TRABAJO SEGUNDA EVALUCACIÓN

SEÑOR DE LOS ANILLOS

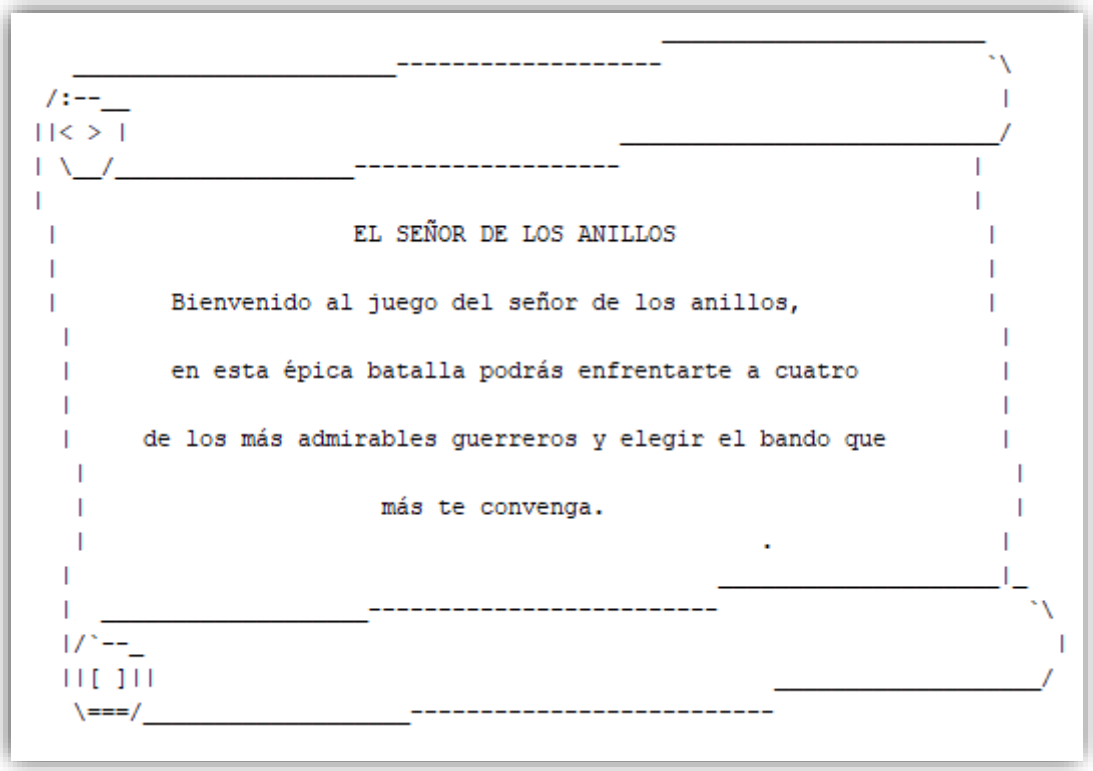
Ivanna Ariza Morán

Índice:

1- Explicación del juego

Mi programa es un juego de RPG básico que te presenta dos bandos (Luz y Oscuridad). Al elegir un bando automáticamente el bando contrario se convierte en tu enemigo.

Lo primero que sale en la pantalla una vez se ejecuta el programa es un pergamino donde en unas breves palabras se explica un poco en lo que consistirá el juego para que el usuario entienda.



Dentro de cada bando hay 4 personajes jugables con sus vidas, defensas y ataques que varían según la elección del usuario. Una vez el usuario ha elegido su personaje para la batalla, en el bando enemigo se genera aleatoriamente un contrincante de los cuatro personajes posibles.

