UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

ELO 329

Tarea 3

- Grupo 1 (paralelo 201)
- Integrantes:
  - Francisco Encina
  - Matías Torres
  - José Beltrán
  - Maximiliano Pozo

- Fecha de entrega: 12 junio de 2023

Índice

Resumen

El siguiente informa trata de la tarea 3 de la clase ELO 329 la cual consiste en programar, a través de C++, un sistema de alarma domiciliaria con sus diferentes funciones y objetos. Luego de dar la descripción del programa a elaborar, se prosigue dando a conocer las soluciones realizadas para poder resolver las diversas problemáticas y dificultades que aparecen al momento de desarrollar el código. Dado lo anterior, el informe continúa con un diagrama UML, el cual se basa en el desarrollo de la tarea. Finalmente, se coloca la información necesaria para comprender de mejor manera la solución desarrollada en la tarea.

Descripción del programa

Este programa consiste en modelar y programar un sistema de alarma domiciliaria a través de C++. Para esto se necesita desarrollar los diferentes elementos que presentan estos tipos de sistemas de seguridad a través de cuatro etapas, las cuales permiten el avance progresivo de la tarea. En este caso la primera etapa consiste en una casa la cual consta con puertas y sus respectivos sensores, los cuales deben identificar cuando cambia el estado de la misma puerta (abierta o cerrada). Siguiendo con la segunda etapa, a lo anterior hay que agregarle ventanas y la central correspondiente para poder verificar e indicar en cada zona si las todas las ventanas y puertas se encuentran cerradas o no. Para todo lo anterior, no se necesita una interfaz gráfica. Por otra parte, en la tercera etapa se debe desarrollar todo lo anterior de las etapas 1 y 2 en una interfaz gráfica, la cual debe mostrar gráficamente las puertas y ventanas. Estas deben reaccionar al botón izquierdo del mouse y en este caso, la central cada 200[ms] debe revisar el estado de los sensores, enviando un mensaje en el caso de que haya ocurrido algún cambio, es decir, si es que hay alguna zona abierta. Finalmente, la cuarta etapa consiste en añadir a la interfaz gráfica ciertos implementos como lo son dos botones que arman y desarman la misma alarma; un “display” que indica aspectos importantes al armar y desarmar la alarma (indica si una zona está abierta al momento de armar); y finalmente una sirena que indique a través de un parpadeo cuando se haya vulnerado la seguridad (cuando esté desarmada será de color verde y cuando sea vulnerada de otro color).

Dificultades

Una de las dificultades que se presentó durante las tres primeras etapas fueron las fugas de memoria ya que, al momento de desarrollar el código, fue bastante complicado encontrar la forma de liberar los bloques de memoria reservados, sobre todo en la tercera etapa. Esto causó un gran desgaste del tiempo disponible para la entrega ya que fue un problema que se mantuvo constante durante el desarrollo de esta tarea.

Otra dificultad que se presentó durante el avance de esta entrega, fueron los códigos de ayuda entregados por parte de los profesores del ramo, todo esto debido a que entre etapas se cambiaban variables y algunas de estas las colocaban como punteros. Todo esto hacía que la lógica del mismo programa fuera más compleja y/o engorrosa de entender, lo cual también afectó la eficiencia del grupo al momento de complementar el código de cada etapa.

Otro inconveniente que se manifestó durante la etapa 4 fue la forma de implementar el display, los botones y la sirena correspondiente. Todo esto debido a que fue complejo encontrar una forma de ordenar la interfaz a través de todas las opciones que nos presentaba QT y también, fue bastante difícil encontrar la manera de coordinar las funciones de los botones con la respectiva acción solicitada en la tarea.

La última dificultad se manifestó durante la etapa 4, en la cual no se logró implementar de buena manera una de las interacciones de los botones, ya que uno de estos no realizaba correctamente lo que se solicitaba en los enunciados de la entrega. Esto fue algo complejo, ya que no se pudo encontrar una solución concreta a esta problemática.

Solución

Como solución a lo solicitado, se desarrollaron en cada etapa las clases correspondientes de forma que, al momento de ejecutar el código, se consiguió cumplir mayoritariamente con lo especificado en las instrucciones de la tarea. Se logró desarrollar de manera satisfactoria la mayor parte de las clases solicitadas en los enunciados de cada etapa, incluyendo la interfaz gráfica con sus respectivas ventanas, puertas y sensores. Cada una de las clases se programó para que funcionaran efectivamente de forma colectiva, dado que la programación orientada a objetos así lo amerita.

Inicialmente se pudo desarrollar exitosamente las etapas 1 y 2 con sus respectivas clases. Se logró progresar de buena manera con cada clase, a pesar de algunas dificultades que se presentaron durante el desarrollo del código. Lo anterior se refiere a las fugas de memoria que se manifestaron sobre el transcurso de la segunda etapa. Durante el desarrollo de la central y del main, no se lograba liberar la memoria de cada objeto creado durante la ejecución del código, pero luego de un par de consultas y de investigar arduamente este problema, se logró solucionar satisfactoriamente el memory leak.

Otro de los problemas que se manifestó durante el desarrollo de la siguiente etapa fueron los cambios de los códigos de ayuda. Con respecto a este punto, como se dijo anteriormente estos cambios hacían que fuera algo más complicado el poder hilar cada una de las etapas. Afortunadamente a través de diversas consultas y de trabajo en equipo se logró progresar de buena manera con los códigos de las 3 etapas, generando un código congruente entre estas mismas. Debido a lo anterior, se pudo avanzar de buena forma a la etapa 4 ya que se tenía un código ordenado y fácil de entender.

