Trabajo Práctico 1

I102 – Paradigmas de Programación

Autor: Bautista García

Universidad: Universidad de San Andrés (UdeSA)

Año: 2025

Introducción

Este informe tiene como objetivo documentar la resolución del Trabajo Práctico 1 de la materia Paradigmas de Programación. Todos los programas fueron compilados con los flags `-Wall` y `-Wextra` sin generar ningún warning.

Compilación

Para compilar cada ejercicio es necesario tener instalado `make` y `g++`. Cada directorio (`ej1`, `ej2`, `ej3`) contiene su propio archivo `makefile` y su correspondiente `main.cpp`.

Para compilar y ejecutar un ejercicio, simplemente hay que posicionarse en la carpeta deseada mediante la terminal y ejecutar el siguiente comando:

```
make
```

Esto generará el ejecutable correspondiente y lo ejecutará. Una vez terminado el programa, se borran todos los ejecutables generados automáticamente por el makefile. A continuación, se detallan las soluciones implementadas para cada uno de los ejercicios planteados.

1. Ejercicio 1 - Diseño de Personajes y Armas

Para comenzar con la implementación, se definieron tres componentes fundamentales: **Armas**, **Personajes** y **Efectos**. Las primeras dos surgen directamente de la consigna, mientras que **Efecto** fue una clase adicional creada con el fin de extender el comportamiento de las armas, permitiendo efectos especiales como curar vida, provocar sangrado o aturdir al oponente.

Clases principales

- **IArma**: interfaz base de todas las armas.
- **ArmaMagica** y **ArmaCombate**: clases abstractas que derivan de IArma.
- **IPersonaje**: interfaz base de todos los personajes.
- **Mago** y **Guerrero**: clases abstractas que derivan de IPersonaje.
- **IEfecto**: interfaz base para definir efectos especiales que puede tener un arma.

Clases derivadas

A continuación se listan las clases derivadas de cada uno de estos componentes:

- **Armas Mágicas**: Bastón, LibroDeHechizos, Poción, Amuleto
- **Armas de Combate**: Hacha, HachaDoble, Espada, Lanza, Garrote
- **Magos**: Hechicero, Conjurador, Brujo, Nigromante
- **Guerreros**: Bárbaro, Paladín, Caballero, Mercenario, Gladiador
- **Efectos**: Sangrado, Stun, RoboDeVida, BoostMagico, CurarVida, Nada

Diseño de clases

En la Figura 1 se presenta el diagrama UML que muestra la relación entre estas clases, sus atributos y métodos, y cómo se vinculan entre sí utilizando composición y herencia. Este diseño permite extender el sistema fácilmente agregando nuevas clases de armas, personajes o efectos.

Mecánicas y restricciones

Los personajes de tipo **Mago** poseen una cantidad de **mana** que utilizan para emplear armas mágicas. Cada arma mágica requiere un costo específico de mana para poder ser utilizada. En contraposición, los personajes de tipo **Guerrero** no cuentan con puntos de mana, por lo cual no están habilitados para utilizar armas mágicas.

Por otro lado, las armas de combate cuentan con un número limitado de usos. Cada vez que son utilizadas, sus usos restantes disminuyen y, eventualmente, pueden romperse si se agotan.

