RETO 2 – PROGRAMACIÓN BÁSICA

VARIANTE 1

La empresa desarrolladora de videojuegos Bugland ha comenzado el desarrollo de uno de sus nuevos títulos.

El juego será desarrollado en Java en un ambiente bidimensional (2-D) y enfocado para usuarios de PC inicialmente.

Usted ha sido contratado como Java Expert Developer, porque ha logrado demostrar habilidades de desarrollo en este lenguaje de programación y se le ha concedido implementar las clases correspondientes a un personaje regular (Clase padre de los personajes involucrados en el juego), personaje del jugador y uno de los jefes que aparecerá en la historia del juego.

La empresa desea que un personaje regular (Enemigo o jugador) pueda golpear, y calcular su distancia con respecto a otro personaje (Esto permitirá realizar el cálculo de la cantidad de daño que le ejerce el personaje a otro).

Por otro lado, el personaje del jugador tendrá las siguientes acciones (Además de las que ya puede hacer un personaje regular):

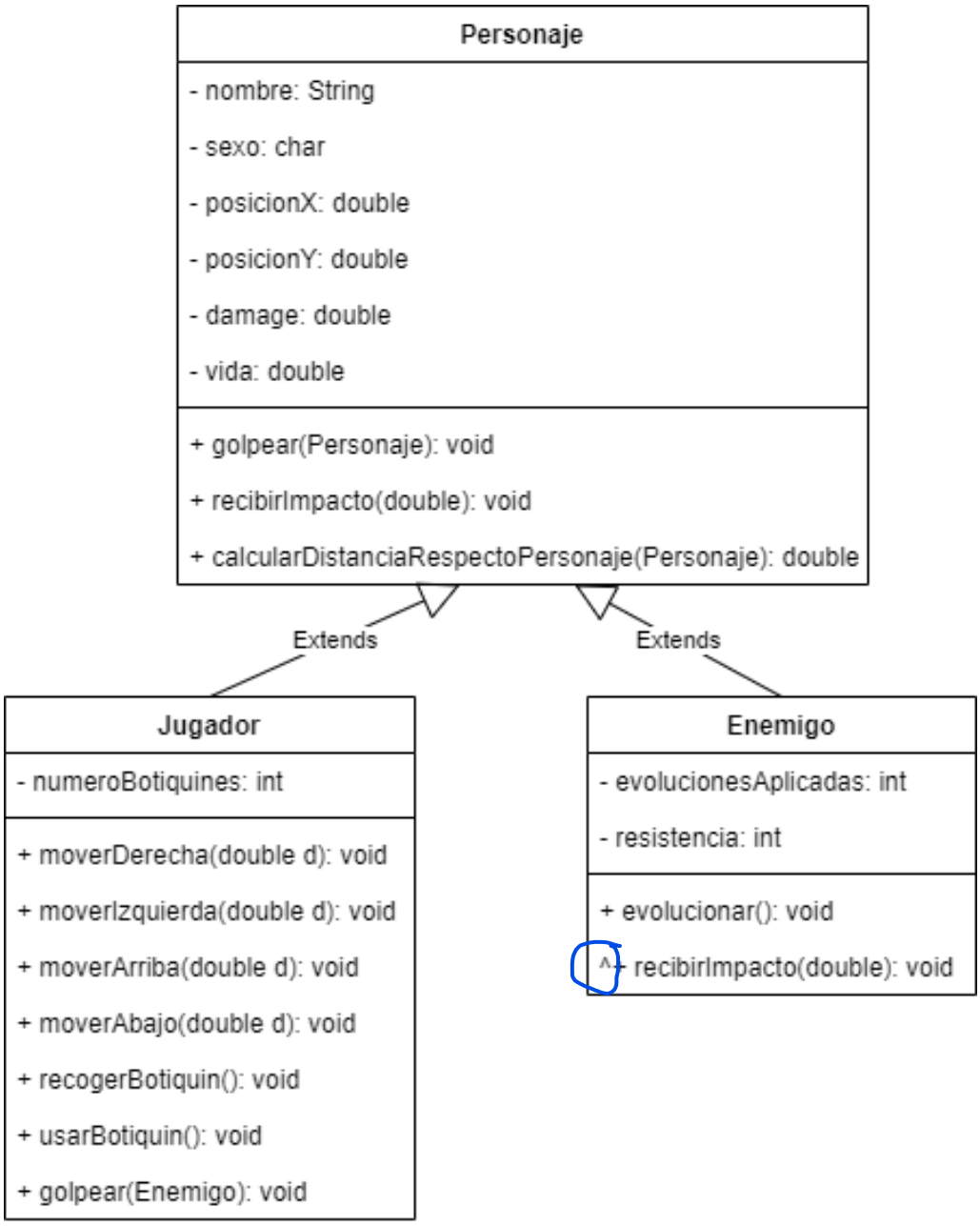
1. Moverse libremente a través del mapa, es decir, que el personaje pueda subir, bajar, ir a la derecha, ir a la izquierda, la unidad de medida será en metros (El personaje no podrá moverse en trayectorias diagonales, solo de manera horizontal y vertical).
2. Recoger botiquines y usarlos cuando se le esté acabando la vida.

