Resolución de ejercicios

Tp1 paradigmas de la programación.

Door Lorenzo.
Universidad De San Andrés .

Compilación:

Para la compilación del código cada carpeta de cada ejercicio 2, 3 y 4 (ahora 1, 2 y 3) tiene su respectivo Makefile por ende deberá tenerlo instalado al igual que g + +. Este Makefile compila cada archivo necesario para la funcionalidad del código y prueba con c + + 11 y 17 poniendo en terminal make. Además, utilizo -Wall -Wextra , que no generaron ningún error, ni ningún Warning. Por lo que, los códigos se compilan y están en funcionamiento. Cada ejercicio tiene su propio main.cpp donde se ejecuta lo pedido, o en el caso del ejercicio 2 se hacen pruebas (tests) para mostrar la funcionalidad del mismo ya que por consigna no se pide realizar ningún programa con funcionalidad.

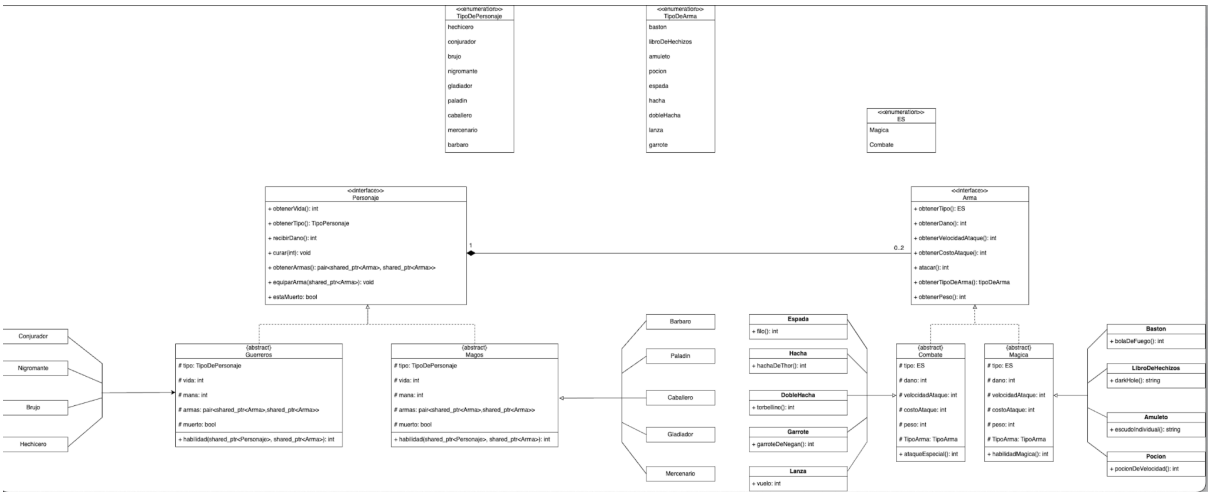
Ejercicio 2: (ahora 1)

2.1) para empezar con la implementación de las armas pide crear una interfaz en mi código llamada Arma de la que derivan dos clases abstractas la de armas de combate en mi caso denominada Combate y la de items mágicos, nombrada Mágica.
la interfaz tiene todos los métodos que van a poder realizar esta dos clases de armas, mientras que estas dos subclases definen la funcionalidad de estos métodos que serán utilizados por sus derivadas finales, además de los atributos que estas tendrán en común, creándose así el constructor de estas subclases de armas, también definirán (en mi caso) nuevas funciones virtuales puras (porque son abstractas) que serán definidas por sus derivadas dándoles o no funcionalidades diferentes.
En cuanto a las derivadas finales:
Las de combate se definen en: Espada, Garrote, HachaDoble, HachaSimple y Lanza.
las mágicas en: Bastón, Amuleto, LibroDeHechizos y Poción.
Cada una recibe un método de su clase abstracta que modifica la funcionalidad (habilidad especial) y otra que es creada dentro (sería como una segunda habilidad) que las diferencia entre sí, por último se establecen los valores en el constructor para así poder crear el arma.

2.2) para la implementación de este ítem, sigue la misma lógica que el 2.1 solo que con los personajes por lo que solo mencionare las clases y sus respectivas derivadas.

