**ESTRUCTURA CODIGO FUENTE NINJAS**

**Nombres:**

* **JUAN DAVID OLIVOS OCAMPO**
* **JUAN PABLO LEZAMA**
* **JONATAN RIOS**

***FactoryAbstractNaruto.py***

***Texto

El contenido generado por IA puede ser incorrecto.***

***Define los productos abstractos:***

* Jutsu: interfaz de jutsus (nombre, descripción, costo, ejecución).
* Ninja: interfaz de ninjas (uso de jutsus).

***Define los productos concretos:***

* CreateJutsu: implementación de un jutsu específico.
* BaseNinja: clase base para ninjas (con lista de jutsus).
* Subclases por aldea (KonohaNinja, SunaNinja, KiriNinja, IwaNinja, KumoNinja).

***Define la fábrica abstracta NinjaFactory y fábricas concretas (KonohaFactory, SunaFactory, etc.), que crean:***

* Un Ninja base (aldea específica).
* Un set de jutsus característicos de esa aldea.

***Rol en el sistema:***

Es la base que proporciona un ninja base con sus jutsus iniciales según la aldea elegida

***NinjaBuilder.py***

***Texto

El contenido generado por IA puede ser incorrecto.***

Implementa el patrón Builder para personalizar el ninja base entregado por la fábrica.

***Clases clave:***

* Stats: encapsula estadísticas del ninja (Ninjutsu, Taijutsu, Velocidad, etc.).
* FinalNinja: clase final que representa al ninja construido, con todos sus atributos (nombre, clan, rango, arma, estilo, aldea y jutsus).
* Builder (interfaz abstracta) → define qué pasos hay para personalizar un ninja.
* NinjaBuilder (concreto) → implementa los pasos (setName, setClan, setStats, etc.).
* Director: orquesta la creación → pide el ninja base a la fábrica, luego lo personaliza con el builder, y devuelve un FinalNinja.

***Rol en el sistema:***

Toma el ninja base de la Factory y lo convierte en un ninja final totalmente personalizado.

***MissionSystem.py***

***Texto

El contenido generado por IA puede ser incorrecto.***

Sistema encargado de manejar misiones y elegibilidad.

***Clases clave:***

* Rank: enum con rangos de ninjas (Genin, Chunin, Jonin).
* MissionRank: enum con rangos de dificultad de misiones (D, C, B, A, S).
* Mission: misión con id, título, recompensa y rango requerido. Verifica si un ninja puede aceptarla según su rank.
* MissionManager: gestiona la lista de misiones y asigna una a un FinalNinja si cumple requisitos.

***Rol en el sistema:***

Permite conectar al ninja final con el sistema de misiones, validando si puede aceptar o no una misión según su rango.

***Training.py***

***Texto

El contenido generado por IA puede ser incorrecto.***

Maneja el entrenamiento de ninjas y mejora de estadísticas.

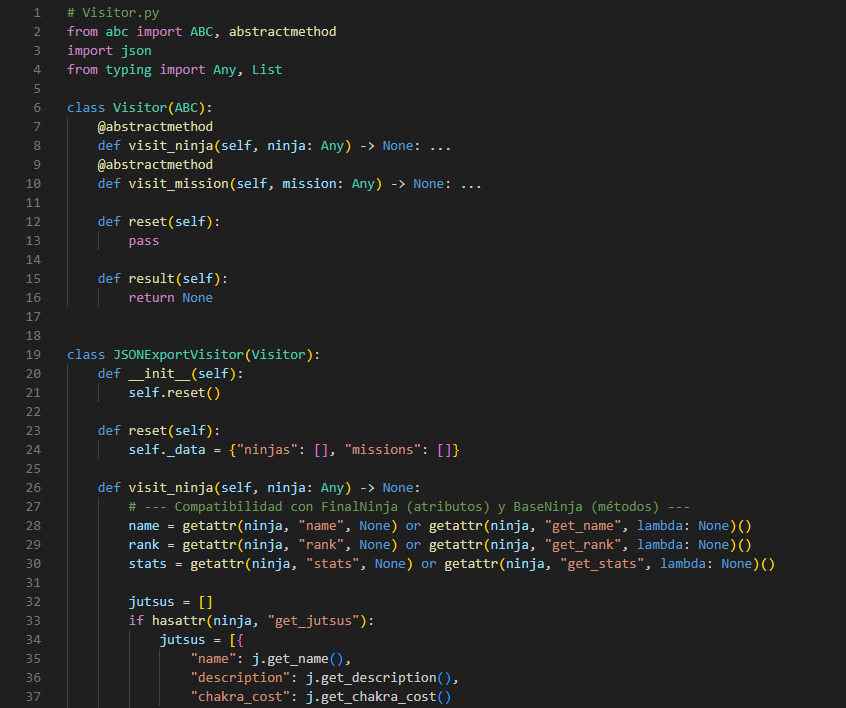
***Clase clave:***

* TrainingSystem con método train\_ninja(ninja, atributo, puntos) que aumenta el valor de una estadística del ninja.
* Verifica si el atributo existe antes de modificarlo.

***Rol en el sistema:***

Permite que el ninja final crezca y mejore sus estadísticas después de ser creado.

***Visitor.py***



**Define el Visitor abstracto:**

* **Visitor:** interfaz con los métodos visit\_ninja(ninja) y visit\_mission(mission) que permiten recorrer distintos objetos sin modificar sus clases.

**Define el Visitor concreto:**

* **JSONExportVisitor:** Recorre ninjas y misiones, Extrae su información (nombre, rango, estadísticas, jutsus, detalles de misión) y construye un diccionario JSON estructurado con todos los datos.

**Funciones auxiliares:**

* **export\_all(visitor, ninjas, missions):** Llama a accept(visitor) en cada ninja y misión y centraliza la recolección de datos y devuelve el JSON final.

**Rol en el sistema:**

* Permite separar la lógica de exportación (ej. generar un reporte en JSON) de las clases Ninja y Mission.
* Gracias al patrón Visitor, se puede añadir nuevas operaciones sobre ninjas y misiones (por ejemplo, generar reportes XML o estadísticas) sin modificar las clases base.

**Flujo completo del sistema**

1. **FactoryAbstract** crea un **ninja base** con aldea y jutsus.
2. **NinjaBuilder + Director** personalizan el ninja → se obtiene un FinalNinja.
3. **MissionSystem** permite asignarle misiones al ninja según su rango.
4. **TrainingSystem** permite entrenar y mejorar estadísticas del ninja final.
5. **Visitor (JSONExportVisitor)** recorre ninjas y misiones mediante accept(visitor), genera un reporte estructurado en JSON con toda la información consolidada del sistema.