En cuanto al enemigo o jefe, debe cumplir con las siguientes observaciones (Además de las que ya puede hacer un personaje):

1. El enemigo estará en una posición fija (No se moverá de ahí).
2. El enemigo evolucionará en dos momentos:
    a. El primer momento, cuando el enemigo cuente con una vida de 30 puntos, la evolución consiste en aumentar su poder de daño en 20 puntos.
    b. El segundo momento, cuando el enemigo cuente con una vida de 10 puntos, la evolución consiste en que la cantidad de daño

recibido causado por los golpes del jugador se reducirán a la mitad (El cuerpo del enemigo adquiere mejor capacidad de resistencia).

Para facilitar la implementación de estas tres clases, el equipo de Ingeniería de software le hace entrega del diagrama de clases, recuerde que los métodos relacionados a los *getters* y *setters*, así como también el método constructor son obviados en el diagrama de clases, pero deberán ser incluidos en el código (Estos métodos deberán ser creados con el estándar camel case: Por ejemplo, si el atributo se llama `nombre`, sus métodos correspondientes a get y set serían `getNombre` y `setNombre`).



Además del diagrama, el equipo de Ingeniería entrega esta documentación para comprender mejor los elementos del diagrama:

Clase Personaje

Atributos

NOMBRE	TIPO DATO	CONCEPTO	INICIALIZACIÓN
`nombre`	`String`	Nombre del personaje	En el método constructor
`sexo`	`char`	Sexo del personaje (`'m'` para masculino y `'f'` para femenino)	En el método constructor
`posicionX`	`double`	Coordenada x del personaje	En el método constructor
`posicionY`	`double`	Coordenada y del personaje	En el método constructor
`damage`	`double`	Parámetro para el cálculo de la cantidad de daño que ejerce el personaje sobre otro	En el método constructor
`vida`	`double`	Cantidad de vida que tiene el personaje durante la partida	Debe estar inicializada en 100 (Este es el valor máximo de vida que puede tener un personaje, no se puede aumentar más ni usando botiquines)

Métodos

NOMBRE	TIPO RETORNO	PARÁMETROS	CONCEPTO
`golpear`	`void`	`Personaje p`: Personaje que recibe el daño	Resta $\frac{damage}{d}$ al atributo `vida` del personaje `p`, donde d es la distancia entre el personaje que invoca el método y el personaje `p`
`recibirImpacto`	`void`	`double d`: Cantidad de vida a restarle al personaje	Resta d a `vida`
`calcularDistanciaRespectoPersonaje`	`double`	`Personaje p`: Personaje con el que se desea calcular la distancia	Retorna la distancia entre el personaje que invoca el método y el personaje `p` (Que entró como parámetro).

Clase Jugador (Hija / Hereda de Personaje)

Atributos

NOMBRE	TIPO DATO	CONCEPTO	INICIALIZACIÓN
`numeroBotiquines`	`int`	Número de botiquines que ha recolectado el personaje	Debe estar inicializada en 0