Los dos bandos posibles que puede elegir el usuario son: **Luz** y **Oscuridad**

```
A: ATAQUE, D: DEFENSA, V: VIDA

      Tienes dos bandos:

       * 1 - Luz
         Compuesto por:
          -> 0 - Gandalf (A:25/D:10/V:100)
          -> 1 - Aragorn (A:20/D:00/V:100)
          -> 2 - Legolas (A:15/D:25/V:100)
          -> 3 - Gimli (A:10/D:20/V:100)

       * 2 - Oscuridad
         Compuesto por:
          -> 0 - Saruman (A:25/D:10/V:100)
          -> 1 - Balrog (A:35/D:10/V:100)
          -> 2 - Lurtz (A:25/D:15/V:100)
          -> 3 - Orco (A:5/D:5/V:100)

      ¿Cúal eliges?
```

Los ocho personajes existentes a su vez pueden ser: guerreros o magos, las diferencias entre ambos son las siguientes:

- Guerrero:

  El guerrero tiene dos habilidades.

  ```
  Que quieres hacer:
  1 - Atacar
  2 - Defender
  ```

  La habilidad de **atacar** consiste en bajarle vida al contrincante en la medida que el personaje elegido pueda. Por ejemplo, si el personaje elegido es Aragorn, su daño máximo es 20, por ende, el daño posible que tu personaje puede hacer va desde 1 hasta 20 de forma aleatoria, habiendo posibilidades de hacer un ataque crítico (el doble del daño aleatorio obtenido).

  La habilidad de **defender** te ayuda a minimizar el daño que recibes por parte de tu enemigo o a no recibir nada en total. El daño que recibes finalmente es la resta del daño aleatorio menos la defensa de tu personaje. Por ejemplo, si tu defensa es de 10 y el daño que te hacen es de 20, 20-10=10, finamente el daño que te ocasionan es de 10, pero si la defensa es mayor al daño no te hace nada.

- Mago:

El mago tiene tres habilidades a diferencia del guerrero.

```
Que quieres hacer:
1 - Atacar
2 - Defender
3 - Curarse
```

La habilidad de **atacar**, explicada anteriormente le baja la vida al tu enemigo.

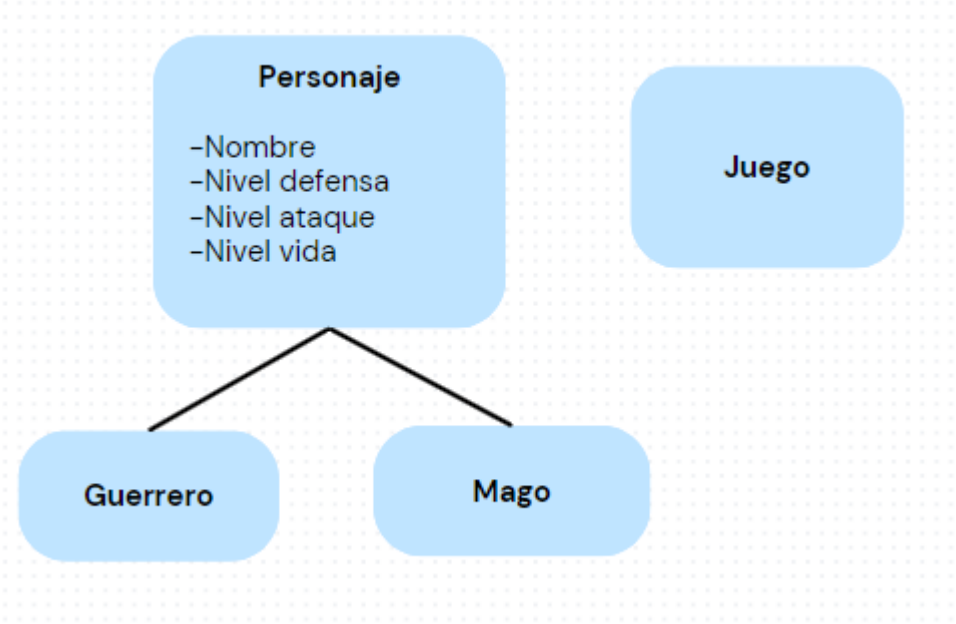
La habilidad de **defender**, también explicada anteriormente consiste en minimizar el daño para recibir menos o hasta ningún daño.

