Documentación Tarea 1
Mascota Virtual

Versión v1.0

Grupo 9

23 de abril de 2024

1. Diagrama UML del desarrollo de la tarea

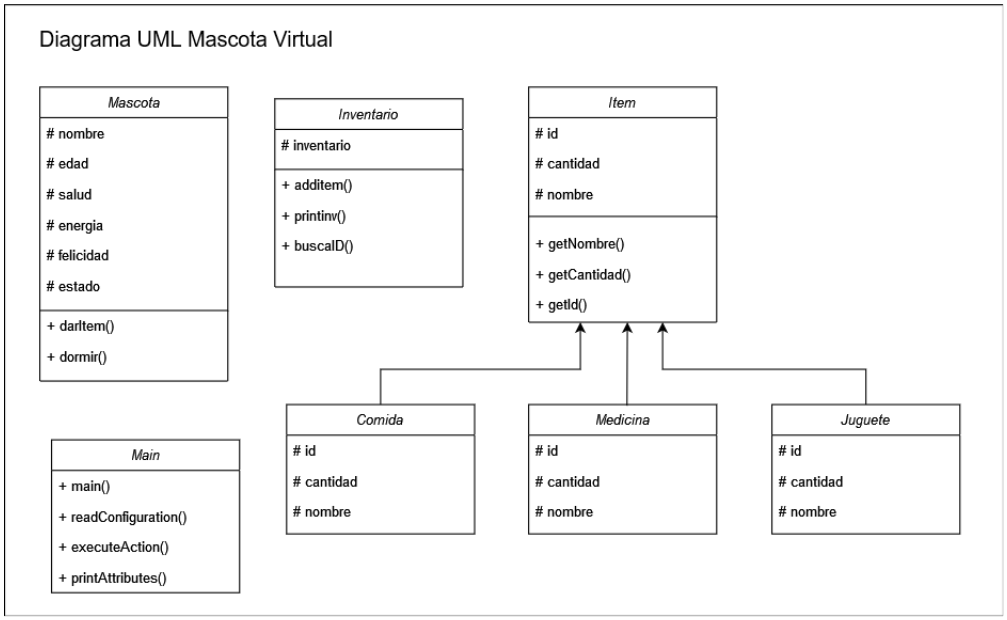

Figura 1: Diagrama UML de la Mascota virtual

2. Explicación solución de la tarea

2.1. Etapa 1

El objetivo de esta etapa era obtener una clase Mascota la cual muestre en pantalla sus atributos actuales.
Para lograr este objetivo se creo un archivo Mascota.java, en donde se creo la clase Mascota con todos sus respectivos atributos, su constructor y el método para imprimir los atributos en pantalla. Además crearon los archivos Estado.java, el cual describe los estados de la mascota virtual y el archivo Main.java, el cual sirve como el punto de inicio de la ejecución del programa. En el archivo Mascota se concentró la parte fundamental de esta etapa, ya que contiene los métodos encargados de imprimir los atributos de la mascota.

2.2. Etapa 2

En esta etapa el objetivo era crear ítems los cuales interactúen con la mascota. Un requisito de esta etapa era crear clases heredadas de la clase ítem para describir los distintos ítems que usa la mascota virtual.
Para esta parte se agrego el archivo Item.java al programa. También esto se edito la clase Mascota y se agregaron los métodos darItem() para interactuar con la nueva clase creada y el método dormir() el cual cambia los atributos de la clase. En cuanto a la nueva clase Items, esta tiene los parámetros id, cantidad y nombre. También tiene 3 clases heredadas, las cuales corresponden a Comida, Medicina y Juguete, las cuales afectan de manera distinta a los atributos de la mascota al ser utilizados.

2.3. Etapa 3

En esta etapa se agrega la clase Inventario, la cual se encarga de ordenar en un atributo utilizando la clase Array todos los ítems de la clase Ítem, para esto se utilizan los métodos addItem() el cual se añade el ítem a la lista, printInv() que cumple la función de mostrar en pantalla el inventario y buscaID() el cual busca un ítem en base a un ID dado. También en esta etapa se modifica la clase Main para agregar el tiempo de simulación el cual sirve para calcular tanto la edad de la mascota virtual, como para actualizar los atributos que cambien en el tiempo de simulación. Unos de los cambios importantes en la ejecución del programa, es que se integra un menú interactivo que permite al usuario ingresar un ID y este se buscará en el inventario, una vez que lo encuentre, se le entregará a la mascota. Dicho cambio permite que el programa no sólo sea imprimir datos, sino que también incluye al usuario en él.

2.4. Etapa 4

El objetivo de esta etapa era lograr la inicialización de inventario a partir del archivo config.csv., además de incluir la funcionalidad en el menú para que la mascota duerma, continúe la simulación sin acción y finalizar la simulación.
Para lograr el objetivo se implementó en la clase Main la lectura del archivo config.csv, con ayuda de la clase Scanner y las funciones readConfiguration() y executeAction() permite que se lea correctamente el formato del archivo y lo ejecute, llenando el inventario y llamando a la función printAttributes(). Dado que esta ultima función se encuentra en un bucle infinito, basta que se cumpla una condición, en este caso ingresar la tecla 'x', para salir del bucle y terminar la simulación imprimiendo que la simulación ha finalizado.

3. Dificultades

En la ejecución de esta tarea como grupo debimos enfrentar diversos desafíos, que van desde entender correctamente el comportamiento de las clases en Java, implementar soluciones al problema presentado y saber exactamente que variables se están modificando dado que se trabajan con clases padres, clases hijas y sus respectivos atributos, sin embargo, podemos destacar 3 desafíos que conllevaron mayor esfuerzo por parte del equipo:

- Implementación del menú interactivo.
- Ingresar los parámetros de la mascota a través de un archivo .csv
- Condición de eliminar un elemento (Item) de una lista (inventario).

4. Soluciones Implementadas

- Para solucionar el problema con el menú interactivo, fue necesario modificar parte del código que habíamos estado construyendo, ya que utilizamos un método de la clase mascota para imprimir los atributos en el Main, y si bbien funcionaba bien, existían problemas a la hora de hacer iterar el inventario que pertenecía a su respectiva clase, ya que no lográbamos representar el ArrayList dentro de la clase Mascota. Para solucionarlo, si bien existen varias formas, como crear un método que itere el menú en la misma clase inventario, se decidió que el método encargado de imprimir los atributos se encontrara en el Main, de esta manera se podían invocar las clases que hemos estado desarrollando sin problema, logrando que el programa no presente fallos a la hora de iterar.
- El problema de ingresar los parámetros a través de un archivo .csv fue mas bien entender el archivo base dejado para la tarea, ya que antes de llenar el inventario con los parámetros del archivo, había que saber que era lo que se estaba extrayendo, el nombre de sus variables y como haciendo uso de ellas se tenia la misma funcionalidad sin la implementación del archivo. Con todo esto se conocieron nuevas funciones y syntax de como trabajar con esta clase de archivos.
- Este problema, si bien puede ser inofensivo, ya que sólo se puede solucionar con el comando For, y encontrando el ID aplicar el comando inventario.remove[ID-1], el problema viene a que el comando que nosotros utilizamos (for item: inventario) al ser de tipo for-each si nosotros

utilizábamos el remove el comando se encontraba con un error fatal. Investigando, se descubre que el error proviene de la manera en que Java maneja este algoritmo, por lo que la solución a implementar corresponde a la de recorrer el inventario y guardar el ID del item a eliminar, para que cuando haya finalizado su recorrido, eliminarlo exitosamente.