La siguiente temática en la que se logró encontrar una solución conveniente fue con respecto a la interfaz gráfica desarrollada en QT. Se pudo desarrollar de buena manera la interfaz con lo solicitado en la misma entrega, es decir que se logró colocar correctamente los botones y el display correspondiente a través de las diversas opciones que nos entregaba esta IDE y también por medio de la ayuda de los docentes en los horarios de consulta correspondientes.

Dificultades no solucionadas

Lo que no logramos solucionar fue que, al armar la central, al abrir una puerta o ventana se sigue imprimiendo el estado de “Zonas cerradas”, sin embargo, esto no debería ser así, ya que al abrir un puerta o ventana el estado debería ser “Alguna Zona abierta”.
Las posibles razones por las que no pudimos solucionar dicha dificultad fueron:

1. Lógica incorrecta: Puede que la lógica implementada en el código no sea la correcta. Puede haber un error en las condiciones o en la forma en que se verifica el estado el estado de las zonas.
2. Falta de actualización de estado: puede ser que el estado de las zonas, no se esté actualizando correctamente y por lo tanto la central sigue mostrando "Zonas cerradas". Esto puede requerir la implementación de métodos o eventos que actualicen el estado de la central cuando se produzcan cambios en las zonas.
3. Problemas de interacción entre clases: puede haber problemas en la comunicación entre las diferentes clases involucradas. Debemos asegurarnos qué estén bien conectadas.

Diagrama UML

En este diagrama, la clase "*Central*" es la clase principal y tiene una relación de composición con las clases "*Siren*", "*Window*" y "*Door*", lo que significa que una instancia de la clase "*Central*" puede contener cero o muchas instancias de estas clases.

A su vez, las clases "*Window*" y "*Door*" tienen una relación de composición con la clase "*Sensor*" y la clase "*MagneticSensor*". Esto indica que cada instancia de "*Window*" y "*Door*" contiene una instancia de "*Sensor*" y "*MagneticSensor*", respectivamente.

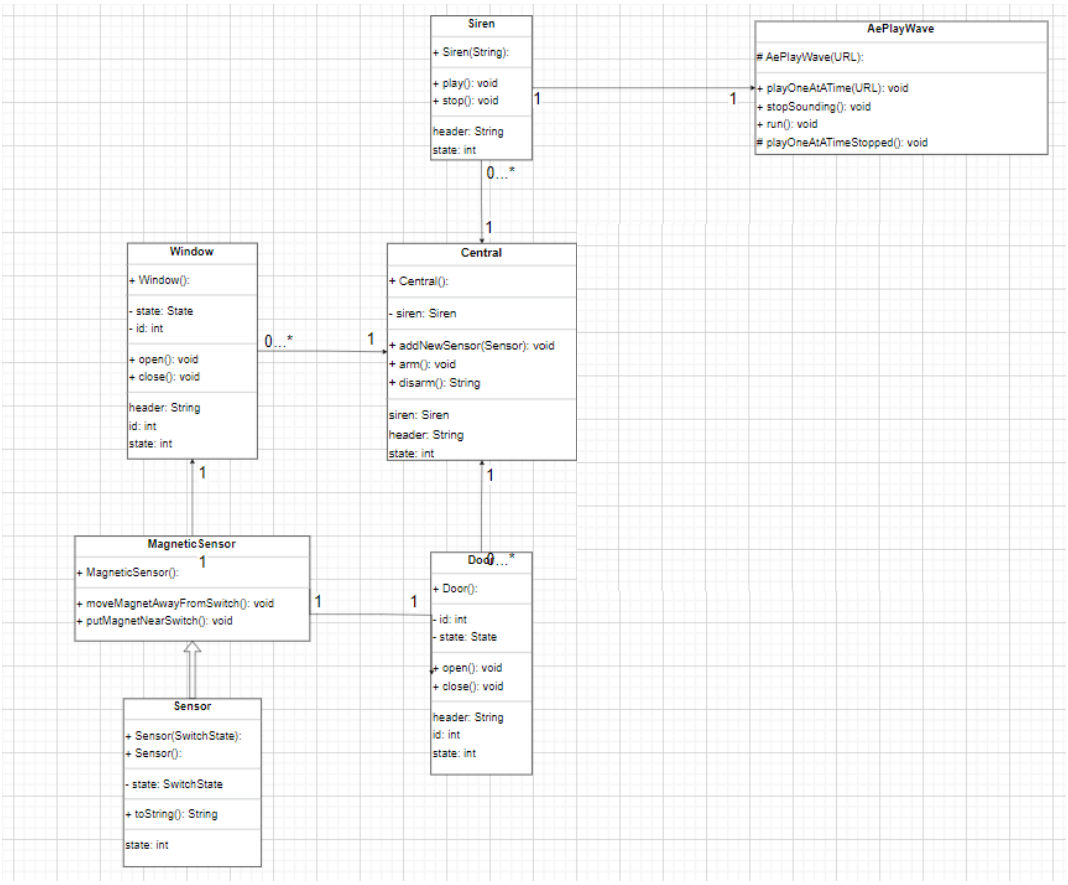

Fig 1. Diagrama de clases de la tarea 3