La habilidad de **curarse** es la única diferente a las habilidades del guerrero. Si el usuario elige curarse hay un porcentaje de posibilidades de que lo haga. De ser así solo puede curarse 10 puntos de vida, pero si tu vida está a 100, que es el máximo, tu personaje no se curaría y perderías el turno.

La batalla acaba una vez que el personaje del usuario muere o por el contrario, muere el personaje enemigo, habiendo siempre un ganador y un perdedor.

2- Esquemas y clases

Dentro de mi programa tengo cuatro clases sin contar el main.



La clase Personaje es una clase abstracta que tiene cuatro atributos que se van a usar durante todo el programa: el nombre del personaje, el nivel de defensa, el nivel de ataque y el nivel de vida.

El constructor de esta clase es el siguiente:

```
//constuctor
public Personaje(String _nombre, int D, int A, int V){
    nombre = _nombre;
    nivelDef = D;
    nivelAtq = A;
    nivelVida = V;
}
```

La vida es estándar para todos los personajes (100), en cambio la defensa y el ataque van cambiando según el personaje elegido.

De la clase Personaje heredan otras dos clases: **Guerrero** y **Mago**, las cuales heredan todos los atributos del padre y no tienen ningún atributo externo.

```
//constuctor
public Guerrero(String nombre, int nivelDef, int nivelAtq, int nivelVida){
    super(_nombre:nombre, D:nivelDef, A:nivelAtq, V:nivelVida);
}
```

En ambas clases tengo el mismo constructor ya que las dos heredan del padre, Personaje.

Por último, tengo la clase **Juego** que es donde se desarrolla todas las interacciones del programa. En esta clase no tengo ningún constructor.