Interfaz: Personaje
Abstractas: Magos, Guerreros
Derivadas finales de los Magos: Hechicero, Conjurador, Brujo y Nigromante.
Derivadas finales de los guerreros: bárbaro, Paladín, Caballero, Mercenario y Gladiador.

2.3) En este punto se muestra un diagrama UML para entender cómo funciona este código y como están implementadas. Mostrando que deriva de que, qué funciones se utilizan, qué atributos poseen y qué métodos.



Ejercicio 3: (ahora 2)

3.1 y 3.2) para estos dos subitems del ejercicio lo que realizó es una función que me genere dos numero aleatorios utilizando la función rand() entre 3 y 7, dependiendo de este dos número se crearán dos vectores de largo el numero que salio, este número (largo del vector) indica la cantidad de magos y guerreros que van a crearse es decir si un vector es de largo 6 y otro de largo 4 se van a crear 6 magos y 4 guerreros. después de esto se crean por la cantidad del largo del vector un número aleatorio entre 0 y 2 que indica la cantidad de armas que va a poseer el personaje. ejemplo de como quedaría.

magos: [0,1,0,2] se crean 4 magos (largo vector) y el primero tendrá 0 armas, el segundo 1 y así como indique el número. lo mismo con el vector de guerreros.

3.3) En este punto lo que se hace es crear una clase PersonajeFactory, que tiene los métodos estáticos que crean dinámicamente en tiempo de ejecución los personajes y armas del ejercicio 2, además como pide la consigna de utilizan smart pointer en mi caso shared_ptr.
estos métodos son: crear Arma, crear personaje y crear personaje armado
explicación del funcionamiento del main: lo que hago es primero crear los vectores como en el 3.1 y 3.2) que me indicará lo ya explicado en esos ítems.
luego en base a esos resultados, se crearán los personajes de forma aleatoria con unas funciones que lo permiten y dependiendo el valor en esa posición del vector también aleatoriamente se le creara un arma, dos armas o ninguna y se le asignará.
Entonces, por ejemplo: se crean 4 magos y 3 guerreros, [1,1,0,2] y [0,0,2]
mago 1: “X(puede ser cualquier mago de las derivadas finales del ejercicio 1)”, arma 1 = “Y(arma aleatoria entre cualquiera de todas, porque cualquier personaje puede usar cualquier arma no hace falta que los magos usan mágicas y los guerreros de combate)”, arma 2 = “sin arma” (en este caso porque en el mago 1 salio un 1 es decir tiene un solo arma) y asi se hace una lista mostrando los magos sus armas (si tienen) y los guerreros con sus armas (si tienen).

ejercicio 4: (ahora 3)

Este ejercicio lo voy a explicar sin subitems ya que no lo veo necesario.
En el punto final del TP, debe simular una batalla entre estos personajes armados creados por el ejercicio 2.
Este juego lo que hace es tener dos personajes únicamente con un arma que pelean en un estilo de piedra papel o tijera hasta que un usuario le gane 10 veces al otro.
primero se crea el personaje que representara al usuario, éste será elegido por el mismo y también será elegida el arma que portará y a su vez se creará aleatoriamente el rival, será un personaje entre los 9 posibles y el arma que usará también aleatoria entre las 9 que existen. Aclaración: aunque en mi implementación mis armas tengan diferente daño, al no parecerme balanceado (para que sea una pelea justa) en este estilo de batalla, seguí la consigna y todas las armas realizan 10 de daño.
una vez creados los dos personajes con sus armas inicia la batalla donde se elige entre 3 opciones (la opción del rival siempre es aleatoria) (1) Golpe Fuerte, (2) Golpe Rápido, (3)

Defensa y Golpe. El “Golpe Fuerte” le gana al “Golpe Rápido” “Golpe Rápido”. El “Golpe Rápido” le gana a la “Defensa y Golpe” “Defensa y Golpe” bloquea el “Golpe Fuerte” . En caso de que los dos personajes realicen la misma acción, ningún personaje recibirá daño y se pasa a la siguiente ronda de elección (10 de daño se le realiza al personaje rival al ganar el “piedra papel o tijeras”).

¿por qué anteriormente dije 10 rondas? Porque el jugador que pierde los 100 de vida que posee pierde y como le resta de a 10, el ganador debería ganar 10 rondas.

Se anuncia el ganador y termina la batalla.