DOCUMENTACIÓN: Situación problema

Marcela Beatriz De La Rosa Barrios A01637239
Victor Javier Quintana Cisneros A01643020

13/06/2023

Programación orientada a objetos (Gpo 315)

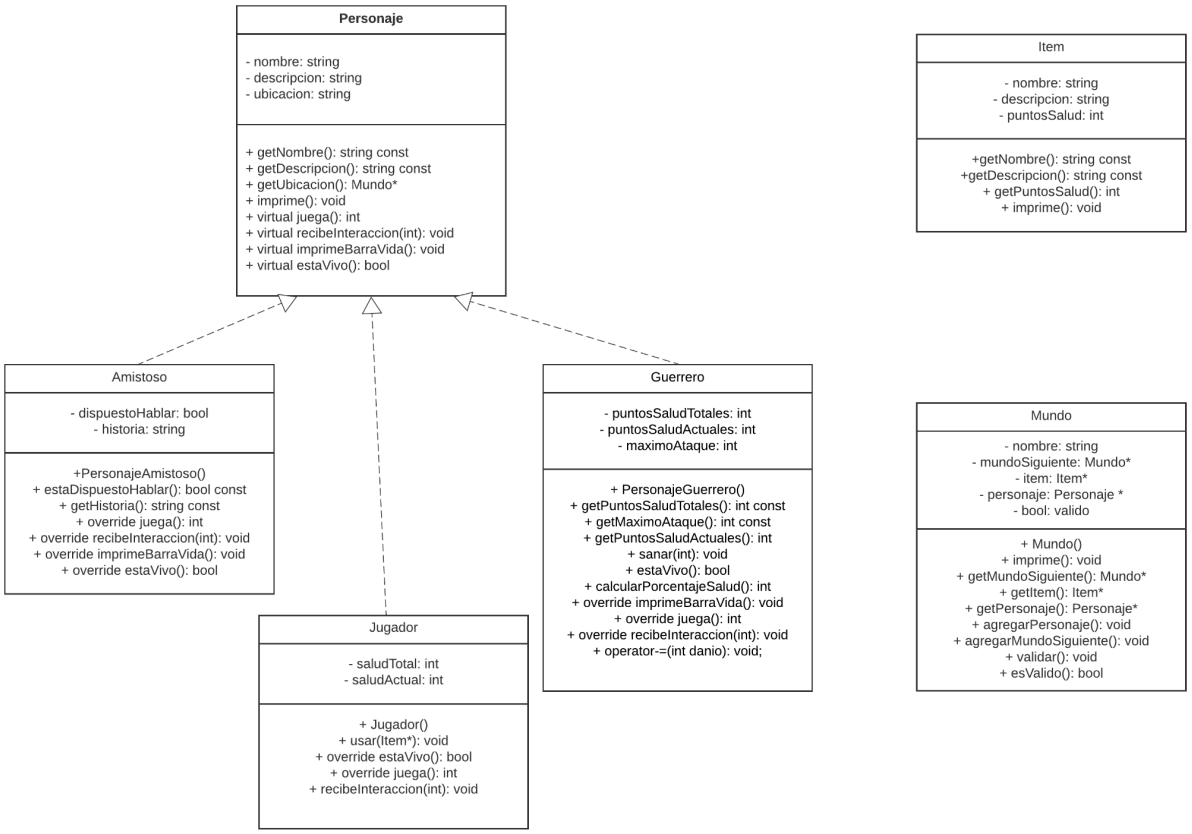
Situación problema: documentación

Introducción

La siguiente situación problema se basa en el lenguaje de C++ con el uso de la programación orientada a objetos para la implementación de la misma dirigida a un videojuego en consola que tiene como objetivo principal la aplicación y demostración de los conocimientos adquiridos en el curso, abordando de manera práctica diversos temas fundamentales, como la herencia, el polimorfismo, las clases abstractas, la sobrecarga de operadores y el manejo de excepciones.

En esta propuesta de juego a consola textual, se enfatiza la utilización correcta de los conceptos mencionados, a través de la creación de un entorno interactivo compuesto por mundos, personajes e items. El jugador, que es parte de la clase guerrero, asumirá el rol protagónico, enfrentándose a personajes guerreros a los cuales tiene que atacar para vencer y se encuentra con personajes amistosos que interactúan contando una historia y otorgando un ítem con puntos de vida al protagonista. La estructura del juego se basa en una secuencia lineal de mundos, los cuales el jugador deberá pasar venciendo o interactuando con personajes.

Diagrama de clases UML:



Explicación de dónde se usaron los conceptos vistos en clase

La **herencia** se utiliza dentro del programa al usarse de la siguiente manera:

- En la clase “PersonajeGuerrero”, que hereda de la clase base “Personaje”.
- En la clase “PersonajeAmistoso”, que también hereda de la clase base “Personaje”.
- En la clase “Jugador”, que hereda de la clase “PersonajeGuerrero”.

La razón por la que se utilizó la herencia en el proyecto es porque permite que las clases derivadas puedan usar las propiedades y métodos de la clase base. En este caso, las clases derivadas obtienen las características y comportamientos definidos en la clase base “Personaje” como por ejemplo los atributos: nombre, descripción y ubicación.