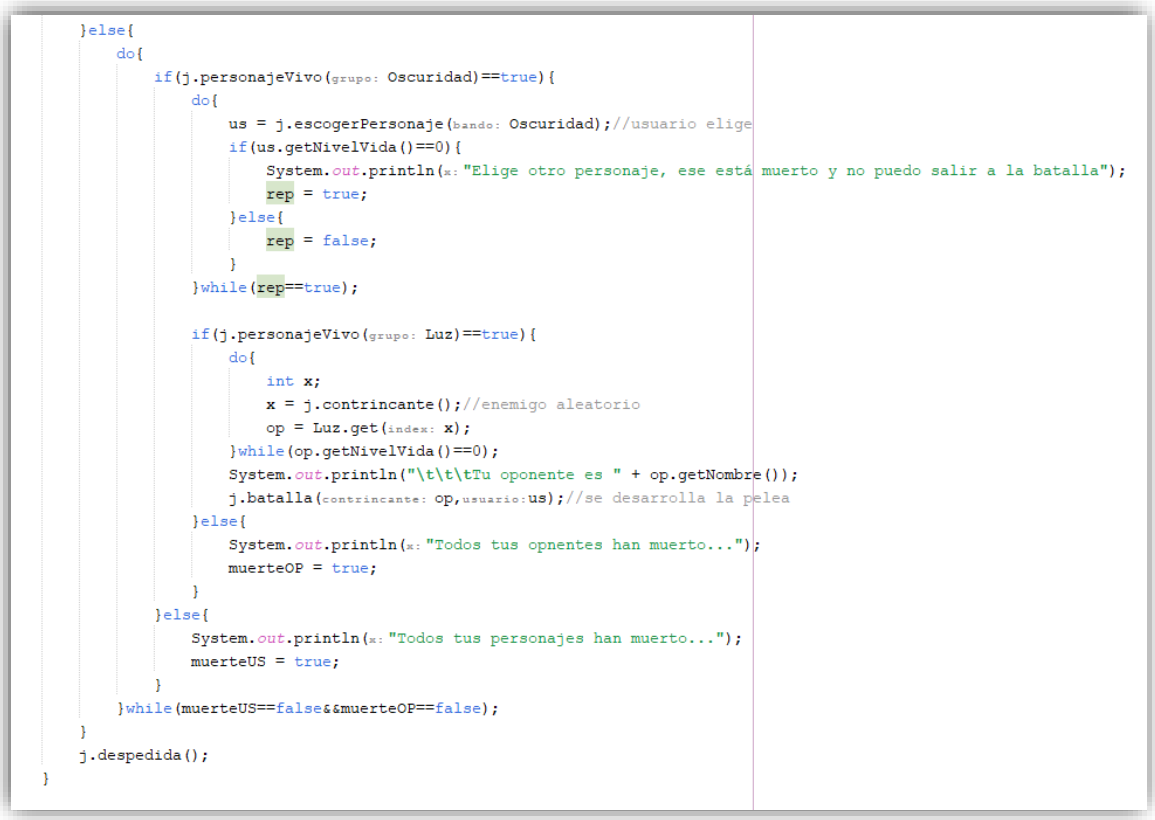
En el main simplemente llamo a los métodos necesarios para que todo se desarrolle sin problema.

```java
public static void main(String[] args) {
    //creacion de los dos arrayList
    ArrayList<Personaje> Oscuridad = new ArrayList();
    ArrayList<Personaje> Luz = new ArrayList();
    Juego j = new Juego();//inicializo la clase del juego
    boolean rep = false;
    Personaje us;
    Personaje op;
    boolean muerteOP = false, muerteUS = false;

    j.presentacion();//imprimo el pergamino
    j.creacion(Oscuridad, Luz);//creo los 8 personajes en sus arrays
    if(j.EscogerBando()==1){
        do{
            if(j.personajeVivo(grupo: Luz)==true){
                do{
                    us = j.escogerPersonaje(bando: Luz);//usuario elige
                    if(us.getNivelVida()==0){
                        System.out.println(x: "Elige otro personaje, ese está muerto y no puedo salir a la batalla");
                        rep = true;
                    }else{
                        rep = false;
                    }
                }while(rep==true);
```

```java
                    if(j.personajeVivo(grupo: Oscuridad)==true){
                        do{
                            int x;
                            x = j.contrincante();//enemigo aleatorio
                            op = Luz.get(index: x);
                        }while(op.getNivelVida()==0);
                        System.out.println("\t\t\tTu oponente es " + op.getNombre());
                        j.batalla(contrincante: op,usuario:us);//se desarrolla la pelea
                    }else{
                        System.out.println(x: "Todos tus opnentes han muerto...");
                        muerteOP = true;
                    }
                }else{
                    System.out.println(x: "Todos tus personajes han muerto...");
                    muerteUS = true;
                }
            }while(muerteUS==false&&muerteOP==false);

        }else{
```



```
    }else{
        do{
            if(j.personajeVivo(grupo: Oscuridad)==true){
                do{
                    us = j.escogerPersonaje(bando: Oscuridad);//usuario elige
                    if(us.getNivelVida()==0){
                        System.out.println(x: "Elige otro personaje, ese está muerto y no puedo salir a la batalla");
                        rep = true;
                    }else{
                        rep = false;
                    }
                }while(rep==true);

                if(j.personajeVivo(grupo: Luz)==true){
                    do{
                        int x;
                        x = j.contrincante();//enemigo aleatorio
                        op = Luz.get(index: x);
                    }while(op.getNivelVida()==0);
                    System.out.println("\t\t\tTu oponente es " + op.getNombre());
                    j.batalla(contrincante: op,usuario:us);//se desarrolla la pelea
                }else{
                    System.out.println(x: "Todos tus opnentes han muerto...");
                    muerteOP = true;
                }
            }else{
                System.out.println(x: "Todos tus personajes han muerto...");
                muerteUS = true;
            }
        }while(muerteUS==false&&muerteOP==false);
    }
    j.despedida();
}
```

Como se puede ver en la foto, lo primero que hago en el main es Crear los dos ArrayList que voy a usar durante todo el programa de los dos bandos (Luz y Oscuridad). También inicializo la clase juego que es la que tiene dentro todos los métodos para poder usarlos a lo largo de todo el main.

Después de esto imprimo la presentación del programa, creo los personajes y los incluyo en sus arrays con j.creacion() y según el bando que elija el usuario me lleva a una parte del if o a la otra.

