# Ejercicios Java - Unidad 6

*Clases, Objetos y Excepciones*

## Instrucciones para resolver los ejercicios

- Resuelve los ejercicios en el proyecto creado en repositorio “Ejercicios 2ª Evaluación”.
- Las clases definidas para cada ejercicio se han de definir en el paquete `unidad6`.
- Cada vez que se resuelva un ejercicio se realizará un *Commit and Push* con el mensaje “Ejercicio *n* de la unidad 6 resuelto”, donde *n* será el número de ejercicio.
- No es obligatorio resolver y confirmar los ejercicios en el orden de numeración.

## Ejercicio 1: Mascota Virtual

Escribir un programa Java que permita la creación y mantenimiento de un número indefinido de mascotas virtuales según las especificaciones siguientes:

- Cada mascota tiene un nombre y un nivel de energía representado por un número entero que determina su estado de salud. Se crea con un nivel de energía inicial de 20 unidades y el nombre que le asigne su dueño.
- El nivel de energía aumenta en 5 unidades cuando comen, en 2 unidades cuando duermen y disminuye en 3 unidades cuando hacen ejercicio.
- Se pueden poner enfermas si se da cualquiera de las circunstancias siguientes:
    - Su nivel de energía supera las 50 unidades por exceso de comida o descanso.
    - Su nivel de energía baja de las 5 unidades por exceso de actividad física.
    - La comida le sienta mal, lo que ocurre de forma aleatoria con una probabilidad de 3 sobre 10 bajando su nivel de energía a 10 unidades.
- Las mascotas enfermas no deben realizar ninguna actividad hasta que la medicación recetada por el veterinario las cure. De no seguir esta indicación se agravará su estado de enfermedad en una unidad cada vez que realice una actividad.
- La curación restablecerá el nivel de energía a su estado inicial.
- Después de cada actividad, las mascotas virtuales comunicarán a su dueño uno de los tres estados de ánimo siguientes:
    - Alegría cuando está sana y a más de 3 unidades de enfermar.
    - Apatía cuando esta sana, pero a menos de 3 unidades de enfermar.
    - Malestar cuando cae enferma.

El manejo de la aplicación se llevará a cabo a través de una interfaz de usuario simple basada en una consola de texto en la que se introducen comandos a través de línea de comando para realizar las acciones siguientes con la sintaxis que se indica:

- Crear una mascota:
  `> crear nombre`
  Si ya existe una mascota con ese nombre, se mostrará un mensaje de error.
- Alimentar a una mascota:
  `> comer nombre`
- Hacer ejercicio:
  `> ejercicio nombre`
- Dormir a una mascota:
  `> dormir nombre`

- Curar a una mascota:
  `> curar nombre`
- Finalizar el programa:
  `> salir`

Después de las acciones de comer, hacer ejercicio y dormir, se mostrará en la consola una onomatopeya asociada al estado de ánimo comunicado por la mascota. Por ejemplo, un ronroneo para mostrar alegría, un gemido para mostrar apatía o un quejido para mostrar malestar.

Si no es posible que la mascota coma, duerma o haga ejercicio por estar enferma, o curarla porque está sana, o no existe una mascota con el nombre especificado, se mostrará en la consola el mensaje correspondiente.

Si el nivel de energía de una mascota es inferior a cero unidades o supera las 55 unidades, se producirá su fallecimiento virtual dejándola a merced del recolector de basura para que proceda a su incineración. Este hecho trágico se anunciará solemnemente en la consola.

Para el desarrollo de esta aplicación se habrá de tener en cuenta que cuando el presupuesto lo permita se producirá una actualización de la misma cuyo funcionamiento estará basado en una interfaz gráfica de usuario. Por tanto, se ha de procurar que para llevar a cabo dicha actualización solo haya que modificar la clase principal del programa.

## Ejercicio 2: Torneo pokemon

Se desea crear un programa que simule el desarrollo de torneos de pokemons en los que participan una serie de entrenadores.

Cada entrenador se conoce por un nombre único y se presenta al torneo con una colección de pokemons con el objetivo de conseguir la mayor cantidad posible de insignias, ya que el vencedor del torneo será el entrenador que consigua el mayor número de insignias.

Para simular los torneos, el programa recibe a través de la entrada estándar los datos relativos al desarrollo de cada torneo que tendrá que procesar para determinar quíen es el vencedor. Las especificaciones de entrada son las siguientes:

- Cada entrenador comienza el torneo con cero insignias.
- Se recibirán un número indeterminado de líneas en la entrada estándar que van a contener información acerca de un pokemon y del entrenador que lo ha capturado con el formato: ***<nombre del entrenador> <nombre del pokemon> <elemento fundamental del pokemon> <salud del pokemon>***. Los tres primeros datos estará formádos únicamente por caracteres alfabéticos. El último será un número entero.
- A estas líneas les sigue una línea con la plabra “torneo” que marca el comienzo del torneo.
- A esta línea le sigue un número indeterminado de líneas, cada una conteniendo el nombre de uno de los elementos fundamentales, de nuevo formádo únicamente por caracteres alfabéticos. Cada una de estas líneas representa una ronda del torneo en la que cada entrenador obtiene uno de los dos resultados siguientes:
  - Si tiene al menos un pokemon ligado al elemento leido recibe una insignia.
  - En caso contrario, todos sus pokemons perderán 10 puntos de salud. Aquellos que se queden con cero o menos puntos de salud morirán y serán eliminados de la colección del entrenador.

- El torneo finaliza cuando se lee una línea que contiene la palabra “fin”. En ese momento se mostrarán los datos del vencedor: nombre, número de insignias y número de pokemons que han sobrevivido.
- **OPCIONAL (son necesarios conocimientos que se imparten en la unidad 7):** mostrar la lista de entrenadores ordenada por número de insignias en orden ascendente, y en caso de empate, por número de pokemos supervivientes en orden ascendente.

Se ha validar que la entrada de datos se ajusta las especificaciones dadas. En caso contrarío se abortará la lectura de los datos de entrada y por extensión el torneo, mostrando el mensaje de error correspondiente.

Si se utiliza un objeto *Scanner* para la lectura de los datos de entrada se puede llevar a cabo la validación de la entrada de datos de una de las dos formas siguientes:

- Utilizando métodos *hasNext…* antes de proceder a la lectura de los datos con los métodos next.
- Capturando las excepciones que lanzan los métodos *next…* cuando el dato leido no se ajusta a lo especificado.

## Ejemplos:

| Input | Output |
|---|---|
| Ash Charizard fuego 100<br>Brock Squirtle agua 38<br>Ash Pikachu electricidad 10<br>torneo<br>fuego<br>electricidad<br>fin | Ash 2 2<br>Brock 0 1 |
| Misty Blastoise agua 18<br>Clemont Pikachu electricidad 22<br>Millo Kadabra psíquico 90<br>torneo<br>fuego<br>electricidad<br>fuego<br>End | Clemont 1 1<br>Millo 0 1<br>Misty 0 0 |