Métodos

NOMBRE	TIPO RETORNO	PARÁMETROS	CONCEPTO
`usarBotiquin`	`void`	No recibe	Resta 1 a `numeroBotiquines` y suma 5 a `vida` (Solo si el jugador tiene botiquines)
`recogerBotiquin`	`void`	No recibe	Suma 1 a `numeroBotiquines`
`moverDerecha`	`void`	`double d`: Cantidad de pixeles a mover el personaje a la derecha.	Suma d a `posicionX`
`moverIzquierda`	`void`	`double d`: Cantidad de pixeles a mover el personaje a la izquierda.	Resta d a `posicionX`
`moverArriba`	`void`	`double d`: Cantidad de pixeles a mover el personaje hacia arriba.	Suma d a `posicionY`
`moverAbajo`	`void`	`double d`: Cantidad de pixeles a mover el personaje hacia abajo.	Resta d a `posicionY`
`golpear`	`void`	`Enemigo p`: Personaje que recibe el daño	Además de realizar lo que se especifica en la clase padre, debe invocar la evolución del enemigo `p`

Clase Enemigo (Hija / Hereda de Personaje)

NOMBRE	TIPO DATO	CONCEPTO	INICIALIZACIÓN
`evolucionesAplicadas`	`int`	Número de evoluciones aplicadas al enemigo	Debe estar inicializada en 0
`resistencia`	`int`	Parámetro para el cálculo del impacto que recibe el enemigo	Debe estar inicializada en 1

Métodos

NOMBRE	TIPO RETORNO	PARÁMETROS	CONCEPTO
`evolucionar`	`void`	No recibe	Si el enemigo que invoca este método tiene 30 puntos de vida (O menos) y `evolucionesAplicadas` es igual a 0, se suma `20` al atributo `damage`, y se suma 1 a `evolucionesAplicadas`. Si el enemigo que invoca este método tiene 10 puntos de vida (O menos) y `evolucionesAplicadas` es igual a 1, se suma 1 al atributo `resistencia`, y se suma 1 a `evolucionesAplicadas`
`recibirImpacto`	`void`	`double d`: Cantidad de vida a restarle al personaje	Resta $\frac{d}{resistencia}$ a `vida`

PRECISIONES

1. No hay métodos estáticos.
2. Deben existir *getters* y *setters* de todos los atributos de cada clase, estos deben ser escritos en la forma estándar camel case.
3. Si el personaje tiene 0 botiquines, al intentar usar un botiquín **NO** modificará el número de botiquines ni la vida del personaje.

4. La vida de un personaje está entre 0 y 100 (No puede tener una vida con más de 100 puntos, ni vida negativa, el valor final cuando le quitan toda la vida es de 0 puntos).