Si introduce 1, como se ve en el código es el bando de la Luz así que el oponente es de la oscuridad aleatoriamente y viceversa. Se ejecuta la batalla, sale un ganador y por último la despedida del usuario.

La siguiente foto es el del método de la creación de los personajes.

En las primeras ocho líneas creo los 4 personajes del bando de Luz y los cuatro personajes del bando de Oscuridad. Según sea mago o guerrero tengo que crearlos con clase y dentro de los paréntesis esta el nombre, su nivel de defensa, su nivel de ataque y su vida que, como dije anteriormente, todos tienen 100 de vida.

```java
public void creacion(ArrayList<Personaje> Oscuridad, ArrayList<Personaje> Luz){
    Personaje Gandalf = new Mago(nombre: "Gandalf", nivelDef:10,nivelAtq:25,nivelVida:100);
    Personaje Aragorn = new Guerrero(nombre: "Aragorn", nivelDef:20, nivelAtq:20, nivelVida:100);
    Personaje Legolas = new Guerrero(nombre: "Legolas", nivelDef:25,nivelAtq:15,nivelVida:100);
    Personaje Gimli = new Guerrero(nombre: "Gimli", nivelDef:20,nivelAtq:10,nivelVida:100);

    Personaje Saruman = new Mago(nombre: "Saruman", nivelDef:10,nivelAtq:25,nivelVida:100);
    Personaje Balrog = new Mago(nombre: "Balrog", nivelDef:10,nivelAtq:35,nivelVida:100);
    Personaje Lurtz = new Guerrero(nombre: "Lurtz", nivelDef:15,nivelAtq:25,nivelVida:100);
    Personaje Orco = new Guerrero(nombre: "Orco", nivelDef:5, nivelAtq:5, nivelVida:100);

    Luz.add(e: Gandalf);
    Luz.add(e: Aragorn);
    Luz.add(e: Legolas);
    Luz.add(e: Gimli);

    Oscuridad.add(e: Saruman);
    Oscuridad.add(e: Balrog);
    Oscuridad.add(e: Lurtz);
    Oscuridad.add(e: Orco);
}
```

En las últimas 8 líneas de código añado cara personaje a su array correspondiente.

Ahora, tras hacer esto se ejecuta EscogerBando() donde enseño este menú ya visto anteriormente:

```
----------------------------------------
   A: ATAQUE, D: DEFENSA, V: VIDA

        Tienes dos bandos:

         * 1 - Luz
           Compuesto por:
            -> 0 - Gandalf (A:25/D:10/V:100)
            -> 1 - Aragorn (A:20/D:20/V:100)
            -> 2 - Legolas (A:15/D:25/V:100)
            -> 3 - Gimli (A:10/D:20/V:100)

         * 2 - Oscuridad
           Compuesto por:
            -> 0 - Saruman (A:25/D:10/V:100)
            -> 1 - Balrog (A:35/D:10/V:100)
            -> 2 - Lurtz (A:25/D:15/V:100)
            -> 3 - Orco (A:5/D:5/V:100)

        ¿Cúal eliges?
```

