MOMENTO EVALUATIVO #4

Equipo
Luz Natalia Ríos Serna

Julián Felipe Vélez Medina

Nilzon Alejandro Gómez Maya

Juan Pablo Salazar Ceballos

Propuesta de trabajo

INTERFAZ DE UTILIDADES PARA UN JUGADOR

CONTEXTO

Es común que en videojuegos tipo RPG el jugador cuente con acceso a distintos módulos los cuales le permiten ya sea desde el acceso a objetos recogidos a través de la sesión como habilidades o hechizos que le permiten completar desafíos de formas más creativas y especializadas.

Con el fin de optimizar la administración de estos módulos, se plantea el desarrollo de una interfaz de utilidades para un jugador genérico, basado en el uso de estructuras de datos tipo Listas Dobles, para simular su integración en otros proyectos.

OBJETIVO DEL SISTEMA

Diseñar e implementar un sistema que permita registrar, gestionar, buscar y eliminar objetos, hechizos o pociones aplicando la estructura de datos Lista Doble construida mediante clases y nodos enlazados, garantizando un flujo organizado y optimizado para la aplicación de mecánicas de juegos tipo RPG

ENTRADAS

• Archivos de texto con los objetos, hechizos y pociones iniciales, donde cada línea contiene:

tipo_objeto, nombre_objeto, efecto, duración_efecto, peso, usos

• Objetos, hechizos o pociones nuevas ingresadas manualmente por el jugador.

• Criterios de búsqueda o eliminación, ingresados por consola (por ejemplo: nombre del objeto o tipo del objeto).

SALIDAS

• Listado de objetos, hechizos o pociones organizados de mayor a menor peso (en consola).

• Archivo de texto con objetos que tengan 3 usos o menos.

• Resultados de operaciones de búsqueda, eliminación y ordenamiento.

COMO SE DISTRIBUYE NUESTRO PROYECTO:

1. Estructura clave:

- Double_node: Es el archivo donde se encuentra la codificación para la clase del nodo doble que usamos para guardar cada uno de los datos que se almacenarán en nuestra lista, este solo cuenta con getters y setters para los parámetros, data, next y prev, además de un método __str__ para traer la información almacenada en el data.
- Double_list: En este archivo esta todo el código de las listas dobles donde se almacenará cada uno de los nodos, en este tenemos getters y setters para atrapar y definir cada uno de los parámetros del método inicializador, además de los métodos de agregación al inicio, final, antes o después de algún nodo en especifico y también tenemos métodos para saber el tamaño, contenido y estado de la lista (con o sin información).
- Controlador: La clase Controlador actúa como el núcleo del sistema de utilidades del jugador, gestionando de forma organizada tres listas dobles independientes que almacenan los objetos, hechizos y pociones disponibles en el juego. Su función principal es administrar el inventario general, permitiendo cargar datos desde archivos, insertar nuevos elementos de manera controlada y ordenada, y mantener la coherencia con la clase Jugador, con la cual se vincula directamente para validar límites de peso y reflejar el estado del inventario. Esta clase encapsula toda la lógica de manipulación de datos sin recurrir a estructuras nativas de Python, garantizando un diseño orientado a objetos conforme a las directrices académicas del curso.
- Main: El archivo main.py actúa como el módulo principal del sistema, encargado de inicializar los componentes esenciales del proyecto y demostrar el funcionamiento integrado de las clases que conforman la interfaz de utilidades del jugador. En este archivo se crea una instancia del Controlador, se asocia un Jugador que gestiona su inventario a través de listas dobles y se cargan los datos iniciales desde un archivo externo. Además, permite ejecutar pruebas de inserción, visualización y validación de peso para los elementos del inventario (objetos, hechizos y pociones), sirviendo como entorno de ejecución y verificación práctica del correcto comportamiento del sistema.

2. Carpeta de Elementos:

En esta carpeta se encuentran todas las clases de los elementos usados en nuestro proyecto

- Hechizo: En este archivo se encuentra una programación básica donde se inicializa una clase Hechizo, la cual tiene como parámetros (nombre, efecto, duración, peso, usos), los cuales son usados para el método usar, el cual tiene la función de imprimir un mensaje que refleja los efectos causados por el elemento para luego efectuar un cambio minucioso en el parámetro de usos
- Jugador: La clase Jugador encapsula los datos y comportamientos asociados al personaje dentro del juego. Permite administrar su estado energético (maná) y controlar el peso total de los elementos que transporta (objetos, hechizos y pociones) mediante la comunicación directa con el controlador, el cual gestiona las tres listas dobles del inventario.
- Objeto: La clase Objeto representa los elementos físicos del inventario del jugador dentro del sistema RPG, almacenando información esencial como su nombre, efecto, duración,

peso y cantidad de usos disponibles. Cada instancia modela un objeto utilizable durante la partida, permitiendo invocar acciones a través del método usar y proporcionando una representación legible mediante el método __str__. La clase emplea encapsulamiento total mediante atributos privados y el uso de getters y setters, asegurando una gestión controlada de sus propiedades conforme a los principios de la programación orientada a objetos.