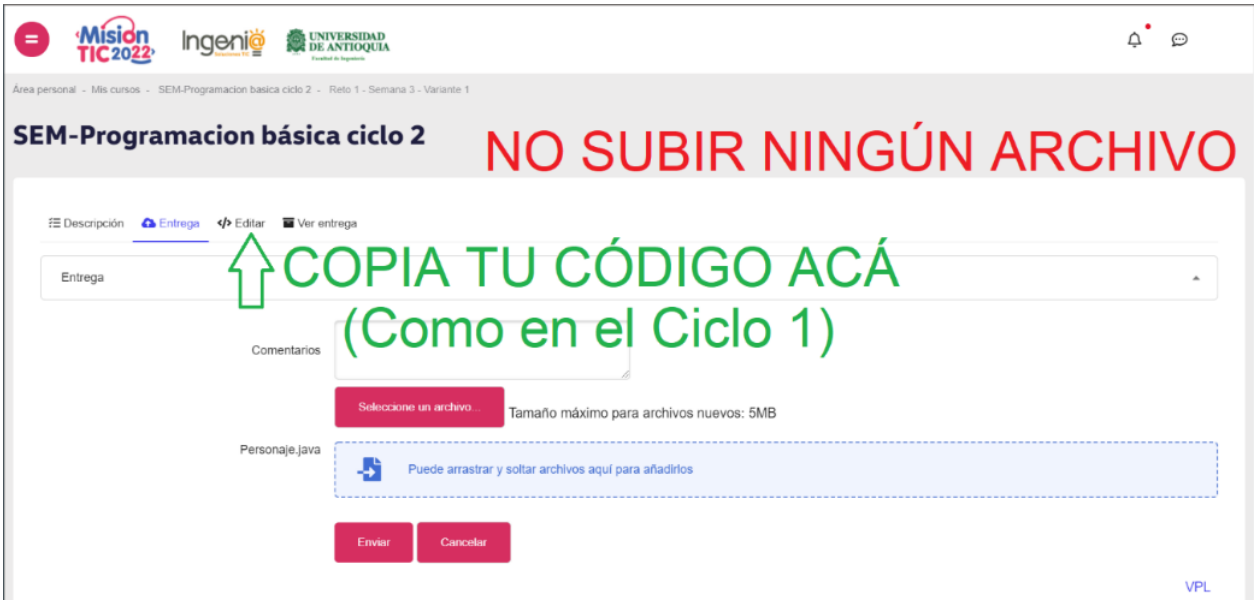
TAREAS

- En el archivo preconstruido en la plataforma Moodle, implementar las clases especificadas en el diagrama de clases, teniendo en cuenta las precisiones dadas por el equipo de Ingeniería de software.
- Los nombres de los métodos y atributos **DEBEN** ser nombrados tal y como aparecen en el diagrama de clases.
- Usted **NO** debe solicitar datos por teclado, ni programar un método `main`, tampoco use `Java Source Package`, usted está solamente encargado de la construcción de la clase.

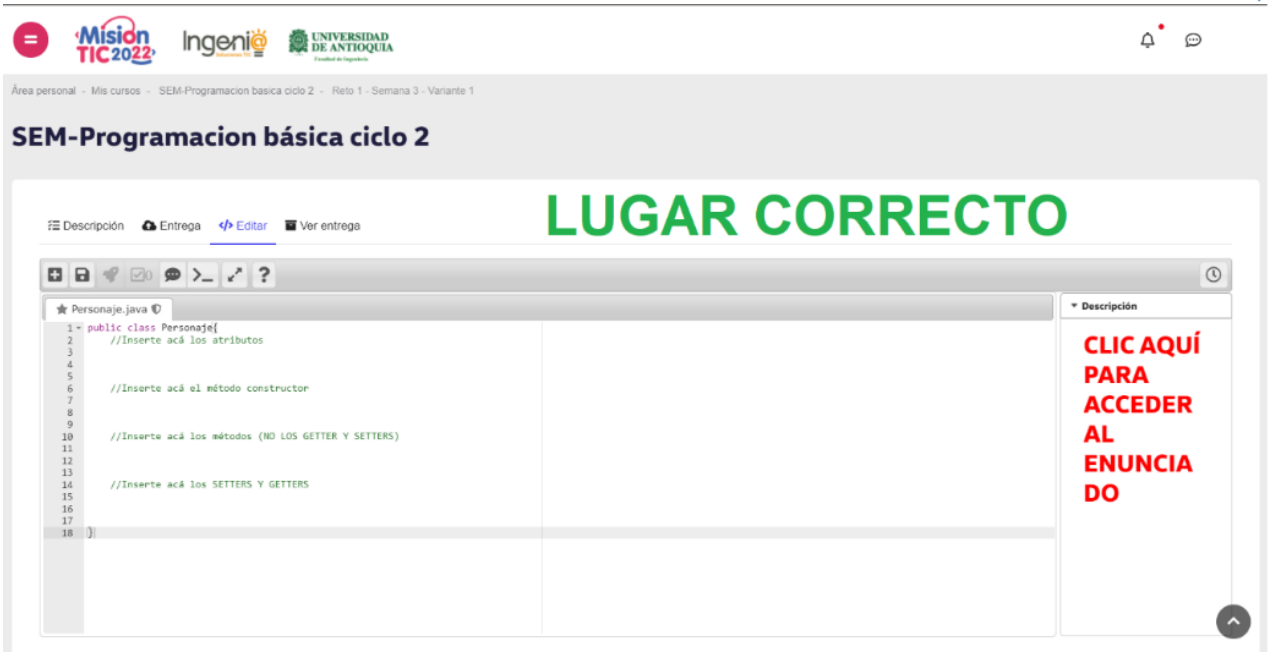
NOTA ACLARATORIA

Usted podrá desarrollar la clase requerida en un IDE como NetBeans, y al final copiar y pegar el código en la herramienta VPL, pero **NO** deberá subir archivos, es decir:

Modo incorrecto:



Modo correcto:



EJEMPLO

1. Se comienza la partida:

```
Jugador j = new Jugador("Explorador", 'm', 0, 5, 100);
Enemigo e = new Enemigo("Bitter", 'f', 0, 0, 80);
```

