TP N° 1 - Chinpokomon

Año: 2022

Profesor: Rodas Bonjour, Alan

Integrantes: Bertolani Camila, Hirribarren Juan Martin y Molina Claudio.

1- Arme un breve informe sobre los patrones aplicados en el desarrollo, cuáles y dónde fueron usados y por qué motivo. Puede agregar información adicional sobre su arquitectura, o sobre opciones de diseño descartadas.

2- Arme un informe sobre los patrones de diseño que aplicó en python, y comente: ¿qué patrones quedaron igual? ¿cuáles cambiaron por las diferencias del lenguaje?

1) Los patrones aplicados en el desarrollo del trabajo fueron:

- **Builder**: Se encarga de la construcción de Chimpokomon, donde setea lista de ataques, nombre, vida, ventaja (Naturaleza) . Nos pareció más escalable para la creación de diferentes Chimpokomones, es encapsulado, ya que el builder no deja puedas acceder a su seteo, porque tiene una variable privada que se va que se va instanciando nuevos chimp con cada llamada a ese construccion
- **creadorDirector(parte del builder**): se encarga del seteo a través del builder, tiene clases predefinidas con los valores según lo que se quiere construir
  También se encarga de la construcción de ataques(son privados), para luego agregar dentro de la creación de Chimpokomones
- **Decorator (ataques) :** Los ataques pueden ser del tipo común (CONCRETE) o pueden tener algún decorado, como críticos, daño elevado o regeneración de HP etc,
- **La clase AtaqueConcret** tiene los gets (nombre, recuperación, danio, ventaja) y el método *realizarAtaque()* que en su implementación básica, solo realiza danio de atacante a atacado
- **La clase Decorator,** es un clase abstracta, la cual tiene una variable protegida, para almacenar un ataque en el momento de instanciarlo, sus métodos son iguales al concrete, solo que hace lookup a los de concrete
  Agrega la capacidad de al momento de realizar ataque, comprobar si este tiene algún efecto(osea son del tipo decorado) que modifique el daño final a realizar.
- Tiene en este caso **ConcreteDecorators**, que son:
  -ProbYDanioElevado -modifica danio
  -ProbYDanioVerdadero -modifica danio
  -PomadaWassington -modifica vida del que lo usa.
- **Logger:** es de tipo **Singleton**, ya que existe una sola vez
  contiene una variable estado, donde se guarda el tipo de estado actual, debug, info, warn, error, y se van alternando usando el patrón **State**
  contiene 4 metodos info debug, warn, error, que reciben un mensaje por parámetro e imprimen, pero solo van a salir por consola, según el seteo de estado que tenga
- **Naturaleza:** es una clase abstracta de tipo **Singleton** que tiene 3 hijos Robot Animal y Cosa el cual hacen override a el método getVentaja para saber para obtener la clase contra la cual tiene ventaja, y en caso de que no este instanciada, la crea por primera vez. Naturaleza es usada por creadorDirector, para la creación de chimp

NUMALEATORIO es una clase que solo se encarga de devolver un num aleatorio para simular batallas, eligiendo un ataque random en cada ejecución

2) Los patrones que cambiaron fueron, el singleton, ya que si se trata de implementar un singleton dentro una clase, no se pueden pasar parámetros al momento de instanciar, solo se puede lograr teniendo una clase padre que almacene instancias en un diccionario interno, usando de key la instancia y de valor, la instancia con sus parámetros

el Builder, tuvimos que crear una variable dentro del init chimpokomon = None y luego el reset porque no tiene variables de clase