En este menú lo que enseño es las cualidades de cada personaje y le pido al usuario que elija uno de los dos bandos introduciendo 1 o 2. Si por algún casual el usuario introduce algún otro número que no es de los que pido, en el código hay un try catch:

```
int bando = 0;
    do{
        try{
            System.out.println("\n\n\n\n\t\t---------------------------------------");
            System.out.println("\t\t   A: ATAQUE, D: DEFENSA, V: VIDA\n\n"
                    + "\t\t\tTienes dos bandos:"
                    + "\n\n\t\t\t * 1 - Luz\n"
                    + "\t\t\t   Compuesto por:\n"
                    + "\t\t\t    -> 0 - Gandalf (A:25/D:10/V:100)\n"
                    + "\t\t\t    -> 1 - Aragorn (A:20/D:20/V:100)\n"
                    + "\t\t\t    -> 2 - Legolas (A:15/D:25/V:100)\n"
                    + "\t\t\t    -> 3 - Gimli (A:10/D:20/V:100)\n"
                    + "\n\t\t\t * 2 - Oscuridad\n"
                    + "\t\t\t   Compuesto por:\n"
                    + "\t\t\t    -> 0 - Saruman (A:25/D:10/V:100)\n"
                    + "\t\t\t    -> 1 - Balrog (A:35/D:10/V:100)\n"
                    + "\t\t\t    -> 2 - Lurtz (A:25/D:15/V:100)\n"
                    + "\t\t\t    -> 3 - Orco (A:5/D:5/V:100)\n"
                    + "\n\t\t\t¿Cúal eliges?");
            System.out.print("\n\t\t ");
            bando = sc.nextInt();

            if(bando<1||bando>2){
                System.out.println("\n\t\t Introduce bien tu respuesta.");
            }
            System.out.println("\t\t---------------------------------------");
        }
        catch(InputMismatchException e){
            e.getMessage();
            System.out.println("Solo se aceptan dos numeros, vuelve a intentarlo");
            sc.nextLine();//limpiar la variable sc
        }

    }while(bando<1 || bando>2);
return bando;
```

Este try catch se encarga de capturar el error y que el programa no falle, me imprime “Introduce bien tu respuesta” y vuelve a repetir el menú para que el usuario vuelva a meter su respuesta dando la posibilidad a volverlo a intentar.

```
            ¿Cúal eliges?

        8

        Introduce bien tu respuesta.
       ---------------------------------------
```

Al introducir bien la respuesta lo siguiente que te pregunta es:

```
 1
----------------------------------------
  ¿Que personaje quieres usar?
introduce solo los números dados anteriormente
```

Ahora el usuario, habiendo elegido el bando 1 que es la Luz, debe elegir entre los 4 personajes.

```
 1
----------------------------------------
  ¿Que personaje quieres usar?
introduce solo los números dados anteriormente

 d
Introduce un numero valido
  ¿Que personaje quieres usar?
introduce solo los números dados anteriormente
```

Como se ha visto anteriormente se ha usado otro try catch que comprueba lo mismo que el anterior, si el usuario introduce letras o caracteres extraños, lo captura en la excepción "InputMismatchException".

```
do{
    try{
        do{
            System.out.println("\t\t  ¿Que personaje quieres usar? \n\t\tintroduce solo los números dados anteriormente");
            System.out.print("\n\t\t ");
            eleccion = sc.nextInt();
            if(eleccion<0||eleccion>3){
                System.out.println("\t\t  Introduce bien tu respuesta");
            }
        }while(eleccion<0||eleccion>3);
        System.out.println( "\t\tTu personaje elegido finalmente es " + bando.get(index: eleccion).getNombre());
        p = bando.get(index: eleccion);
    }catch(InputMismatchException e){
        rep = true;
        e.getMessage();
        System.out.println("\t\tIntroduce un numero valido");
        sc.nextLine();
    }
}while(rep);//repite el bucle
return p;//devuelvo el personaje elegido por el usuario
```

Tengo una variable booleana llamada rep que la uso para que el bucle se repita siempre y cuando entre en el catch porque ha introducido un carácter no válido o si el número introducido es diferente a los posibles (0, 1, 2, 3).

Ahora, ya entrando en la batalla tengo 4 métodos aleatorios: el primero elige aleatoriamente al contrincante. El segundo sirve para, también con un random, elegir el daño que va a hacer el personaje, tanto el del usuario como el de la máquina.

```java
public int probCritico(){
    Random cr = new Random();
    int critico = cr.nextInt(origin: 1,bound: 100);
    if(critico<15){
        return critico;
    }else{
        return 0;
    }

}
```

El tercero, el cuarto y el quinto tienen la misma dinámica, también con un random. Con una probabilidad del 15%, si sale se usa la defensa que tenga el usuario, con el mismo porcentaje, se genera un ataque crítico (el doble del ataque que haya salido) y se recupera un 10 de vida.

