Trabajo Práctico 3
Programación Orientada a Objetos

Paradigmas de Lenguajes de Programación – 2° cuat. 2008

Fecha de entrega: 27 de noviembre

Gotta catch 'em all

Pokemones

Un Pokemon es un bichito agresivo al que le gusta pelearse con otros Pokemones.

Todo Pokemon tiene una cierta energía vital, que se representa con un número natural. Si en alguna circunstancia la energía de un Pokemon disminuye a un valor de 0 o menos, se considera que el Pokemon ha muerto. La energía inicial de un Pokemon es siempre un valor fijo, puede fijarse por ejemplo en 100.

Cada Pokemon tiene también una cierta fuerza, que se representa con un entero positivo. La fuerza de un Pokemon es un valor que depende de su raza.

Asimismo, cada Pokemon tiene un cierto valor de experiencia, que representa la cantidad de veces que mató a otro Pokemon.

El impacto de un Pokemon se calcula como el producto entre su fuerza y su experiencia.

Cuando un Pokemon golpea a otro Pokemon, la energía del segundo disminuye en una cantidad igual a la fuerza de impacto del primero.

Ejercicio 1. Implementar la clase Pokemon, que reconozca los siguientes mensajes (puede implementar también otros):

Métodos de clase:

- `new` — crea un nuevo Pokemon.

Métodos de instancia:

- `energia` — devuelve la energía vital del Pokemon.
- `vive` — devuelve `true` si el Pokemon está vivo y `false` si no.
- `fuerza` — devuelve la fuerza actual del Pokemon.
- `experiencia` — devuelve la experiencia actual del Pokemon.
- `golpearA: otroPokemon` — golpea al otro Pokemon.

Notar que no tiene mucho sentido considerar instancias directas de la clase Pokemon, porque el valor de la fuerza está definido sólo para Pokemones de alguna raza en particular.

Razas de Pokemones

Ejercicio 2. Implementar la clase Pikachu, una raza de Pokemon cuya fuerza es constantemente 5.

Ejercicio 3. Implementar la clase Raichu, que es una evolución de Pikachu (léase: clase derivada) con cola eléctrica. Cuando un Raichu golpea a otro Pokemon, el segundo recibe, además de la fuerza de impacto, el voltaje de la cola[1]. Las instancias deben reconocer el siguiente mensaje:

- `voltaje: v` — establece la fuerza de voltaje de la cola.

Ejercicio 4. Implementar la clase Deoxys. La fuerza de los Pokemones de esta raza es siempre 1. Un Deoxys tiene además un campo de fuerza que atenúa los impactos que recibe. El poder del campo de fuerza se deduce de la fuerza de cualquier impacto que reciba este tipo de Pokemones. Las instancias deben reconocer el siguiente mensaje:

- `poderCampoDeFuerza: c` — establece el poder del campo de fuerza.

Ejercicio 5. Implementar la clase Bulbasaur. La peculiaridad de los Pokemones de esta raza es que, cada vez que reciben un impacto, su energía disminuye de la manera habitual, pero su fuerza aumenta en un número igual al del impacto. La fuerza inicial de los Pokemones de esta raza es 0.

Pelea de Pokemones

Ejercicio 6. Implementar el siguiente mensaje en la clase Pokemon:

- `pelearAMuerteContra: p` — lucha contra `p` hasta que alguno de los dos Pokemons muere.

Una lucha a muerte se desarrolla del siguiente modo:

1. El Pokemon que recibe el mensaje golpea.
2. El Pokemon argumento del mensaje golpea.
3. Se repite desde el primer paso mientras los dos Pokemones viven.

Exportación e importación de paquetes

Para poder trabajar controladamente en el entorno Visual Works, ubicarse en la ventana "Packages" (F5) Crear un nuevo paquete desde el menú Package ↦ New Package. Llamarlo, por ejemplo, "Mini TP". Aconsejamos crear todas las clases nuevas dentro de ese paquete, y no modificar las definiciones de las clases ya existentes para simplificar la exportación.

Una vez finalizada la implementación, ubicándose sobre el paquete, se puede importar y exportar todo su contenido a un archivo `.st` desde las opciones Package ↦ File in y Package ↦ File out ↦ Package.

[1] Notar que la suma del voltaje y la fuerza a su vez se multiplica todo por la experiencia.

Pautas de entrega

Se debe entregar un archivo `.st` (en formato XML) con la implementación de las clases pedidas. Cada método debe contar con un comentario donde se explique su funcionamiento. El código debe poder ser ejecutado en `Visual Works`[2]. No es necesario entregar un informe sobre el trabajo, alcanza con que el código esté adecuadamente comentado.

Los objetivos a evaluar en la implementación de las clases son:

- Corrección.
- Declaratividad.
- Uso adecuado de conceptos de programación orientada a objetos: herencia, polimorfismo, `super` y `self`.

[2]Existen versiones de `Visual Works` para Windows, Linux y Mac, entre otros. Es importante **no** utilizar otros entornos ya que la forma en la que trabajan con bloques de código difiere.