Diseño y Programación Orientados a Objetos

1er. Semestre 2024

Tarea 3: Simulación de una mascota virtual en C++ con interfaz gráfica Qt

Lea detenidamente la tarea. **Si algo no lo entiende, consulte. Si es preciso, se incorporarán aclaraciones al final**. Esta interacción se asemeja a la interacción entre desarrolladores y clientes cuando algo no está del todo especificado.

1. Objetivos de la tarea

- Modelar objetos reales como objetos de software.
- Ejercitar la creación y extensión de clases dadas para satisfacer nuevos requerimientos.
- Reconocer clases y relaciones entre ellas en códigos fuentes C++.
- Ejercitar la compilación y ejecución de programas C++ desde una consola de comandos.
- Ejercitar la configuración de un ambiente de trabajo para desarrollar aplicaciones en C++ y Qt.
- Manejar proyectos vía Git.
- Ejercitar la preparación y entrega de resultados de software (creación de makefiles, readme, documentación).
- Familiarización con desarrollos "iterativos" e "incrementales".
- Desarrollar software con interfaz gráfica utilizando la biblioteca Qt.

2. Descripción General

El propósito de esta tarea es simular una **mascota virtual**. Esta corresponde a un símil simplificado de lo ya desarrollado en las tareas 1 y 2 de la asignatura, pero sobre el lenguaje de programación C++ y con el uso de la bibliotecas Qt.

La tarea está dividida en cuatro etapas, siguiendo un enfoque iterativo e incremental. Las dos primeras etapas deben ser desarrolladas en C++ puro, mientras que las dos etapas posteriores se deben desarrollar en C++ en conjunto con el uso de bibliotecas Qt, haciendo uso del IDE Qt Creator. La etapas con interfaz gráfica pueden ser desarrolladas con Qt Designer o vía código.

Pueden ver un ejemplo de funcionamiento de mascota virtual en el siguiente enlace: Juego Pou

2.1 Mascota virtual

El modelo para una mascota virtual en esta tarea contiene las siguientes características e indicadores:

- **Nombre** de la mascota
- **Edad**, la cual se mide en segundos de simulación
- **Salud**, la cual toma valores entre 0 y 100 puntos
- **Energía**, la cual toma valores entre 0 y 100 puntos
- **Felicidad**, la cual toma valores entre 0 y 100 puntos

Algunos indicadores de la mascota pueden verse afectados entre sí:

- Si `edad` <= 5 y `salud`<=10 puntos, la `felicidad` bajará 20 puntos por cada 0.5 segundos de simulación.
- Si 5 < `edad` <= 10 y `salud`<=50, la `felicidad` bajará 20 puntos y la `energía` bajará 10 puntos por cada 0.5 segundos de simulación.
- Si `edad`>10 y `salud`<=50, la `felicidad` bajará 30 puntos y la `energía` bajará 20 puntos por cada 0.5 segundos de simulación.

En base a los indicadores de edad, salud, energía y felicidad, es posible determinar el **estado** en el que se encuentra la mascota. Estos estados están definidos como un dato tipo `enum`. A continuación se presentan la lista de estados posibles según orden de prioridad (de menor a mayor), así como sus condiciones específicas:

1. **Neutro**: Estado por omisión de la mascota.
2. **Feliz**: La felicidad de la mascota es mayor o igual a 60 puntos.
3. **Triste**: La felicidad de la mascota es menor o igual a 20 puntos.
4. **Hambriento**:
    - Si la salud es menor o igual a 20 puntos para una edad menor o igual a 5.
    - Si la salud es menor o igual a 50 puntos para una edad entre 5 y 10.
5. **Enojado**: La edad es mayor a 5, mientras la salud y energía son menores o iguales a 30 puntos.
6. **Cansado**: La energía de la mascota es menor o igual a 15 puntos.
7. **Muerto**: La salud o la energía de la mascota ha alcanzado el valor mínimo de 0, o la edad es mayor o igual a 15.

En caso de verificarse las condiciones para más de un estado a la vez, se privilegia el estado de mayor prioridad.

2.2 Ítems

Los ítems corresponden a diferentes elementos con los que puede interactuar la mascota virtual. Estos se pueden agrupar en las siguientes categorías/clases:

- **Comida**, la cual, al ser utilizada en la mascota, le aumenta 20 puntos de energía y salud.
- **Medicina**, la cual, al ser utilizada en la mascota, le aumenta 40 puntos de salud.
- **Juguete**, el cual, al ser utilizado en la mascota, le aumenta 30 puntos de felicidad.