Dependiendo de que personaje elija el usuario (mago o guerrero) enseñaré un menú de ataque u otro:

```
Comienza Gandalf
Vida de Gandalf: 100
Vida de Balrog: 100
Que quieres hacer:
1 - Atacar
2 - Defender
3 - Curarse
```

En este caso, Gandalf es un mago así que tiene tres opciones, atacar, defender o curarse.

```
Comienza Aragorn
Vida de Aragorn: 100
Vida de Saruman: 100
Que quieres hacer:
1 - Atacar
2 - Defender
```

Por otro lado, Aragorn es un guerrero así que estos solo pueden atacar o defender ya que no se curan.

Al elegir el usuario, la maquina también elige de manera aleatoria.

```java
public int EleccionContrincante(Personaje p){
    Random rd = new Random();
    if(p instanceof Guerrero){
        int g = rd.nextInt(origin: 1,bound: 2);
        return g;
    }else{
        int m = rd.nextInt(origin: 1,bound: 3);
        return m;
    }
}
```

En esta parte de código utilizo el instance of para distinguir si el personaje del contrincante entra dentro del array de guerreros o del array de magos, si es el primero, como he dicho anteriormente solo tiene 2 opciones, por ello hago un random de 1 a 2 para que elija. Por otro lado, en los segundos es un random del 1 al 3 ya que se curan, así que son 3 habilidades.

Una vez terminada la batalla uno de los dos personajes gana, en caso de que sea el usario vería el siguiente mansaje:

```
¡Felicidades, has ganado!
```

De no ser así, y el usuario pierde se ve este otro mensaje:

```
Oh no...has perdido.
```

Tras esto, vuelve a salir el menú de elección y el usuario puedo volver a elegir un de los cuatro personajes siempre y cuando sigan vivos.

Si por el contrario todos tus personajes ha muerto

```java
}else{
    System.out.println(x: "Todos tus personajes han muerto...");
    muerteUS = true;
```

Se imprime este mensaje que te avisa que todos han muerto y eso significa que has perdido si el oponente sigue vivo.

En cambio, si todos los que mueren son del contrincante aparece lo siguiente:

```java
}else{
    System.out.println(x: "Todos tus opnentes han muerto...");
    muerteOP = true;
}
```

De esta manera terminaría el programa con un breve mensaje de despedida:

```
public void despedida(){
    System.out.println(x: "\n\nEl juego se ha termiando, gracias por jugar :)");
}
```

Dándole a entender al usuario que todo ha acabado.

3- Test unitarios

En los test unitarios se ha incluido un @BeforeEach para que antes de cada uno de los test, se ejecute esta parte de código donde creo los dos ArrayList que usaré y la clase juego que es la que contiene todo.

```
Juego j;
ArrayList<Personaje> Luz;
ArrayList<Personaje> Oscuridad;
```

```
@BeforeEach
public void setUp() {
    j = new Juego();
    Oscuridad = new ArrayList();
    Luz = new ArrayList();

    j.creacion(Oscuridad, Luz);
}
```

Y los dos test que he hecho han sido unas comprobaciones que tanto el array de Oscuridad como el array de Luz no sean nulos para evitar errores y ambos test se pasaron perfectamente.

```
@Test
public void contrincanteTest(){
    assertNotNull(actual: j.contrincante(contrincante: Oscuridad), message:"Error, valor nulo");
    //el valor es nulo en la variable que elige aleatoriamente el contrincante con el bando de la oscuridad
}

@Test
public void contrincanteTest2(){
    assertNotNull(actual: j.contrincante(contrincante: Luz), message:"Error, valur nulo");
    //el valor es nulo en la variable que elige aleatoriamente el contrincante con el bando de la luz
}
```

4- Ampliaciones

El código tiene aún bastantes carencias ya que se puede quitar bastante código y refactorizarlo mucho más para una misma usabilidad, pero muchísimo menos espacio utilizado para así tener un código más rápido y adecuado.

Hay bastantes partes duplicadas que podrían reducirse a una sola o muchas variables también duplicadas o inservibles.