INF-253 Lenguajes de Programación
Tarea 3: Java

Profesor: Esteban Daines - Roberto Diaz
Ayudante Cátedras: Sebastian Godinez
Ayudante Tareas: Gabriel Valenzuela - Monserrat Figueroa

22 de Octubre de 2018

1. Dunyons & Doragons

Tras años de un extenso análisis y diversas investigaciones, los desarrolladores de los más populares juegos de rol han decidido lanzar una versión virtual que recopile las mecánicas de sus clásicos más exitosos: **Dunyons & Doragons**. Para comenzar con su desarrollo, la empresa a cargo le ha solicitado a los alumnos de Lenguajes de Programación que desarrollen un prototipo funcional del juego en lenguaje Java mediante el uso de **listas y la orientación a objetos**, entre otros aspectos del lenguaje. Este prototipo buscará que los jugadores aprendan sobre la creación de personajes y las batallas que éstos pueden tener.

2. Reglas del Juego

- El juego consiste en una variación de la creación de personajes en juegos de rol y sus mecánicas de pelea.
- Para una mayor comprensión, se utilizará la siguiente nomenclatura para explicar el uso de dados:
  - Dado de 20 caras o d20: se obtiene un número al azar que varíe entre el 1 y el 20.
  - Dado de 8 caras o d8: se obtiene un número al azar que varíe entre el 1 y el 8.
  - Dado de 6 caras o d6: se obtiene un número al azar que varíe entre el 1 y el 6.
- Existen 4 tipos de razas, las cuales poseen estadísticas (modificadores) por defecto y habilidades pasivas cuyo uso se explicará más adelante. Las razas se resumen en la siguiente tabla:

Raza / Mod.	Fuerza	Destreza	Constitución	Habilidad
Humano	+1	+1	+1	Suerte
Elfo	-	+2	+1	Evasión
Enano	+1	-	+2	Resistencia
Orco	+2	-	+1	Atacante

Las habilidades se detallan a continuación:

- **Suerte**: no se puede obtener un 1 al lanzar un dado d20 (en cualquier circunstancia). Si se obtiene un 1, se debe volver a tirar el dado.
- **Evasión**: el personaje se debe sumar un valor de +2 al d20 para evadir o resistir un ataque mágico.
- **Resistencia**: cada vez que el personaje comience su turno, se curará 1 de vida.
- **Atacante**: el personaje debe sumar un valor de +2 al d20 para efectuar un ataque físico.

- Existen 4 tipos de clases, las cuales definen la forma en que el personaje realiza sus ataques y sus defensas. Una clase sólo puede realizar un tipo de ataque, que puede ser físico o mágico. Los ataques físicos pueden ser de fuerza o de destreza, los que se realizan directamente sobre el contrincante con posibilidad de fallar o acertar. Por otro lado, los ataques mágicos siempre acertarán sobre el contrincante, con la diferencia de que estos podrán evadir o resistir parte del daño haciendo uso de la destreza o la constitución respectivamente (esto se explicará con mayor precisión más adelante). Las clases se resumen a continuación:

Clase / Mod.	Ataque Físico	Ataque Mágico	Armadura
Bárbaro	Fuerza	-	15
Pícaro	Destreza	-	10
Mago	-	Destreza	10
Clérigo	-	Constitución	15

- Al comenzar el juego, el jugador deberá seleccionar una raza y una clase para concluir otorgándole un nombre.
- El personaje creado deberá enfrentarse a 3 contrincantes en enfrentamientos 1 vs 1. Al derrotar a todos, el personaje gana el juego.
- La computadora seleccionará 3 contrincantes para el enfrentamiento desde el listado de personajes predefinidos (los cuales pueden repetirse). Estos son:
  - Klrak, el enano bárbaro.
  - Adran, el elfo pícaro.
  - Isaac, el humano clérigo.