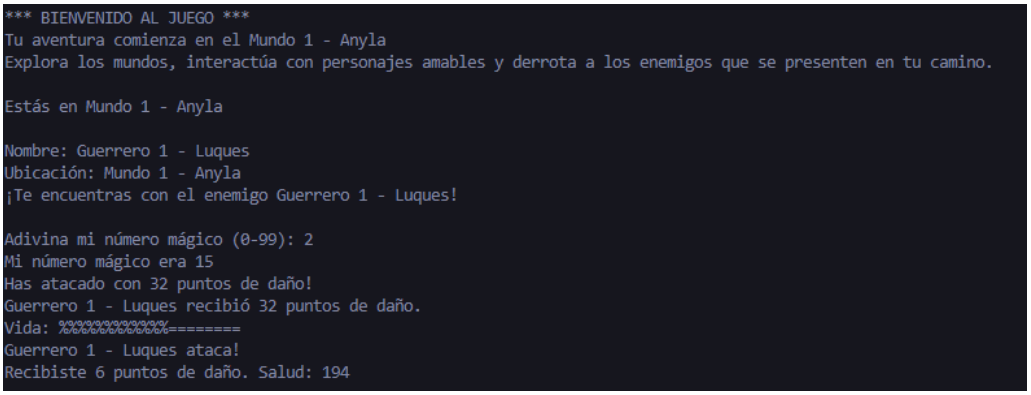
El **polimorfismo** se utiliza en varias partes del código a través de las funciones virtuales y puras que se utilizaron en la clase base “Personaje” y que se implementan de forma diferente en cada clase derivada. Los métodos: juega(), recibeInteraccion(), imprimeBarraVida() y estaVivo() en la clase “Personaje” son ejemplos de funciones virtuales puras, lo que significa que no se implementan en la clase base y se usan en las clases derivadas a ella. Esto se hizo con la intención de ahorrar código y repeticiones innecesarias para que las clases derivadas se comporten de manera específica a través del uso de estos métodos.

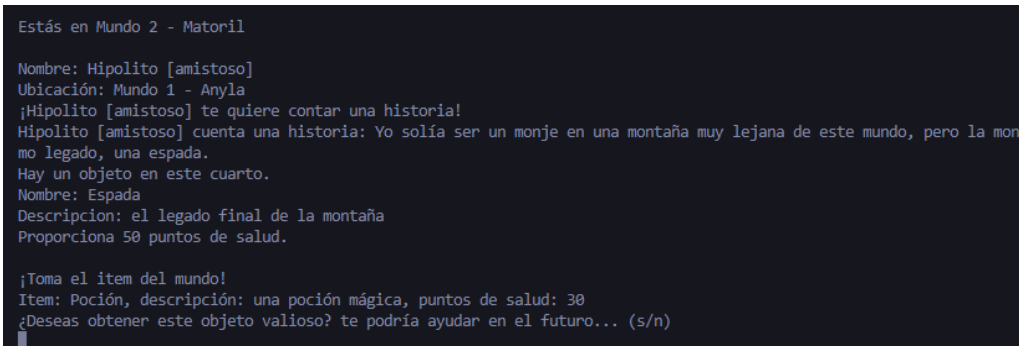
La clase “Personaje” es una **clase abstracta** debido a la presencia de las funciones virtuales puras descritas dentro de ella, las cuales no se instancian dentro de la clase base pero sí se pueden utilizar como un tipo de plantilla para las otras clases que heredan de “Personaje”.

En la clase PersonajeGuerrero se utiliza la **sobrecarga de operadores**: el operador **-=** (asignación y sustracción) se sobreescribe para aplicarle daño a un personaje. Se utiliza en el juego para infligir daño tanto en el personaje como el contrincante que está peleando.

Se utilizó el **manejo de excepciones** para capturar errores que puedan ocurrir al intentar imprimir el personaje del mundo actual. Esto se realizó de tal forma que, al obtener el puntero al personaje del mundo actual con getPersonaje(), se utiliza un bloque `try-catch` para llamar “imprime()” del objeto “personaje”. Dentro del bloque “try”, se intenta llamar esta función para que, si ocurre alguna excepción del tipo `std::runtime_error` durante la ejecución de la función `imprime()`, el flujo de control se transferirá al bloque `catch`. El cual captura la excepción en una variable `e` del tipo `std::runtime_error` para luego, mostrar un mensaje de error indicando que hubo un problema al obtener el personaje del mundo actual, que se imprime utilizando la

función `e.what()`. Esto se realizó para que, si se produce algún error al imprimir el personaje del mundo actual, se captura la excepción, se muestra un mensaje de error y actualiza "mundoActual", lo que evita la interrupción del programa.





Conclusión personal:

Marcela Beatriz de la Rosa Barrios :

Durante la creación del videojuego a consola he aprendido conceptos fundamentales en la programación cómo la herencia, el polimorfismo, la sobrecarga de operadores y el manejo de excepciones en C++, y los he aplicado en este proyecto junto con mi compañero de equipo, Victor. El desarrollo de este proyecto me ha permitido comprender la importancia de los temas y las bases de la programación vistos en el curso por la practicidad que demostró durante la realización del código. En cuanto a estos temas, considero que el que más trabajo me ha costado entender dentro de la teoría y la práctica ha sido el de apuntadores, aunque esto no significa que no quiera aprender, por lo que concluyo el curso con bases fuertes para los temas y algoritmos que me quedan por aprender.

Victor Javier Quintana Cisneros:

C++ es un lenguaje que se me dificulta un poco, al programar usualmente en alto nivel,

tratar con punteros y referencias se me dificulta. Sin embargo, este proyecto me permitió poner a práctica estos conceptos, junto con otros de OOP en general, como el polimorfismo, herencia y clases abstractas. Además, aprendí sobre la sobrecarga de operadores en el lenguaje, que es algo que nunca había hecho. Termino esta clase con nuevos conocimientos y mejor seguridad en C++, sin embargo, me es claro que debo practicar más en este lenguaje y en lenguajes de bajo nivel en general.