Universidad del Valle de Guatemala
CC2008 - Introducción a la Programación Orientada a Objetos
Semestre II, 2021

Ejercicio 4: modelación con herencia

Usted es encargad@ de desarrollar el sistema de batalla para un sencillo juego de rol, en el que se enfrentan uno o más jugadores a uno o más enemigos en combate por turnos. Se le ha requerido presentar un prototipo incluyendo, al menos, dos tipos de enemigo diferentes, cada uno con una versión normal y una versión "jefe". Además, debe incluir un breve simulador de batallas. Las batallas se desarrollan por turnos. Tome en cuenta lo siguiente:

- Tanto el jugador o la jugadora como los enemigos son combatientes en una batalla.
    - L@s combatientes tienen nombre, puntos de vida y poder de ataque.
    - Tod@ combatiente puede:
        - Desplegar un mensaje: que diga algo al iniciar batalla y al morir o ganar.
        - Tomar un turno: durante una batalla, cada combatiente deberá tomar un turno. En su turno, el/la combatiente puede atacar, pasar su turno o hacer uso de un *ítem* o habilidad especial (ver detalles más abajo).
            - Algunas de estas opciones deberán proveerle al usuario la posibilidad de elegir un(a) combatiente objetivo.
            - Atacar es reducir los puntos de vida de un(a) combatiente objetivo.

- Cada tipo de enemigo debe tener una habilidad especial distintiva que afecta a un(a) combatiente (como curar, esquivar, envenenar, etc.; use su imaginación).

- Un enemigo en versión "jefe" debe poseer una habilidad especial adicional a la que incluye su versión normal (también aplica a un(a) combatiente). Además, los jefes son más poderosos que la versión normal (el ataque es más fuerte, las habilidades especiales hacen mayor efecto, tienen más vida, los ataques les hacen menos daño, etc.).

  La habilidad especial del jugador es usar *ítems*. El jugador debe poseer y poder usar *ítems* sobre un@ o más combatientes. Los *ítems* pueden recuperar una cantidad fija de vida o incrementar el poder de ataque por un turno; nuevamente, use su imaginación. Los *ítems* del jugador o jugadora son como las habilidades especiales de los enemigos, pero tienen cantidad limitada y podrían afectar a más de un combatiente. Provea a su jugador de los *ítems* que considere necesarios, tomando en cuenta que se consumen con el uso.

  No es necesario que el diseño restrinja los *ítems* a los tipos mencionados, ni las habilidades especiales. Usted puede inventar nuevos *ítems* y habilidades, si lo desea.

- Un jugador puede escoger un rol entre dos:
    - Guerrer@: bastante vida, bastante ataque, poca capacidad para *ítems*.
    - Explorador(a): vida normal, ataque normal, amplia variedad de *ítems* (el doble o más que lo que el guerrero).

- Cada batalla debe incluir entre uno y tres enemigos de cualquier tipo. La cantidad es elegida al azar.
- El simulador concluye la batalla cuando el jugador o todos los enemigos quedan sin puntos de vida (o si el jugador elige salir).
- Durante la batalla, el usuario del programa controla las acciones del jugador y de los enemigos, mediante menús.
- El programa ejecuta el simulador, con cada iteración (turno) mostrando el *status* de tod@s l@s combatientes y un registro de las últimas tres acciones tomadas (entre tod@s), además de permitiendo a l@s combatientes tomar sus turnos cuando les toque.
- Las propiedades del jugador y los enemigos pueden ser predeterminadas en el código o solicitadas al usuario, como lo desee usted.

Instrucciones

Desarrolle un diagrama UML de su diseño. Asegúrese de aprovechar el uso de herencia y *subtyping* (polimorfismo por inclusión o vía herencia). No olvide emplear *overriding* y/u *overloading* donde lo necesite, ni de seguir los principios de diseño vistos en clase incluyendo la aplicación del patrón MVC. Implemente el prototipo incluyendo todas las funcionalidades requeridas. Su programa debe ser fiel a su diseño.

Nota: sea creativ@. Aquello que no esté explícitamente determinado en las instrucciones puede ser determinado por usted, en su diseño. No tema hacer más uso de herencia que el obvio. Para ejemplos de habilidades especiales, ataques, *ítems* y del sistema de batalla en general, refiérase a las siguientes fuentes (principalmente al contenido visual, no tanto a las explicaciones):

- https://www.youtube.com/watch?v=G1X0PhGD73E&ab_channel=IanTalksAboutStuff
- https://www.youtube.com/watch?v=kSvqhQHDGyk&ab_channel=JinKisaragi

Debe entregar en Canvas:

- **[50 pts.] Parte 1 -** Diagrama de clases.
  - **[10 pts.]** Sintaxis y correcto uso de relaciones y modificadores de visibilidad.
  - **[12.5 pts.]** Correcto uso de herencia en las relaciones entre clases.
  - **[12.5 pts.]** Correcto uso de polimorfismo vía herencia en la especificación de parámetros para comportamientos de los componentes y las especificaciones de *overriding*.
  - **[10 pts.]** Correcta separación de responsabilidades y aplicación de MVC y los principios de diseño para el cumplimiento de los requisitos funcionales.
  - **[05 pts.]** Buenas prácticas de programación:
    - *Override* de `toString` (e `equals`, donde sirva).
    - Encapsulación mediante modificadores de visibilidad y *getters/setters*.
- **[50 pts.] Parte 2 -** Programa que implemente su diseño.
  - **[15 pts.]** Correcto uso de *subtyping*.
  - **[20 pts.]** Cumplimiento de requisitos funcionales.
  - **[10 pts.]** Usabilidad; interfaz amigable con el o los usuarios.
  - **[05 pts.]** Comentarios y encabezados.

Recuerde: haga PRIMERO su diseño. En caso identifique, durante la implementación, necesidad de cambios, asegúrese de detener la programación y rediseñar con una revisión general del impacto de los cambios a realizarse. Luego de anotar los cambios en el diseño aplicarlos en el código, siempre con comentarios explicativos en todos lados. Finalmente **incluir el rediseño al entregar el código**. Cambios menores pueden pasar por alto, pero si su programa difiere demasiado de su diseño original puede perder puntos. Diseñe con meticulosidad.