    - Elysium, el elfo mago.
    - Krrogh, el orco bárbaro.
    - Jenkins, el humano mago.
- El personaje no recupera su salud entre peleas.
- La vida del jugador se define como 10 + Constitución, mientras que la de los contrincantes se define como 8 + Constitución.
- La pelea se efectúa por turnos, en donde sólo se puede efectuar un ataque o entrar en defensa.
- Para atacar con un ataque físico se debe lanzar un d20; si este número es mayor o igual a la armadura del enemigo, el ataque es efectuado. Para calcular el daño de un ataque correctamente efectuado se debe lanzar un d8, valor al cual se le debe sumar el modificador de la estadística asociada al ataque físico del atacante (presentado en el **cuadro 2**). A continuación se presenta un ejemplo:
    - El jugador como un **humano bárbaro** con 5 de vida intenta atacar a Klrak con 12 de vida.
    - Se lanza un d20 y sale 7, por lo que el ataque falla ($7 < 15$).
    - Es el turno de Klrak, quien obtiene un 16 en su d20.
    - Como el ataque fue exitoso ($16 > 15$), Klrak lanza un d8 y obtiene un 4.
    - El daño final es de 5 (4 del dado + 1 de la fuerza de Klrak).
    - La nueva vida del jugador es de 0 y pierde el juego.
- Para atacar con un ataque mágico, el personaje que recibe el daño debe lanzar un d20 para intentar evadirlo o resistirlo, valor al cual se le debe sumar su modificador de la estadística asociada al ataque mágico del atacante (presentado en el **cuadro 2**); si este número es mayor o igual a 13, el daño se reduce a la mitad (redondeado hacia abajo). En caso contrario, el daño se recibe completamente. Para calcular el daño se debe lanzar un d6. A continuación se presenta un ejemplo:
    - El jugador como un **elfo mago** con 7 de vida intenta atacar a Isaac con 12 de vida.
    - Isaac lanza un d20 y sale 7. El valor final es de 8 (7 + 1 de la destreza de Isaac).
    - Dado que el valor es menor al valor requerido ($8 < 15$), Isaac recibe todo el daño.
    - El jugador lanza un d6 y obtiene un 4.
    - El daño es de 4, y la nueva vida de Isaac es de 8.
- Si al lanzar un d20 de ataque físico se obtiene un 20, el daño se duplica. Por otro lado, si se obtiene un 20 al evadir un ataque mágico, no se recibe daño.

- Un personaje puede dedicarse a defender hasta su próximo turno. En este estado, si fuera a recibir un ataque físico, el atacante debe tirar dos d20 en lugar de uno y ocupar el de **menor** valor. Por otro lado, si fuera a recibir un ataque mágico, el personaje defensor podrá lanzar dos d20 en lugar de uno y ocupar el de **mayor** valor.