Jugador		Enemigo	
nombre	"Explorador"	nombre	"Bitter"
sexo	'm'	sexo	'f'
posicionX	0	posicionX	0
posicionY	5	posicionY	0
damage	100	damage	80
vida	100	vida	100
numeroBotiquines	0	evolucionesAplicadas	0
		resistencia	1

2. El jugador golpea al enemigo 2 veces, por lo que la vida del enemigo sería $100 - \frac{2 \times 100}{5} = 60$, el jugador se mueve 10 metros a la derecha y 5 hacia abajo:

```
j.golpear(e);
j.golpear(e);
j.moverDerecha(10);
j.moverAbajo(5);
```

Jugador		Enemigo	
nombre	"Explorador"	nombre	"Bitter"
sexo	'm'	sexo	'f'
posicionX	10	posicionX	0
posicionY	0	posicionY	0
damage	100	damage	80
vida	100	vida	60
numeroBotiquines	0	evolucionesAplicadas	0
		resistencia	1

3. El enemigo golpea al jugador 2 veces, por lo que la vida del jugador sería $100 - \frac{2 \times 80}{10} = 84$, el jugador recoge un botiquin:

```
e.golpear(j);
e.golpear(j);
j.recogerBotiquin();
```

Jugador		Enemigo	
nombre	"Explorador"	nombre	"Bitter"
sexo	'm'	sexo	'f'
posicionX	10	posicionX	0
posicionY	0	posicionY	0
damage	100	damage	80
vida	84	vida	60
numeroBotiquines	1	evolucionesAplicadas	0
		resistencia	1

4. El jugador usa dos veces botiquín, pero solo podrá aumentar en 5 su vida, porque solo dispone de un solo botiquín, y golpea 3 veces al enemigo, por lo que la vida del enemigo sería $60 - \frac{3 \times 80}{10} = 30$ (Esto activa la primera evolución del enemigo, que sube en 20 el poder de daño):

```
j.usarBotiquin();
j.usarBotiquin();
j.golpear(e);
j.golpear(e);
j.golpear(e);
```

Jugador		Enemigo	
nombre	"Explorador"	nombre	"Bitter"
sexo	'm'	sexo	'f'
posicionX	10	posicionX	0
posicionY	0	posicionY	0
damage	100	damage	100
vida	89	vida	30
numeroBotiquines	0	evolucionesAplicadas	1
		resistencia	1

5. El enemigo golpea dos veces al jugador, por lo que la vida del jugador sería $89 - \frac{2 \times 100}{10} = 69$:

```
e.golpear(j);
e.golpear(j);
```

Jugador		Enemigo	
nombre	"Explorador"	nombre	"Bitter"
sexo	'm'	sexo	'f'
posicionX	10	posicionX	0
posicionY	0	posicionY	0
damage	100	damage	100
vida	69	vida	30
numeroBotiquines	0	evolucionesAplicadas	1
		resistencia	1

6. El jugador golpea dos veces al enemigo, por lo que la vida del enemigo sería $30 - \frac{2 \times 100}{10} = 10$ (Esto hace que el enemigo evolucione y aumente en uno la resistencia):

```
j.golpear(e);
j.golpear(e);
```

Jugador		Enemigo	
nombre	"Explorador"	nombre	"Bitter"
sexo	'm'	sexo	'f'
posicionX	10	posicionX	0
posicionY	0	posicionY	0
damage	100	damage	100
vida	69	vida	10
numeroBotiquines	0	evolucionesAplicadas	2
		resistencia	2

7. El jugador golpea tres veces más al enemigo, por lo que la vida del enemigo sería $10 - \frac{3 \times 100}{10 \times 2} = -5$ (Como la vida no puede ser menor que cero, el enemigo queda con 0 puntos de vida), recuerde que cuando se aumenta la resistencia el impacto generado por los golpes recibidos por el enemigo se reduce a la mitad:

```
j.golpear(e);
j.golpear(e);
j.golpear(e);
```

Jugador		Enemigo	
nombre	"Explorador"	nombre	"Bitter"
sexo	'm'	sexo	'f'
posicionX	10	posicionX	0
posicionY	0	posicionY	0
damage	100	damage	100
vida	69	vida	0
numeroBotiquines	0	evolucionesAplicadas	2
		resistencia	2