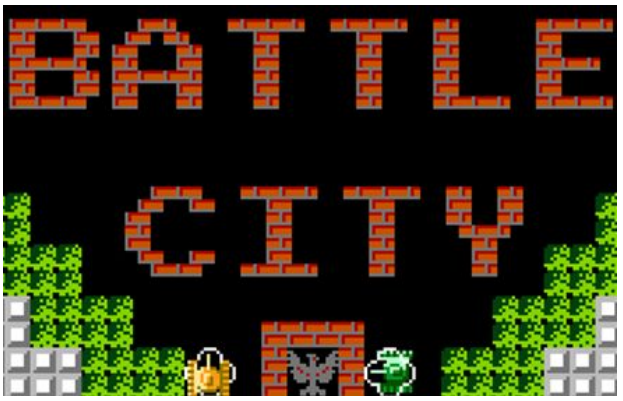
# Recuperatorio

*Tecnología de Programación*

Nombre:_____L.U.:_____Cant.Hojas: _____

## ENUNCIADO

La empresa “El Lechonceto S.A.” quiere realizar una nueva versión del juego de NES “Battle City”. Este juego es un shooter en tercera persona de batalla de tanques.




Los personajes principales del Juego son los Tanques y los Disparos que se realizan entre ellos. Por esta razón el juego los categoriza como Entidades y conoce un conjunto de ellas.

Las entidades tienen un método “recibirDaño(int)” y un metodo “morir()”.

Hay dos tipos esenciales de Tanques: el jugador y los tanques enemigos. Todos los tanques tienen un atributo “vida” que inicia en 100, un atributo “velocidad base” cuyo valor depende de cada tipo de tanque y un atributo calculado “velocidadFinal():int”. Adicionalmente, todos los tanques se encuentran en dos posibles estados: Sano y Dañado. Sano es el estado que se utiliza por defecto en los tanques. Los estados son quienes determinan la “velocidad final” y controlan la recepción del daño por lo que los tanques siempre delegan esta responsabilidad a los mismos.

Cuando la vida de los tanques se reduce a menos de 30 puntos pasan al estado “Dañado”.

La “velocidad final” de los tanques resulta de un cálculo que realizan los estados de la siguiente forma:

- En el estado “Sano” la “velocidad final” es la velocidad base del tanque más 10
- En el estado “Dañado” la “velocidad final” es la velocidad base del tanque más 5

Cada tanque tiene un método “mover(int)” que permite realizar movimientos en distintas direcciones.

El tanque del jugador tiene una velocidad base de 5.

Existen tres tipos de tanques enemigos:

- Los básicos tienen una velocidad base de 4
- Los rápidos tienen una velocidad base de 7
- Los acorazados tienen una velocidad base de 4

La diferencia de los tanques acorazados es que siempre reciben la mitad del daño.

Los tanques enemigos tienen un método “mover()” que, utilizando la “velocidad final”, mueven al tanque dependiendo de su inteligencia. Existen dos tipos de inteligencias: la primera es “Aleatoria” y la segunda es más “Incisiva” e intenta acercarse al jugador en todo momento. Se planean agregar más tipos de inteligencias en el futuro. Todos los tanques enemigos comienzan con la inteligencia

"Aleatoria". Los tanques básicos cuando tienen menos de 20 puntos de vida pasan a la inteligencia "Incisiva".

Cuando la vida de un tanque llega a 0 se llama al metodo morir(). En cambio, los disparos siempre llaman a su método "morir()" cuando reciben daño.

Todos los tanques tienen un Arma que es un *Abstract Factory* para producir disparos (único producto) y existe un disparo específico por cada tipo de tanque.

Los disparos tienen un atributo calculado "daño():int" que retorna el daño que provoca el disparo. El disparo del tanque jugador, al igual que el disparo del tanque básico hacen 10 puntos de daño. El disparo del tanque rápido hace 8 puntos de daño y el disparo del tanque acorazado hace 15 puntos de daño.

En el estado "Dañado" cualquier tanque al querer disparar produce un nuevo disparo con daño 0.

El juego controla todas las colisiones entre las entidades utilizando el patrón de diseño *Visitor*. Existe un Visitor por cada tipo concreto de entidad que permite realizar las interacciones de manera acorde. Ningún disparo enemigo puede dañar a un enemigo. Cuando el jugador y el enemigo colisionan ambos mueren automáticamente. El jugador recibe daño de los disparos de los diferentes enemigos y los enemigos reciben daño de los disparos del jugador.

## EJERCICIOS

1. Diseñe el juego "Battle City" mediante un diagrama UML reducido.
2. Realice el diagrama expandido del "visitor acorazado" y del enemigo "rápido" y de sus clases "padre". No olvide indicar con * lo Abstracto, con + los métodos que implementan a uno Abstracto y con ++ los que redefinen un método. Indique qué métodos se heredan y de qué clase *.
3. Realice un diagrama de interacción para el "visitor jugador" recibiendo el disparo de un tanque acorazado y todos los diagramas que se desencadenan. Si es necesario, divida el diagrama en varios más pequeños. No diagrame el interior del método "morir()".
4. Implemente en Java el método "disparar():Disparo" de la clase "tanque rápido" y el disparo de este tanque. Implemente las clases necesarias para que la operación se lleve a cabo.
5. Implemente en Java el método del tanque básico para recibir daño. Tenga en cuenta que se debe implementar los métodos que sean necesarios de las clases que correspondan para que la operación se lleve a cabo correctamente.

PRESTE ESPECIAL ATENCIÓN A LA MODULARIZACIÓN DE MÉTODOS Y LA NO REESCRITURA DE CÓDIGO

NO IMPLEMENTE NI DIAGRAME EL DESARROLLO DEL MÉTODO "morir()" NI TAMPOCO COMO EL JUEGO ENCUENTRA LAS COLISIONES