Se trata de desarrollar una aplicación Java, denominada **PROG07_Tarea** que permita gestionar varios tipos de **elementos,** que suponemos nos podrían valer para programar un juego... Imaginad un elemento como cualquier objeto que pueda aparecer en un juego y que pueda tener interacción con otros elementos.

Se debe desarrollar una estructura de clases e interfaces para gestionar los elementos y sus posibles derivados.

La clase **Elemento** será una clase **abstracta** con los siguientes atributos y métodos.

· **id** // identificador del objeto (int)

· **posicionX** **//** coordenadas x e y de un elemento (float,float)

· **posicionY** // marcan la posición del elemento en el escenario.

· **nombre**

· **void posicionar(float x,float y)** //método **protected** que establece las coordenadas de un elemento.

· **boolean esPintable()** // método **abstracto** que retornará true o false dependiendo de si un elemento es pintable en pantalla o no.

· setters y getters ()

De la clase Elemento tendremos dos Clases herederas.

La clase **ElementoFijo**. Esta clase será instanciable (también llamada 'concreta', o sea clase normal o no abstracta). No aportará ningún atributo ni método nuevo (salvo un constructor con todos los parámetros). Se describe más adelante el comportamiento de su método esPintable().

La clase **ElementoMovil**. Que será a su vez una **clase abstracta** que incorporará el método abstracto:

boolean mover(float incrementoX, float incrementoY), cuyos parámetros se usarán en las herederas para sumarlos a la posición del Elemento.

De la clase **ElementoMovil** tendremos a su vez dos Clases herederas instanciables.

Clase **Heroe,** que incorpora los siguientes atributos y métodos:

· **fuerza** // número entero que vale de 0 a 9.

· Método; **boolean colisiona(Elemento p)** que retornará true si la posición del objeto es menor a 0,5 en ambas coordenadas (en valores absolutos) respecto a la posición de p.

· Constructor con todos los parámetros, setters y getters () necesarios de añadir.

Se describe más adelante el comportamiento de su método esPintable().

Clase **Fantasma,** que incorpora los siguientes atributos y métodos:

· **vivo** // booleano que indica si un fantasma está 'vivo' (activo en el juego) o no.

· Constructor con todos los parámetros, setters y getters () necesarios de añadir.

Se describe más adelante el comportamiento de su método esPintable().

Metodo toString():

Todas las clases implementarán el método toString() mostrando todos los atributos que tenga la clase. Se debe hacer uso del método toString() de la Superclase. La cadena retornada estará formateada con String.format.

Método esPintable():

Para que en las tres clases instanciables (ElementoFijo, Heroe y Fantasma) el método esPintable() retorne un true es necesario que las coordenadas del elemento no sean inferiores a 0,5 ni superiores al (Tamaño de la pantalla – 0,5) Suponemos una pantalla cuadrada por simplicidad. Utiliza constantes definidas en la clase Elemento por si hay que cambiar estos valores en tiempo de compilación.

Además, para que un fantasma sea pintable también tiene que estar vivo.

En cualquier otro caso el método retornará false.

Interfaz Pintable().

Tendremos también, en el mismo paquete, un interfaz, **Pintable**, con un único método llamado **pintar(),** que de momento, hasta que no implementemos el aspecto gráfico, solo nos mostrará por consola datos del objeto dependiendo de a que clase pertenezca el mismo.

Las tres clases instanciables (ElementoFijo, Heroe y Fantasma) deben implementar el interfaz Pintable y en los tres casos, **si el elemento es en ese momento pintable**, el método pintar() mostrará por consola la siguiente información.

"PINTANDO; " + id + nombre + coordenadas + Clase a la que pertenece el objeto.

En el caso de un héroe también se indicará su fuerza "FUERZA " + fuerza.

Detalles y Mejoras.

En las clases pertinentes se debe redefinir el método posicionar() o mover() para que no actúen si el elemento está muerto o si tiene fuerza 0.

Cuando un héroe colisiona con un fantasma pierde 5 puntos de fuerza y el fantasma muere. Cuando un héroe colisiona con un elemento fijo pierde 1 punto de fuerza.

Cuando un héroe colisiona con otro héroe ambos ganan 1 punto de fuerza (utiliza constantes como atributos de la clase, por si luego tenemos que ajustar estos valores)

…….

Programa Principal:

En un paquete distinto realizaremos un Principal.. En este caso no usaremos ningún tipo de menú, sino una secuencia de instrucciones para probar las clases creadas. **No se pedirán datos al usuario**. Entre otras cosas se requiere:

· Se deben crear al menos, un elemento fijo, dos fantasmas y dos héroes.

· Todos se deben almacenar en un array de Elementos.

· Mueve y también provoca colisiones entre distintos Elementos para comprobar los métodos creados.

· Muestra recorriendo el array los elementos de vez en cuando para ver cómo cambian.

· Pinta recorriendo el array los elementos de vez en cuando para ver cuales se pintarían.

Observa si se aplica bien el polimorfismo y la ligadura dinámica cuando Muestres y Pintes el array de objetos.

La nota de este apartado dependerá de lo bien que queden probadas las clases y sus métodos.

Además del programa podrás escribir también un **informe** con todas las consideraciones oportunas que se necesiten para entender cómo has realizado la tarea. De no hacerlo, el código del Principal deberá estar bien comentado indicando los pasos que se siguen para probar los métodos.