Cada instancia de un ítem podrá ser usado en la mascota una única vez, **a excepción de los juguetes**. Los juguetes pueden ser utilizados tantas veces como se desee sin que estos desaparezcan.

Adicionalmente, cada una de estas tres clases corresponde a subclases de la clase base `Item`. Esta contiene todos los elementos comunes de los ítems. Esto incluye los siguientes atributos:

- `id`, el cual se utilizará para poder acceder a cada ítem.
- `cantidad`, correspondiente a la cantidad de elementos existente para dicho elemento. Si la cantidad tiene valor `-1`, se considera el objeto ilimitado (válido para los juguetes).
- `nombre`, correspondiente al nombre de dicho ítem.

2.3 Inventario

El inventario contiene todas las instancias de ítems disponibles. Este permitirá la adición o eliminación de ítems según sea necesario, garantizando una actualización constante de los elementos almacenados.

2.4 Configuración inicial

El programa contará con un archivo de configuración `config.csv`, cuyos elementos se separan por `;`. Este archivo contiene tanto el nombre de la mascota, como los ítems que tendrá a disposición en su inventario siguiendo el siguiente formato:

```
Garfield
1;Juguete;Pelota;-1;images/toys/ball.png
2;Comida;Queso;5
3;Comida;Pan;3
4;Medicina;Jarabe;4
```

Las líneas posteriores al nombre de la mascota contienen la siguiente información sobre cada ítem:

```
<identificador_ítem>;<tipo_item>;<nombre_item>;<cantidad_inicial>;
<ubicacion_imagen (opcional)>
```

Si el ítem no tiene una imagen asociada, el campo de la ubicación de la imagen puede omitirse.

El valor **-1** en el campo de cantidad indica que el ítem es ilimitado y puede ser usado tantas veces como se desee. Este valor es válido especialmente para los juguetes.

3. Etapas del Desarrollo

3.1 Etapa 1: Inicialización de la clase Mascota y creación de la clase Item

En esta etapa se desarrollará la clase `Mascota` que representará a la mascota virtual del programa. Esta clase contendrá los atributos y métodos necesarios para modelar el comportamiento y estado de la mascota. También se desarrollará la clase base `Item` y sus subclases `Comida`, `Medicina`, y `Juguete`.

Los valores iniciales para los indicadores de Mascota serán los siguientes:

- `edad = 0`
- `salud = 100`
- `energia = 100`
- `felicidad = 50`
- `estado = "Neutro"`

Considere el código de ayuda disponible en el siguiente enlace para el desarrollo de su código.

Ejemplo de salida esperada

```
$ ./main
```

```
Nombre: Garfield
Edad: 0
Salud: 50
Energía: 50
Felicidad: 50
Estado: Neutro

ID: 1
Nombre: Queso
Cantidad: 5

ID: 2
Nombre: Jarabe
Cantidad: 3

ID: 3
Nombre: Pelota
Cantidad: -1

Usando comida: Queso en la mascota
Nombre: Garfield
Edad: 0
Salud: 70
Energía: 70
Felicidad: 50
Estado: Neutro

Usando medicina: Jarabe en la mascota
Nombre: Garfield
Edad: 0
Salud: 100
Energía: 70
Felicidad: 50
Estado: Neutro

Usando juguete: Pelota en la mascota
Nombre: Garfield
Edad: 0
Salud: 100
Energía: 70
Felicidad: 80
Estado: Feliz
```

3.2 Etapa 2: Creación del inventario y simulación del paso del tiempo

En esta etapa se procederá a la creación de la clase `Inventario`, donde se almacenarán diversas instancias de ítems correspondientes a las clases `Comida`, `Medicina` y `Juguete`. También se implementará el paso del tiempo vía código para la simulación de la mascota. Cada acción de la mascota o interacción con un ítem, produce que el tiempo avance 0.5 segundos (similar a la Tarea 1).

La simulación termina una vez la mascota fallece o se exceden los 15 segundos de tiempo de simulación.

Para verificar la correcta implementación de esta etapa, se solicita lo siguiente:

- Dentro del código de la función `main`, se deberá crear un inventario con ítems iniciales.
- Mostrar el tiempo de simulación a medida que este avanza.
- Mostrar el estado de la mascota, atributos e indicadores en cada incremento de tiempo.
- Mostrar el estado del inventario en cada incremento de tiempo como un menú que permita seleccionar un ítem del mismo.
- Utilice el archivo `config.csv` para inicializar el inventario.

