Practica 2 Prgramación
UF 4
Leyre Iirarte liriarte@ilerna.com
Ilerna BCN
DAX
Entrega: P2_NombreApellidos.zip
Git enviar URL

ZOMBICIDE

¡Bienvenidos al oscurantismo de la Edad Media! En su delirante búsqueda de conocimientos prohibidos y el poder definitivo, los nigromantes han desatado el apocalipsis sobre el mundo de los vivos. Sus zombis han destruido los ejércitos del Rey y ahora campan a sus anchas, sembrando el caos y la muerte por toda la región.

DESCRIPCIÓN

Vamos a necesitar de implementar una práctica que nos permita simular el comportamiento del juego de mesa "Zombicide". Para ello, necesitaremos de una cantidad importante de ficheros java para crear una estructura como la que se plantea a continuación.

HUMANOIDE

Necesitaremos de una clase **padre que es Humanoide** que registrará todos los datos comunes y los métodos necesario tanto para los jugadores como para los zombies: Nombre, Salud, Salud Máxima i si está vivo o no.

ZOMBIE

La Clase zombie será la clase padre de 3 distintas. Todos los zombies tienen en común:

- **Movimiento**: determina que armas pueden alcanzarle.
- **Daño**: determina las heridas que hacen a los jugadores.
- **Tipo**: determina qué tipo de zombie es (puedes utilizar char, int, string, …, lo que prefieras).

Practica 2 Prgramación
UF 4
Leyre Irarte liriarte@ilerna.com
Ilerna BCN
DAX
Entrega: P2_NombreApellidos.zip
Git enviar URL

JUGADOR

La clase jugador tendrá como atributos exclusivos el Arma del jugador y un booleano para saber si ese jugador ha pasado en esa ronda o no.

ARMA

Las armas son importantes en el juego, por lo que deberemos tener también una estructura definida:

Las armas tendrán como atributo:

- **Nombre**: nombre del arma (se asigna como nombre el nombre de la clase hijo).
- **Daño**: determina si puede matar a un zombie o no (va del 1 al 3).
- **Alcance**: determina el número de objetivos al que puede atacar el arma.
- **Acierto**: determina el resultado a obtener para matar a un zombie.

FUNCIONAMIENTO DE LAS ARMAS:

Cuando un jugador realiza un ataque deberá realizar los siguientes pasos según el número de alcance de su arma (**tantas veces como indique el alcance**):

1) Seleccionamos un zombie aleatoriamente de los presentes.
2) Miramos si el arma tiene igual o más daño que la salud del zombie:
    a. **SI**: realizamos un aleatorio de 1 a 6 (caras de un dado) y miramos si es igual o mayor al acierto del arma:
        i. **SI**: avisamos de que el zombie ha muerto: `Corredor muerto`
        ii. **NO**: avisamos de que el ataque ha fallado: `Ha fallado tu ataque con un 3`
    b. **NO**: avisamos que el ataque ha sido esquivado: `Gordo ha evitado el ataque`

Practica 2 Prgramación
UF 4
Leyre Iirarte liriarte@ilerna.com
Ilerna BCN
DAX
Entrega: P2_NombreApellidos.zip
Git enviar URL

CLASES ARMA

La clase padre Arma deberá tener un método de ataque especial, que no deberá hacer nada más que sacar por consola un "no hay habilidad especial". Sin embargo, todas las otra armas, deberán sobreescribir este método para implementarlo:

- **Hacha**: Mata gratis a 1 gordo.
- **Hechizo**: Mata gratis a 2 caminantes.
- **Arco**: Mata gratis a 1 corredor.
- **Espada**: Mata gratis a 2 zombies aleatorios.

Deberá disponer de un **constructor vacío** en el cual añadiremos al jugador una arma por defecto que será una **"Daga" de daño 1 y alcance 1 y acierto 4** (NO se requiere de una clase Daga).

EL JUEGO

Nuestro primer menú:

```
1- Nueva partida
2- Nuevo personaje
0- Salir
```

0) Salir nos saldrá del programa.
1) Nueva partida nos permite empezar un nuevo juego (lo veremos a continuación).
2) Nuevo personaje nos permite crear un nuevo personaje. Los nuevos personajes tendrán siempre salud 5 y empezarán con un arma por defecto (Daga).

Podemos tener un máximo de 10 personajes pero siempre deberemos jugar con 6, para ello, al pulsar la opción de menú de "Nueva partida" nos deberá preguntar con qué personajes queremos jugar (mínimo 3 máximo 6).

- Si solo dispones de 3 personajes deberán seleccionarse automáticamente: `Todos tus personajes han sido seleccionados`
- De haber más de 3 se **permitirá al jugador escoger de 3 a 6.**

Practica 2 Programación
UF 4
Leyre Iriarte liriarte@ilerna.com
Ilerna BCN
DAX
Entrgea: P2_NombreApellidos.zip
Git Enviar URL

RONDA Y NIVEL

Cada nivel constará de unas fases muy marcadas. Para ello se recomienda que la opción de menú de "**Nueva partida" inicie una nueva clase "Partida"** que reciba los jugadores que van a luchar y el nivel (que empezará entre el 3 y el 6 según el número de jugadores que participan).

