Informe de patrones

Patrones utilizados en Java

* **Singleton**
* **Template**

**Singleton en Java**

Este patrón fue aplicado para la clase “Logger”, se pedía que este sea siempre el mismo y pueda ser llamado desde cualquier lugar del modelo. A diferencia de un Singleton tradicional, este presenta 2 getInstance(): Logger.getInstance(nivelDeLogueoInicial) o Logger.getInstance().

Nivel De Logger: Puede ser INFO, WARNING, ERROR, DEBUG. Si se llama a la instancia por primera vez sin valor, se le asigna nivel INFO por defecto, luego, quedara en el nivel asignado hasta que haga llamada a Logger.setNivel(nivelDeLogueo). Ej: si hago por primera vez Logger.getInstance(Nivel.DEBUG), el Logger iniciara en nivel DEBUG, si luego llamo a Logger.getInstance(Nivel.ERROR), el Logger seguira siendo nivel DEBUG.

**Template En Java**

Se usa este patrón para llamar al método de la clase abstracta “Ataque” al llamar atacar(Chimpokomon chipo), las subclases hacen lo mismo en este método y luego llaman a atacar(Chimpokomon chipo, Integer valor) donde hacen distintos cálculos para enviar el valor final del daño en chipo.recibirDanio(Integer valor).

Patrones utilizados en Python

* **Singleton**
* **Template**

**Singleton en Python:**

A diferencia de Java en Python se instancio una sola método instance(), porque este lenguaje no permite y no es recomendado una sobrecarga de métodos.

**Template en Python:**

Este patrón, con respecto al código en java no hubo cambios.