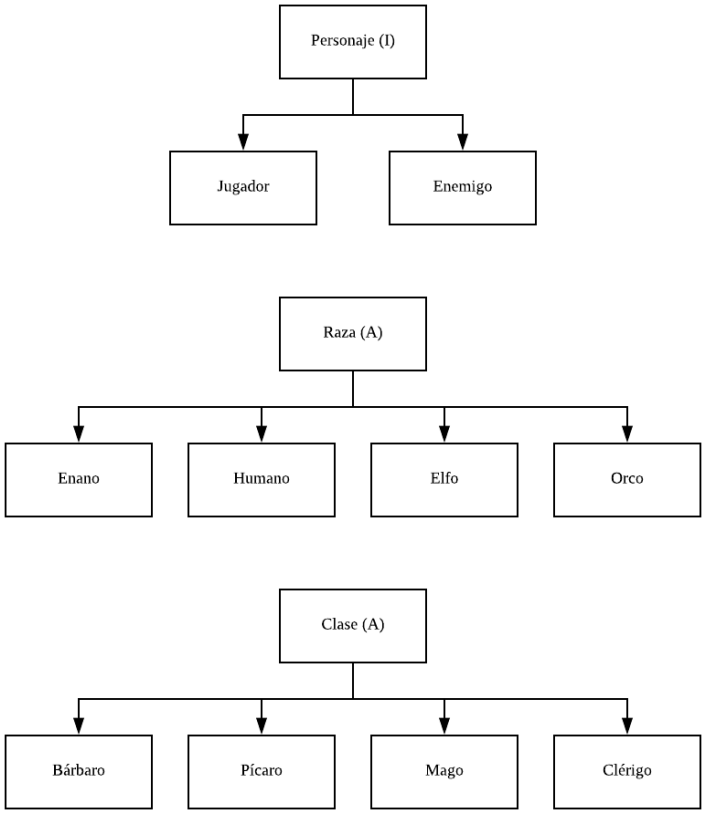
3. Requerimientos

- Se debe implementar un programa que permita jugar el juego descrito en la sección anterior.
- El juego se jugará mediante linea de comando y se debe entregar feedback de lo que ocurre en la batalla mediante texto. El jugador utilizará números para seleccionar la raza y la clase de su personaje (se debe mostrar un menú ad-hoc); además para tener control de lo que ocurre durante el juego.
- El jugador y los enemigos serán personajes, cada uno con su propia raza y clase. Los enemigos entrarán a la pelea en el mismo orden que fueron creados e ingresados en la lista de enemigos.
- Los personajes, las razas y las clases deberán ser creados según el esquema de herencia presentado en la **Sección 5**.
- En la interfaz de Personaje se deben declarar las siguientes funciones a implementar:
  - **asignarRaza**: crea una instancia de la clase Raza dentro de Personaje.
  - **asignarClase**: crea una instancia de la clase Clase dentro de Personaje.
  - **asignarVida**: asigna la cantidad de vida del personaje.
  - **asignarNombre**: asigna un nombre al personaje.
- En la clase abstracta Raza se deben declarar las siguientes funciones abstractas, las cuales deben ser implementadas en las clases hijo respectivas:
  - **crearRaza**: asigna las estadísticas asociadas a la raza.
  - **habilidad**: aplica los efectos de la habilidad respectiva de cada raza.
- En la clase abstracta Clase se deben declarar las siguientes funciones abstractas, las cuales deben ser implementadas en las clases hijo respectivas:
  - **crearClase**: asigna los valores asociados a la clase.
  - **ataque**: efectúa la acción de atacar, ya sea de tipo físico o mágico, cambiando la vida del enemigo en caso de efectuar daño.
  - **defender**: cambia el estado de un personaje a defensivo.

- En la clase Juego se debe implementar el método estático **lanzarDados**, el cual debe retornar el número obtenido por lanzar cualquier tipo de dado (ya sea d6, d8 o d20).
- Se debe utilizar una lista (java.Util.List) para la para la creación y manejo de personajes enemigos.
- Se debe trabajar sobre sistema operativo CentOS presente en los computadores del LDS.
- El programa debe incluir un archivo **makefile** o **ant** para su compilación y ejecución. Su funcionamiento debe explicarse con detalle en un archivo **README**.
- Cada clase debe tener su propio archivo .java asociado.
- Cada atributo debe tener un getter y setter asociado para su manipulación de forma externa. Los getter y setter deben ser utilizados en el programa.

4. Esquema de Herencia

Los siguientes son los esquemas de herencia que se deben implementar en el programa, estos compuestos por las clases, clases abstractas (A) e interfaces (I).



5. Archivos a Entregar

Los archivos a entregar deberán ser los siguientes:

- Raza.java
- "Nombre de Raza".java (un archivo por cada Raza)
- Clase.java
- "Nombre de Clase".java (un archivo por cada Clase)
- Jugador.java
- Personaje.java
- Enemigo.java
- Juego.java
- makefile o ant (Archivo de compilación automática)
- README.txt (archivo manual).

Los alumnos pueden crear más archivos si lo estiman necesario, siempre y cuando adjunten los solicitados anteriormente.

6. Sobre Entrega

- La revisión se efectuará sobre el sistema operativo CentOS presente en los computadores del LDS.
- **Se debe trabajar en grupos de DOS personas.**
- La entrega debe realizarse en tarball (tar.gz) y debe llevar el nombre: "Tarea3LP_RolIntegrante-1_RolIntegrante-2.tar.gz". **Tareas que no cumplan esta regla llevarán descuento**.
- El archivo README.txt debe contener nombre y rol de los integrantes del grupo e instrucciones **claras** para la compilación y utilización de su programa. Aquí pueden agregar también distintos supuestos que hayan utilizado para el desarrollo (conversado previamente con los ayudantes). **Tareas que no cumplan esta regla llevarán descuento**.
- Debe estar presente el archivo **makefile** para que se efectúe la revisión.
- Su código no debe presentar warnings y debe estar comentado de tal forma que sea comprensible. Éstos deben venir antes del código que comentan y, en caso de funciones, debe especificar los parámetros, los valores de retorno y una breve descripción de lo que hace. **Tareas que no cumplan esta regla llevarán descuento**.

- Todas las dudas deben ser hechas por la plataforma *Moodle*.
- La entrega será mediante *Moodle* y el plazo máximo de entrega es hasta el Domingo 11 de Noviembre a las 23:55.
- Por cada día o fracción de atraso se descontarán **30 puntos**. Si la fracción es menor a 20 minutos, el descuento será de sólo **5 puntos**.
- Las copias serán evaluadas con nota 0.

7. Calificación

La escala a utilizar para la revisión de la tarea es la siguiente:

- Reglas del juego implementadas (Ejecutable) (40 Puntos)
- Clases bien definidas (Encapsulamiento) (30 Puntos)
- Código (Orden, implementación) (30 Puntos)

Los descuentos que pueden ocurrir son:

- No compila (100 puntos)
- Warnings (5 Puntos c/u)
- No respetar reglas de entrega (40 puntos c/u)