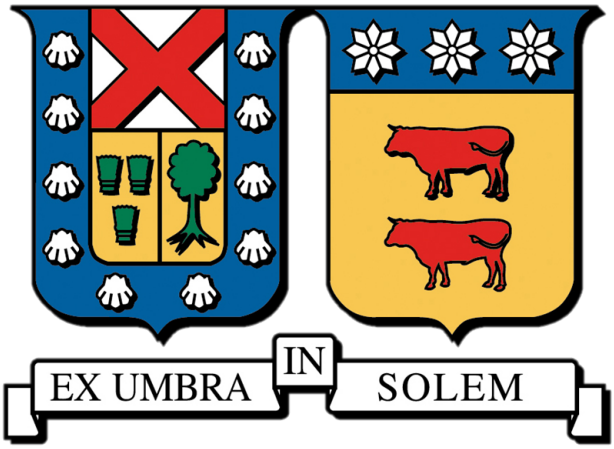
Universidad Técnica Federico Santa María



Tarea 3

Fecha: 11/06/2023
Integrantes: Sofía Martínez
Asignatura: Programación Orientada a Objetos

Diagrama UML

El diagrama de la figura 1 describe la situación de una alarma domicilia donde existen sensores magnético únicamente, los cuales se encuentran asociados a las puertas y ventanas. Además todos estos sensores son controlados por una central la cual verifica si algún sensor ha sido activado y activa la alarma cambiando de color entre rojo y verde. De acuerdo a lo que se observa en el diagrama las flechas de color negro continuas corresponde a la relación de herencia donde la clase padre es Sensor y su hijo corresponden a MagneticSensor, las flechas que continuas son aquellas flechas representan asociaciones de tiene, como lo sería en este caso como afectan las funciones de Door a la clase MagneticSensor, mientras que por el contrario las semi continuas representa que la clase desde donde parte la flecha, usa código para crear nuevo código a partir de la clase a la que apunta, es decir invoca código de otro clase.