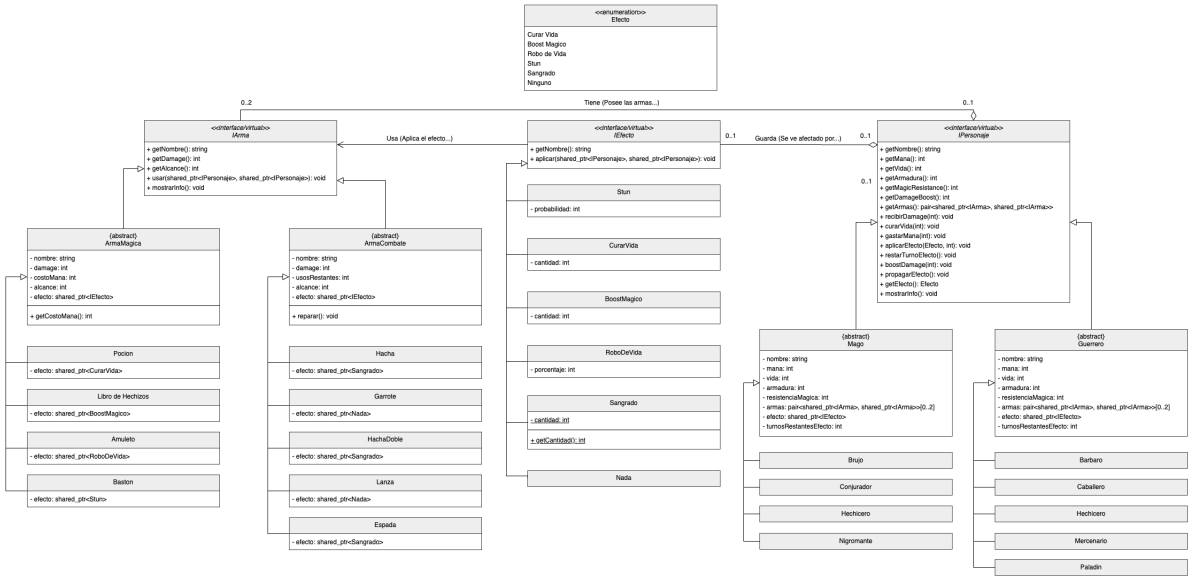

Figura 1: Diagrama UML del sistema de personajes, armas y efectos

Todas las armas del sistema pueden aplicar un **efecto especial** sobre el objetivo, que se pueden ver en la Figura 1. Cada arma tiene un efecto único. Por ejemplo, el Bastón aplica el efecto *Stun*, el cual inhabilita al personaje enemigo por los próximos dos turnos, impidiéndole ejecutar cualquier acción ofensiva o defensiva.

Listado de efectos y funcionamiento

- **Stun**: Inmoviliza al objetivo por 2 turnos con una probabilidad del 20 % de que sea efectivo.
- **CurarVida**: Cura 100 puntos de vida al usuario.
- **BoostMagico**: Causa un incremento permanente del daño mágico del 25 %.
- **RoboDeVida**: Roba el 15 % de la vida restante del objetivo y se transfiere al atacante.
- **Sangrado**: Provoca un daño adicional de 30 puntos planos durante 3 turnos consecutivos.
- **Nada**: Este efecto no tiene ningún impacto y no produce ningún cambio.

2. Ejercicio 2 - Generación aleatoria y factoría de personajes

Para resolver este ejercicio, se implementó una clase denominada `Factory`, que cuenta con tres métodos estáticos: `CrearArma`, `CrearPersonaje` y `CrearPersonajeArmado`. Estos métodos permiten la creación dinámica en tiempo de ejecución de instancias de personajes y armas, haciendo uso de los tipos definidos previamente en el Ejercicio 1.

Se utilizan punteros inteligentes `shared_ptr` para manejar la memoria de forma segura. Cada método recibe parámetros que indican el tipo de objeto a crear, definidos mediante enumeraciones.

Como prueba de funcionamiento, hay un archivo main que genera una cantidad aleatoria `N` de personajes magos y guerreros, utilizando la función `rand()`:

- Se generan entre 3 y 7 personajes magos (tipos al azar).
- Se generan entre 3 y 7 personajes guerreros (tipos al azar).
- A cada personaje se le asigna aleatoriamente entre 0 y 2 armas.
- Tanto las armas como los personajes se seleccionan usando enumeraciones.

Durante todo el proceso se controlan posibles errores de `bad_alloc` al fallar la asignación de memoria.

3. Ejercicio 3 - Simulación de batalla

En este ejercicio se simula una batalla entre dos personajes creados a partir de la clase `Factory` del ejercicio anterior. Un personaje es elegido por el usuario y el otro se genera aleatoriamente, junto con sus acciones durante la batalla.

El combate sigue una mecánica de tipo piedra-papel-tijera con las siguientes opciones:

- (1) Golpe Fuerte
- (2) Golpe Rápido
- (3) Defensa y Golpe

Las reglas son simples: Golpe Fuerte vence a Golpe Rápido, Golpe Rápido vence a Defensa y Golpe, y Defensa y Golpe bloquea el Golpe Fuerte causando daño al atacante.

La batalla se muestra en la terminal con una interfaz clara y estructurada. Un ejemplo del output es el siguiente:

```
==================== ¡Batalla! ====================
Jugador 1:
Barbaro con Hacha
[######### ] 90/100

Jugador 2 (CPU):
Mercenario con Espada
[##########] 100/100
================================================

==================== Resultado ====================
Barbaro ataca con Hacha y hace 10 puntos de daño.
================================================

==================== Turno {N} ====================
Su opción: (1) Golpe Fuerte, (2) Golpe Rápido, (3) Defensa y Golpe:
```

Se usan valores externos a los pertenecientes a cada clase para una pelea más "justa", cada personaje cuenta con 100 puntos de vida y hace o no 10 puntos de daño como mucho.

Toda la creación de personajes y armas, así como la ejecución del combate, contempla el uso de memoria dinámica. Se controlan posibles errores de `bad_alloc` en caso de que falle la asignación.