Las fases de cada ronda son:

1) Generar Zombies aleatorios entre los 3 tipos disponibles. (se generarán tantos zombies como **nivel**).
2) Por cada jugador, mostrar el menú de acciones (lo veremos a continuación).
3) Cuando han jugado todos los jugadores atacan los zombies que estén vivos (harán una herida a un personaje aleatoriamente por cada punto de movimiento que tengan).
4) Incrementamos la ronda en 1 y volvemos a empezar.

NOTA: El nivel se incrementa siempre que cuando le toca jugar a un jugador no hay zombies. Si se da el caso se reinicia del mismo modo la ronda a 0 y se empieza el nuevo nivel.

```
Todos tus personajes han sido seleccionados

|----- NIVEL: 3 - 0 -----|

==| Gordo Gordo Corredor |==
JUGADOR: James S:6 Arma[MANDOBLE DANO:2 DIST:1 ACIER:4]
1- Atacar
2- Habilidad Especial
3- Buscar
4- Cambiar Arma
0- Pasar
```

La estructura del menú es la siguiente:

0) **Pasar**: El jugador no hace nada en esta ronda.
1) **Atacar**: El jugador realiza un ataque con el arma que tenga puesta (llamando a un método atacar de la clase arma).
2) **Habilidad Especial**: El jugador realiza el ataque especial del arma. Solamente se puede realizar una vez por nivel.
3) **Buscar**: Obtendremos un objeto aleatorio (lo veremos en el apartado MAIN).

4) **Cambiar Arma**: En caso de tener varias armas, podemos escoger la que queremos. Para ello se mostrará un menú con todas las armas disponibles y seleccionaremos la que queremos tener equipada (requerirá de un arraylist de armas disponibles en cada personaje).

FUNCION PRINCIPAL (INIT)

Como clase principal tendremos la clase Zombicide. Esta clase nos permitirá realizar unas inicializaciones:

- **InitPersonajes**: Hay 3 personajes por defecto en el juego, se deberán inicializar para poder realizar partidas sin tener que crear más personajes obligatoriamente:

  James tiene salud 7 y le crearemos un arma especial llamada "Mandoble" de daño 2, alcance 1 y acierto 4 (básica, de tipo Arma).

  Marie tiene salud 5.

  Jaci tiene salud 5.

- **InitObjetos**: Deberemos tener una arrayList de tipo Arma que nos permita guardar las distintas armas disponibles en el juego, para cuando un personaje realice la opción de buscar pueda obtenerlas.

  Arco Largo de tipo Arco de daño 1, alcance 2 y acierto 3.

  Hacha doble de tipo Hacha daño 2, alcance 1 y acierto 3.

  Bola de fuego de tipo Hechizo de daño 1, alcance 3 y acierto 4.

  Espada corta de tipo Espada de daño 1, alcance 1 y acierto 4.

Dado que este arrayList de tipo Arma estará en el fichero con el main, se puede generar un **método estático que nos permita coger un arma aleatoria**. Este método **getArma()** debería devolver un arma pero con un determinado porcentaje de acierto (es decir, que si tienes buena suerte te da una buena arma, sino te da de nuevo una Daga (arma básica) o nada).

Practica 2 Prgramación
UF 4
Leyre Iirarte liriarte@ilerna.com
Ilerna BCN
DAX
Entrega: P2_NombreApellidos.zip
Git enviar URL

INIT ZOMBIES

Para inicializar a los zombies de manera aleatoria deberás tener en cuenta los siguientes valores. Ten en cuenta que sus habilidades especiales podrán ser activadas en el momento de morir, realizando un aleatorio de 0 a 100 y de sacar 95 o más se activarán.

- **Caminante**: Salud 1, movimiento 1, daño 1 y su habilidad especial es la de invocar a tantos caminantes como hay (dobla la cantidad de caminantes vivos).
- **Gordo**: Salud 2, movimiento 1, daño 1 y su habilidad especial permite eliminar a otro gordo cuando muere.
- **Corredor**: Salud 1, movimiento 2, daño 1 y su habilidad especial le permite eliminar a todos los corredores aún vivos.

EVALUACIÓN

Se requiere:

- Utilizar métodos toString() sobreescritos para mostrar las informaciones de toda las clases que sea necesario (la mayoría).
- Utilizar herencia en todos los apartados especificados.
- Crear métodos correctamente en cada una de las clases hijos según explicado en la práctica y en teoría.
- Respetar la estructura de los prints para facilitar la corrección.
- Implementar partes extra para tener la máxima nota o subir puntos.
- Crea comentarios justo encima de cada función para explicar su función.

Para la entrega se deberá copiar **todo el proyecto** en un zip con el formato:

P2_NombrApellidos.zip (o **.rar**).