion) {
            1 -> {entrenador.agregarPokemon(lista[0])}
            2 -> {entrenador.agregarPokemon(lista[1])}
            3 -> {entrenador.agregarPokemon(lista[2])}
            4 -> {entrenador.agregarPokemon(lista[3])}
            5 -> {entrenador.agregarPokemon(lista[4])}
            6 -> {entrenador.agregarPokemon(lista[5])}
            7 -> {entrenador.agregarPokemon(lista[6])}
            8 -> {entrenador.agregarPokemon(lista[7])}
            9 -> {entrenador.agregarPokemon(lista[8])}
            10 -> {entrenador.agregarPokemon(lista[9])}
            11 -> {entrenador.agregarPokemon(lista[10])}
            12 -> {entrenador.agregarPokemon(lista[11])}
        }
    }
    entrenador.mostrarEquipo()
}

```

- Borrar el metodo toString de pokemon, no se usaba