**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: o Su nivel de energía supera las 50 unidades por exceso de comida o descanso.
  + o Su nivel de energía baja de las 5 unidades por exceso de actividad física.
  + o 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: o Alegría cuando está sana y a más de 3 unidades de enfermar.
  + o Apatía cuando esta sana, pero a menos de 3 unidades de enfermar.
  + o 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.

## Solución VERO

**package** unidad6;

**import** java.util.ArrayList;

**import** java.util.Scanner;

**public** **class** Mascota {

//atributos

**private** String nombre;

**private** Integer energia = 40;

**private** **boolean** vivo = **true**;

**private** **boolean** enferma = **false**;

//constructor

**public** Mascota(String nombre) {

**this**.nombre = nombre;

}

//metodos

**public** String getNombre() {

**return** nombre;

}

**public** **void** setNombre(String nombre) **throws** Exception {

**this**.isVivo();

**this**.nombre = nombre;

}

**public** Integer getEnergia() **throws** Exception {

**try** {

**this**.isVivo();

} **catch** (Exception e) {

System.***err***.println("getEnergia: " + e.getMessage());

}

**return** **this**.energia;

}

**private** **boolean** isVivo() **throws** Exception {

**if** (**this**.vivo)

**return** vivo;

**else**

**throw** **new** Exception("isVivo: La mascota no existe");

}

**private** **void** setVivo(**boolean** vivo) **throws** Exception {

**this**.isVivo();

**this**.vivo = vivo;

}

**public** **void** setEnergia(**int** energia) **throws** Exception {

**this**.isVivo();

**this**.energia = energia;

}

**private** **boolean** isEnferma() {

**return** enferma;

}

**private** **void** setEnferma(**boolean** enferma) {

**this**.enferma = enferma;

}

**private** **void** comprueba() **throws** Exception {

**this**.isVivo();

**if** (**this**.energia > 55 || **this**.energia < 0) {

**this**.setVivo(**false**);

**this**.energia = **null**;

**this**.nombre = **null**;

**throw** **new** Exception("La mascota muere");

}

**if** (**this**.energia > 50 || (**this**.energia < 5 && **this**.energia > 0)) {

**this**.setEnferma(**true**);

**throw** **new** Exception("La mascota está enferma");

}

}

**private** **void** aleatorio() {

**int** n = (**int**) Math.*round*(Math.*random*()\*10);

**if** (n < 4)

**this**.energia -= 10;

}

**public** **void** come(){

**try** {

**this**.isVivo();

**this**.energia += 5;

aleatorio();

comprueba();

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

**public** **void** duerme() {

**try** {

**this**.isVivo();

**this**.energia += 2;

comprueba();

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

**public** **void** ejercita() {

**try** {

**this**.isVivo();

**this**.energia -= 3;

comprueba();

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

**public** **void** cura(){

**try** {

**this**.isVivo();

**if** (**this**.enferma)

**this**.energia = 20;

**this**.enferma = **false**;

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

**public** String feedback() {

**if** (**this**.enferma) {

**return** "Buu... snif =Y.Y=";

} **else** **if** (**this**.energia>8 && **this**.energia < 47) {

**return** "Rrrrrr =^.^=";

} **else** {

**return** "Ñee =>.<=";

}

}

@Override

**public** String toString() {

**return** **this**.nombre + " tiene " + **this**.energia + " puntos de energía. Enferma: " + **this**.enferma;

}

**public** **static** **void** comando(String cmd, String nombre, ArrayList<Mascota> mascotas) {

**try** {

cmd = cmd.toLowerCase();

**switch** (cmd) {

**case** "crear":

Mascota nuevaMascota = **new** Mascota(nombre);

mascotas.add(nuevaMascota);

System.***out***.println("Creada mascota: " + mascotas.get(mascotas.size()-1).getNombre());

**break**;

**case** "comer":

*encuentra*(nombre,mascotas).come();

//System.out.println(encuentra(nombre,mascotas));

System.***out***.println(*encuentra*(nombre,mascotas).feedback());

**break**;

**case** "ejercicio":

*encuentra*(nombre,mascotas).ejercita();

//System.out.println(encuentra(nombre,mascotas));

System.***out***.println(*encuentra*(nombre,mascotas).feedback());

**break**;

**case** "dormir":

*encuentra*(nombre,mascotas).duerme();

//System.out.println(encuentra(nombre,mascotas));

System.***out***.println(*encuentra*(nombre,mascotas).feedback());

**break**;

**case** "curar":

*encuentra*(nombre,mascotas).cura();

//System.out.println(encuentra(nombre,mascotas));

System.***out***.println(*encuentra*(nombre,mascotas).feedback());

**break**;

**case** "salir":

**break**;

**default**:

**throw** **new** Exception("Comando no válido");

}

} **catch** (Exception e) {

System.***err***.println(e.getMessage());

}

}

**public** **static** Mascota encuentra(String nombre, ArrayList<Mascota> mascotas) **throws** Exception {

Mascota localizada = **null**;

**for**(Mascota x: mascotas) {

**if** (x.getNombre() != **null**)

**if** (x.getNombre().equalsIgnoreCase(nombre))

localizada = x;

}

**if** (localizada == **null**)

**throw** **new** Exception("No existe la mascota");

**return** localizada;

}

**public** **static** **void** main(String[] args) **throws** Exception {

// **TODO** Auto-generated method stub

ArrayList<Mascota> mascotas = **new** ArrayList<Mascota>();

Scanner in = **new** Scanner(System.***in***);

String teclado = in.nextLine();

**while** (!teclado.equalsIgnoreCase("salir")) {

String palabras[] = teclado.split(" ");

**try** {

*comando*(palabras[0], palabras[1], mascotas);

} **catch** (ArrayIndexOutOfBoundsException e) {

System.***err***.println("Comando incorrecto");

} **catch** (Exception e) {

System.***err***.println(e.getMessage());

}

teclado = in.nextLine();

}

**for**(Mascota x: mascotas) {

System.***out***.println(x);

}

in.close();

}

}