MODULO 1: DISEÑO DE UN PERSONAJE DE VIDEOJUEGO

A lo largo de las siguientes clases de prácticas vamos a construir una clase para representar un personaje de videojuego. Inicialmente diseñaremos un personaje muy simple, con una serie de propiedades (nombre, puntos, nivel de energía, posición, orientación…), y la facultad de realizar una serie de operaciones (avanzar, girar, ganar/perder puntos, coger bonos,…). Estas acciones podrán modificar las propiedades de nuestro personaje.

Mediante un sencillo interfaz de usuario en modo texto (que no necesitará desarrollar el alumno de momento), el usuario del programa podrá jugar con su personaje. El interfaz únicamente pedirá al usuario que realice una operación y mostrará el resultado por pantalla.

Un elemento fundamental en la programación de videojuegos es el diseño de **escenarios,** en los que se sitúan los distintos personajes. Más adelante añadiremos un **escenario** a nuestra aplicación, de momento para simplificar solo nos ocuparemos del diseño del personaje.

En la clase de hoy vamos a comenzar modelando en UML una clase **Personaje.** Usaremos la herramienta gratuita **UMLPad**, disponible en la web de la asignatura, para diseñar el esqueleto de nuestro personaje, que tendrá:

- Un **estado (propiedades)** definido por una serie de atributos de distintos tipos, que iremos aumentando progresivamente en número y complejidad. Todos los atributos serán privados.

Al menos se definirán: nombre, número de vidas, puntos, nivel de energía, posición y orientación. La posición quedará definida por dos coordenadas (en dirección Norte y Este), de tipo entero. La orientación queda determinada por un carácter ('N' = norte, 'S' = sur, 'E' = este, 'O' = oeste). Usaremos además un atributo lógico para indicar si el personaje está vivo. Cuando el nivel de energía desciende a un valor menor o igual que cero, el personaje pierde una vida, y cuando agota todas sus vidas y se vuelve a quedar sin energía, el personaje se muere.

- Un **comportamiento** definido por los métodos que iremos incluyendo en la clase Personaje (avanzar, girar, ganar/perder puntos, coger bonos,…).

Una vez que tengamos el diseño UML, crearemos un nuevo proyecto con Jbuilder, por ejemplo **modulo1.jpx**, y le añadiremos una nueva clase, **Personaje** (en este caso desactivaremos la opción "Generar método `main`"). Ahora se trata de traducir a lenguaje Java nuestro diseño en UML, pero no podemos hacerlo libremente, sino de manera que se adapte al interfaz de usuario proporcionado. Este interfaz está en el fichero **clase3.zip**, que se puede descargar de la página web de la asignatura. Se aconseja extraer su contenido, **InterfazUsuario.java,** a la carpeta **Modulo1\src\modulo1**. De esta manera, será muy sencillo añadir a nuestro nuevo proyecto, **modulo1.jpx,** la clase **InterfazUsuario.java,** por ejemplo yendo al menú *Proyecto/Añadir/Archivos/Paquetes/Clases...*

En este momento deberíamos tener en el proyecto dos ficheros fuente con código en lenguaje Java:

1. **InterfazUsuario.java**: Consiste en un sencillo interfaz de usuario en modo texto que permite utilizar la clase **Personaje**. Todavía no se ha visto suficiente materia en las clases teóricas para comprender su código. Esta clase tiene un atributo de tipo **Personaje**, y contiene el método **`main`,** por el cual comienza la ejecución del proyecto. Presenta un menú para seleccionar operaciones muy sencillas, lee la opción correspondiente del usuario, e invoca el método de la clase **Personaje** que realiza la operación seleccionada.
2. **Personaje.java:** Creada por el alumno, inicialmente es solo un esqueleto de la clase que vamos a desarrollar. Para empezar vamos a añadir los atributos y métodos de nuestro modelo en UML. Al ser privados, cada alumno puede elegir los nombres y tipos de sus atributos. Para los métodos privados también se puede elegir nombre y signatura. Sin embargo los métodos

públicos deben obligatoriamente adaptarse al interfaz que los utiliza. Los siguientes métodos son invocados desde el interfaz, y por tanto sus signaturas deben adaptarse a las indicadas:

- Un constructor: Crea el Personaje y asigna valores iniciales a los atributos. Su signatura es:

```
public Personaje(String nombre, int vidas);
```

Antes de invocar al constructor para crear el personaje, el interfaz pedirá por teclado un nombre para el personaje y un valor entero para su número de vidas. Como posición inicial se usará el origen de coordenadas (0,0). La orientación de salida será 'N'. El personaje se crea con cero puntos, y deberá tantearse un valor para su nivel inicial de energía (usar un identificador constante).

- Método para avanzar:

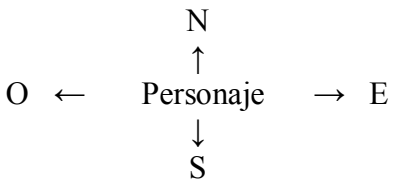
```
public void avanzar(int dist);
```

Este método modifica las coordenadas del personaje, que se moverá `dist` unidades en la dirección en que está orientado. Si la orientación es norte o sur cambiará su coordenada norte, pero no su coordenada este. Si su orientación es este u oeste entonces cambiará su coordenada este pero no la norte. Además, el movimiento implica un consumo de energía por el personaje.

- Método para girar:

```
public void girar();
```

Al girar, cambia la orientación del personaje, que siempre gira en sentido horario.



De la figura anterior se deduce como cambia la orientación al girar:

N → E S → O E → S O → N

- En algunas posiciones hay bonos de puntos, de energía o vidas extra, que pueden ser buenos (suman) o malos (restan). Si lo hay, el personaje puede coger el bono de su posición. La signatura del método correspondiente es:

```
public void cogerBono(){
```

Puesto que aun no tenemos escenario, necesitamos que el usuario nos diga si en una posición había o no bono y de qué tipo. Esta comunicación usuario-interfaz se hace mediante otro atributo privado del usuario, para acceder al cual el interfaz usa este método selector:

```
public String toString();
```

Una vez eliminados los errores sintácticos, conseguiremos que nuestro proyecto compile y se ejecute, pero todavía hay que comprobar que funciona correctamente. ¿Qué pasa cuando el nivel de energía toma valores negativos? ¿No hay límite a los movimientos del personaje? Al coger un bono, ¿desaparece de su posición?

En sesiones posteriores iremos añadiendo complejidad a nuestro personaje. Para poder probar el funcionamiento del personaje, conviene incluir las cabeceras de todos los métodos indicados, aunque dejemos el cuerpo vacío.

Y ahora .... ¡a trabajar completando Personaje.java!