Ejemplo de salida esperada

```
$ ./main

Tiempo de simulación: 0.5
Nombre: Garfield
Edad: 0.5
Salud: 45
Energía: 45
Felicidad: 50
Estado: Neutro

Inventario:
ID: 1, Nombre: Pelota, Cantidad: -1
ID: 2, Nombre: Queso, Cantidad: 5
ID: 3, Nombre: Pan, Cantidad: 3
ID: 4, Nombre: Jarabe, Cantidad: 4

Seleccione un ítem del inventario por su ID: 2

Usando comida: Queso en la mascota
Tiempo de simulación: 1.0
Nombre: Garfield
Edad: 1.0
Salud: 65
Energía: 65
Felicidad: 50
Estado: Neutro

Inventario:
ID: 1, Nombre: Pelota, Cantidad: -1
ID: 2, Nombre: Queso, Cantidad: 4
ID: 3, Nombre: Pan, Cantidad: 3
ID: 4, Nombre: Jarabe, Cantidad: 4

Seleccione un ítem del inventario por su ID: 4

Usando medicina: Jarabe en la mascota
Tiempo de simulación: 1.5
Nombre: Garfield
Edad: 1.5
Salud: 100
```

```
Energía: 65
Felicidad: 50
Estado: 0  # Representa el estado Neutro

Inventario:
ID: 1, Nombre: Pelota, Cantidad: -1
ID: 2, Nombre: Queso, Cantidad: 4
ID: 3, Nombre: Pan, Cantidad: 3
ID: 4, Nombre: Jarabe, Cantidad: 3

Seleccione un ítem del inventario por su ID: 1

Usando juguete: Pelota en la mascota
Tiempo de simulación: 2.0
Nombre: Garfield
Edad: 2.0
Salud: 100
Energía: 65
Felicidad: 80
Estado: 1  # Representa el estado Feliz

Inventario:
ID: 1, Nombre: Pelota, Cantidad: -1
ID: 2, Nombre: Queso, Cantidad: 4
ID: 3, Nombre: Pan, Cantidad: 3
ID: 4, Nombre: Jarabe, Cantidad: 3

Seleccione un ítem del inventario por su ID:
...
```

3.3 Etapa 3: Creación de la interfaz gráfica básica

En esta etapa deberá desarrollar **solo** la interfaz gráfica para la simulación de la mascota y cada diferente menú de opciones (no es necesario implementar las funcionalidades de la interfaz en esta etapa).

La interfaz debe tener los indicadores de la mascota (nombre, edad, salud, energía, felicidad), mostrar el inventario y la imagen de la mascota. Ud. puede definir estos parámetros manualmente para que sean visibles en la etapa presente.

Utilice la siguiente imagen como referencia:

Figura 1. Imagen de interfaz generada en Qt. La imagen de la mascota fueron generadas por el modelo de AI DALL-E, de la empresa OpenAI.

La imagen de la mascota puede ser descargada en el siguiente enlace.

Hints:

- Se pueden agregar imágenes estáticas en una interfaz Qt utilizando la clase `QLabel` en conjunto con `QPixmap`.
- En Qt, el sistema de recursos permite incluir archivos como imágenes, íconos y texto directamente en el ejecutable mediante archivos `.qrc`, mejorando la portabilidad y organización. Puede considerar esta opción para agregar elementos a la interfaz gráfica.

3.4 Etapa 4: Implementación de funcionalidades de la interfaz

En esta etapa completará la implementación de la interfaz gráfica con Qt, añadiendo las funcionalidades respectiva a cada elemento dentro de la misma.