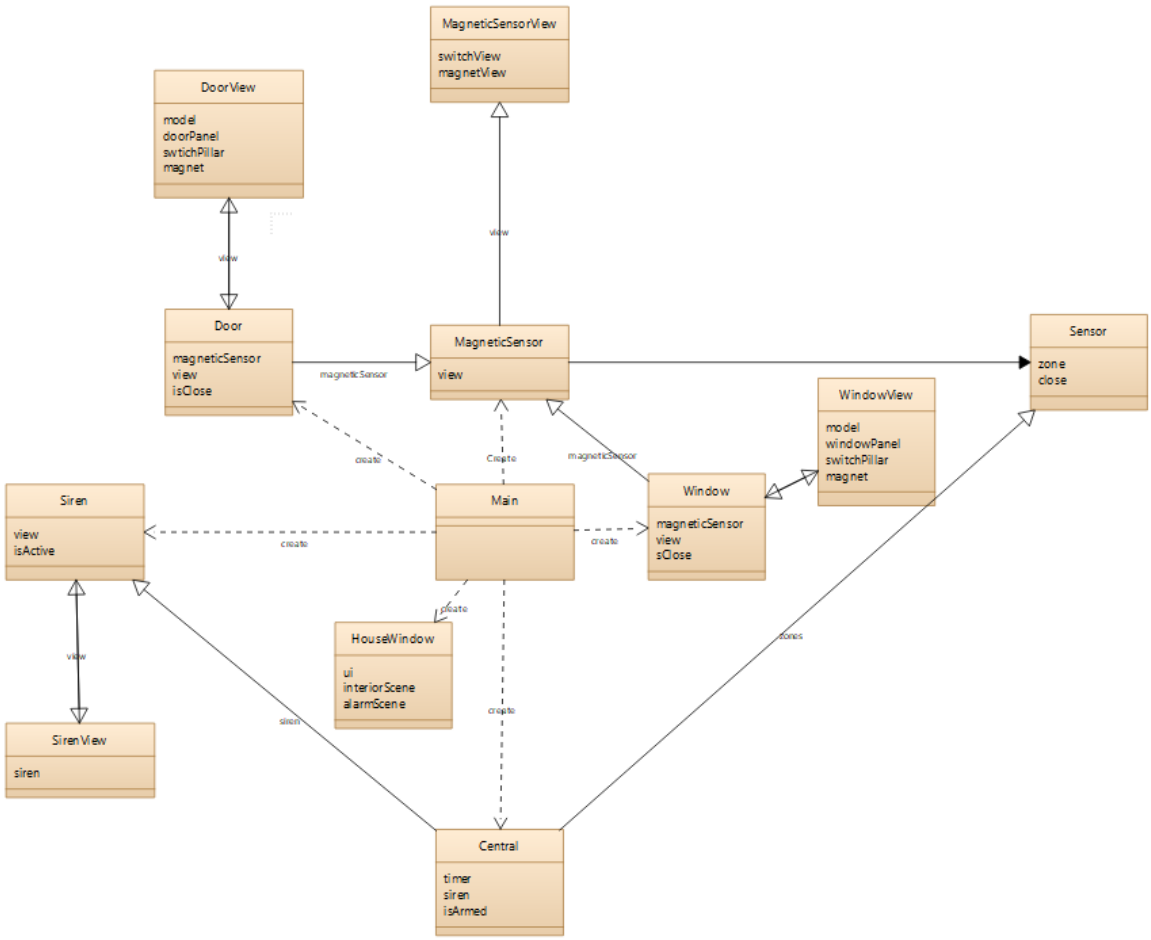


Figura 1: Diagrama UML Alarma Domiciliaria

Dificultades

La dificultades presentada fueron:

1. Una dificultad presentada fue el poder ejecutar el código de ayuda de la etapa 3 en el editor de código "Qt Creator"debido a que no se estaba generando el qmake de manera correcta. La solución fue consultar

al profesor en donde indico como poder importar correctamente el archivo utilizando el archivo con extensión .pro entregado en la tarea.

2. Otra dificulta fue la aparición del error “reference to ’byte’ is ambiguous“ en el stage 3 al incluir el paquete QWidgets. La solución para esto se obtuvo realizando búsquedas por internet donde se encontró que el problema era debido al uso de üsing namespace std”, por lo que se tuvo que eliminar esto y realizar el uso con std:: para poder mostrar información por pantalla o para vector¡¿.

3. La última dificultad presentada fue el poder realizar la parte gráfica de la central donde inicialmente se estaba tratando de utilizar QHBoxLayout y QVBoxLayout, pero este se podía utilizar únicamente creando una nueva ventana, por lo que se tuvo que cambiar la manera de creación realizando directamente en el creador de interfaz gráfica de la clase HouseWindow.

Desarrollo

La solución propuesta para el problema planteado consta de 12 clases las cuales son Central, Door, DoorView, HouseWindow, MagneticSensor, MagneticSensorView, Sensor, Siren, SirenView, Window, WindowView y main. En primer lugar, main es la clase principal desde donde se encuentra el método main y por consiguiente desde donde se ejecuta el programa, dado que en la ejecución del programa se entrega el archivo de configuración de las puertas y ventanas en el se crean estos, utilizando las clases Door, DoorView, Window y WindowView; donde en las clases WindowView y DoorView se entrega la ubicación en el plano x, y, junto con el ángulo; dado las puertas y ventanas son entidades independientes de los sensores magnéticos cuando se crea una puerta a este se le asocia un sensor magnético utilizando la clase MagneticSensor, que en principio se encuentra abierto al no estar asociado a una puerta o ventana, pero luego cuando se asocian a estos (puertas y ventanas) son cerrados por defecto para la ejecución de esta tarea, además las puertas poseen un estado que se representa mediante un booleano definida como isClose que puede ser true (en caso de estar cerrada) o false (en caso de estar abierta). Por otro lado, debido a que MagneticSensor tiene algunas características en común de Sensor, que puede estar abierto o cerrado, se utilizó una clase desde la cual dicha clase hereda, llamado Sensor. El estado de abierto y cerrado se asigna con un valor booleano.
Adicionalmente, en la clase main se crea la clase Siren y SirenView en donde su estado, es decir si esta encendida o no por defecto no esta encendida y también tiene los métodos para iniciar o detenerlo.
Por otro lado, esta la Central la cual maneja el armado o desarmado del perimetral de la alarma, además de almacenar la información de los sensores y las zonas a las cuales se encuentran asociados de acuerdo al archivo de configuración. Además de armar y desarmar, la central verifica si algún sensor se encuentra activado, por lo que se utiliza el método checkZones, una vez que la central identifica que algún sensor se fue activo este no dejara de marcar en rojo la sirena hasta que sea desarmada. Finalmente, la clase HouseWindow es la encargada de la parte gráfica del programa en donde se realiza el dibujo de las puertas, ventanas y la central en donde se encuentran los controles como los botones para armar y desarmar la central, además de la sirena que cambia de color entre verde y rojo.