- La clase Poción extiende la funcionalidad de la clase Objeto, representando elementos consumibles dentro del inventario del jugador que otorgan efectos temporales al ser utilizados. Sobrescribe el método usar para simular el consumo de la poción, reduciendo su número de usos y su peso después de cada utilización. Esta clase mantiene la estructura y el encapsulamiento de su clase base, aplicando los principios de herencia para reutilizar y especializar el comportamiento de los objetos dentro del sistema de listas dobles del juego.

3. Carpeta recursos:
Esta carpeta esta destinada para almacenar todos los recursos que se necesitan para hacer uso de nuestro proyecto

- Iniciales.csv: En este archivo se encontrarán todos los “recursos” necesarios para empezar la ejecución de nuestro juego, pues aquí se encuentran todas las pociones, hechizos y objetos con los cuales se puede interactuar y con los cuales se podrá hacer uso del proyecto.

¿Cuál es el resultado esperado de nuestro proyecto?

El resultado esperado del proyecto es el desarrollo de un sistema funcional de administración de inventario para un jugador tipo RPG, implementado completamente mediante Listas Dobles como estructura de datos central, con un diseño orientado a objetos que garantice modularidad, encapsulamiento y reutilización del código. Este sistema debe simular el manejo de los recursos de un jugador como objetos, hechizos y pociones a través de una interfaz lógica y controlada que permita realizar operaciones de inserción, búsqueda, eliminación, ordenamiento y generación de reportes.

En términos funcionales, se espera que el programa sea capaz de leer información inicial desde archivos de texto, construyendo automáticamente tres listas dobles independientes que representen los distintos tipos de elementos disponibles en el inventario. Cada lista, implementada mediante nodos enlazados hacia adelante y hacia atrás, permite recorrer la información de manera eficiente en ambos sentidos y realizar operaciones de inserción o eliminación sin necesidad de recurrir a estructuras nativas de Python. Gracias a este diseño, el proyecto demuestra el dominio técnico sobre la manipulación de estructuras dinámicas, cumpliendo con uno de los principales objetivos del componente práctico del curso.

El controlador juega un papel fundamental como eje de coordinación del sistema. Esta clase actúa como un inventario general, encargándose de gestionar las tres listas dobles (objetos, hechizos y pociones), manteniendo la coherencia entre ellas y centralizando todas las operaciones que

afectan al inventario. Desde el controlador se administran procesos como la carga de datos, la inserción ordenada de elementos (por ejemplo, en función del peso de los objetos), la verificación de límites de peso mediante la interacción con la clase Jugador, y la exportación de resultados hacia archivos externos. De esta manera, el controlador garantiza un flujo de datos organizado, estableciendo una comunicación bidireccional con el jugador y asegurando que todas las modificaciones del inventario sean reflejadas correctamente en su estado general.

A su vez, el jugador actúa como el núcleo lógico del sistema, enlazado directamente con el controlador para acceder a las listas y gestionar sus propios límites de recursos (como maná y capacidad de carga). El jugador puede consultar el contenido total de su inventario, visualizar el peso acumulado de los elementos almacenados y validar si es posible agregar nuevos objetos sin exceder su límite máximo. Esta integración entre el jugador y el controlador constituye una simulación coherente y realista del manejo de recursos en un videojuego de rol.

En conjunto, el resultado esperado es una aplicación modular, totalmente funcional y académicamente sólida, que demuestre la aplicación correcta de las Listas Dobles como estructura base de almacenamiento y manipulación de datos, reforzando los principios de la Programación Orientada a Objetos (POO). El sistema debe evidenciar la capacidad de los estudiantes para diseñar clases interrelacionadas, aplicar herencia, mantener la integridad de los datos y desarrollar soluciones reutilizables y escalables dentro del ámbito de la gestión de inventarios en entornos interactivos.