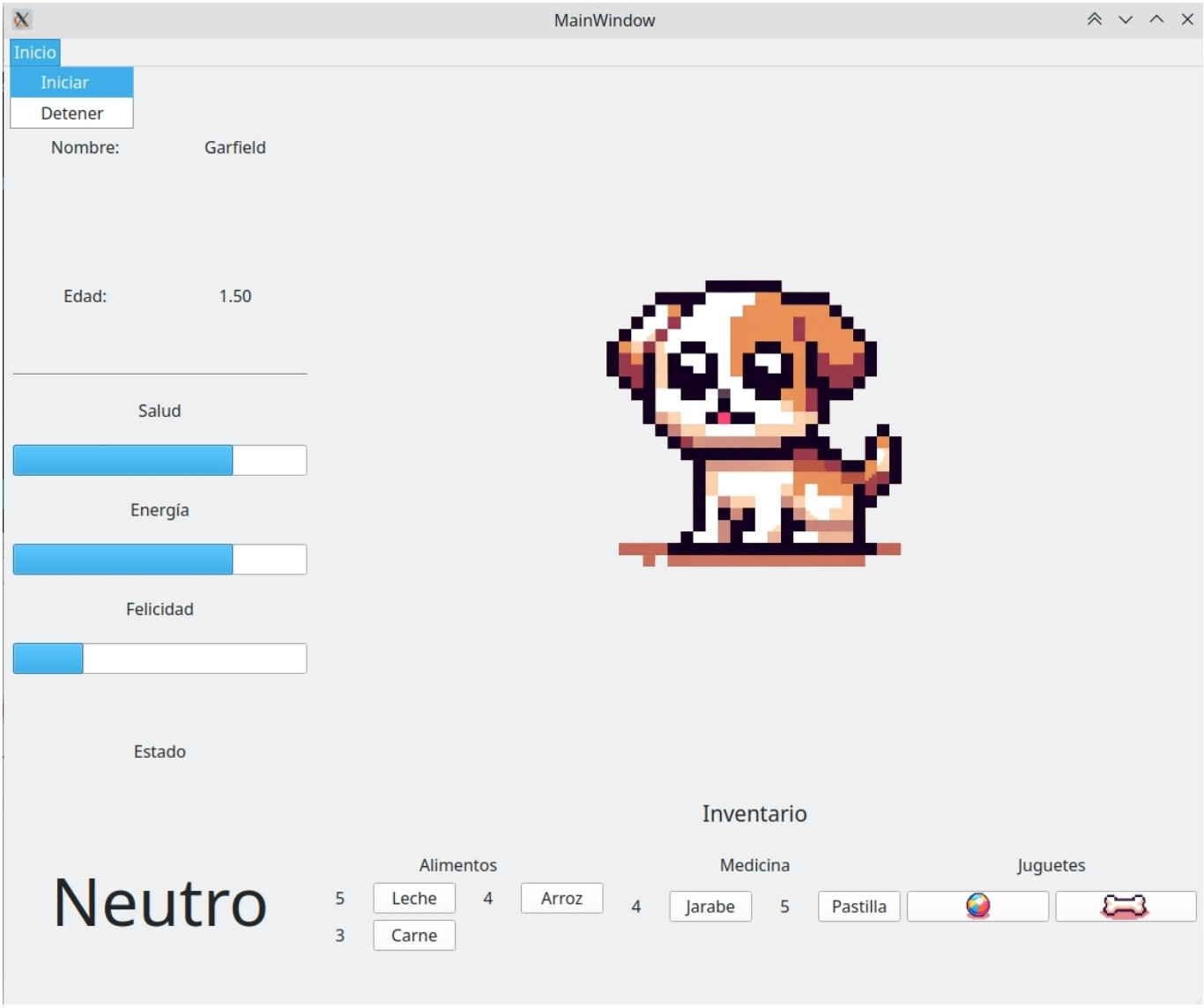
Para ello deberá integrar lo desarrollado en las etapas previas, simulando el comportamiento de la mascota con el paso del tiempo y el efecto de las interacciones en la interfaz.

La simulación comienza inmediatamente después de iniciado el programa y termina una vez la mascota fallece o se exceden los 15 segundos de tiempo de simulación

Hint: Puede simular el paso del tiempo utilizando la clase `QTimer`.

3.5 Extra crédito: Agregación de menú

Agregue a la interfaz un menú que permita tanto iniciar como detener la simulación. Considere la imagen siguiente como referencia.



Esta parte es voluntaria, y su desarrollo otorga 10 puntos adicionales (la nota final se satura en 100). Si desarrolla esta parte, indíquelo en su documentación.

4. Elementos a considerar en su documentación

Entregue todo lo indicado en "Normas de Entrega de Tareas" (entrega por github/gitlab **obligatorio en esta tarea**).

Deberá entregar un proyecto QtCreator que incluya su desarrollo

En su archivo de documentación (`pdf` o `html`) incorpore el diagrama de clases final de la aplicación.

Para la entrega, deberá subir a Aula un enlace a su repositorio Github/Gitlab, dando acceso a los ayudantes para su revisión.