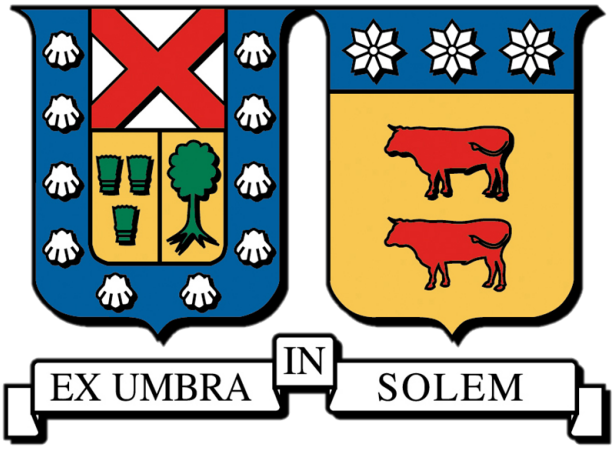
Universidad Técnica Federico Santa María



Tarea 3

Fecha: 14/05/2023
Integrantes: Manuel Torres
Florencia Norambuena
Darael Badilla
Asignatura: Programación Orientada a Objetos

Diagrama UML

El diagrama de la figura 1 describe la situación de una alarma domicilia donde existen solamente sensores magnéticos, estos se encuentran asociados a las puertas y ventanas. Además, estos sensores son controlados por una central la cual verifica si algún sensor ha sido activado cada 200[ms], activando la sirena visual correspondientemente. De acuerdo a lo que se observa en el diagrama, la flecha continua con punta blanca corresponden a relaciones de herencia, donde la clase padre es Sensor y su hijo corresponde a MagneticSensor, las flechas de trazo continuo de punta negra representan asociaciones que una clase posee con otra, como lo sería en este caso la relación entre la clase Door con las clases State, MagneticSensor y DoorView (Door tiene por miembro a Door). Finalmente las flechas semi continuas representan que se utiliza código de la clase del origen de la flecha para crear código en la clase a donde apunta la flecha, es decir, invoca código de otra clase.

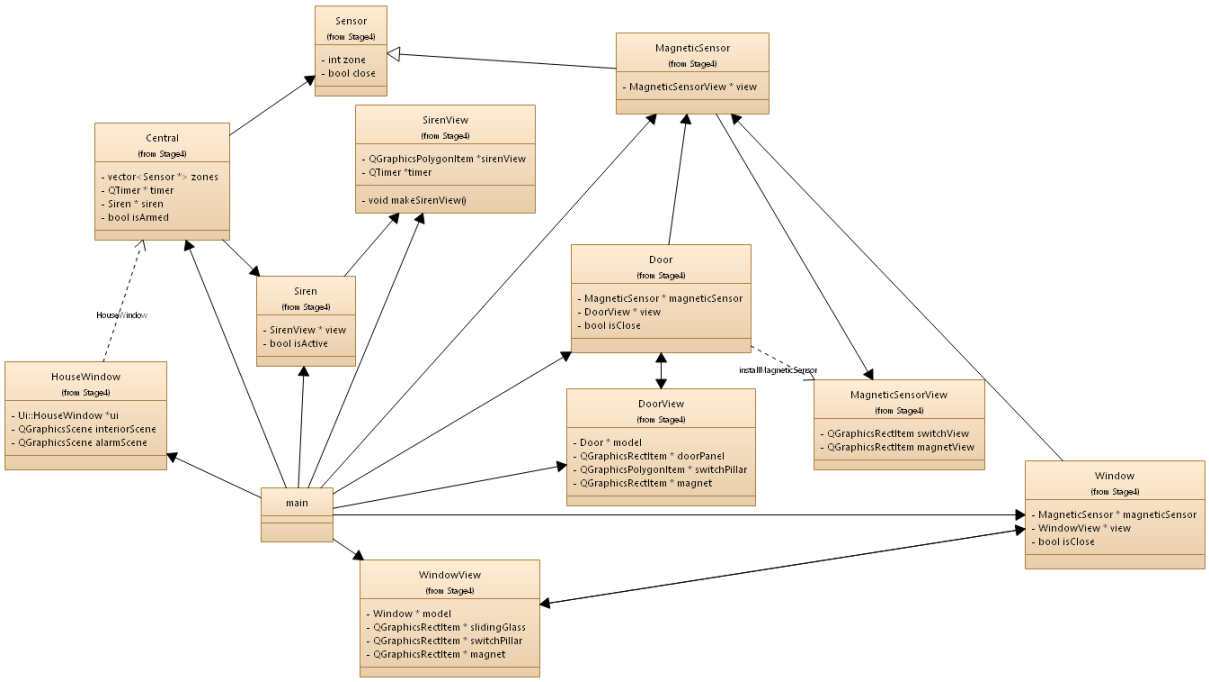


Figura 1: Diagrama UML Alarma Domiciliaria

Dificultades

La dificultades presentadas fueron:

1. El primer problema fue poder conectar los 2 botones con los métodos de la clase "Central" en otra clase (clase "HouseWindow"). Para solucionar esto, se agrega un nuevo argumento en el constructor "HouseWindow()" el cual es de tipo "Central". Se considera como dificultad ya que se tuvo que estudiar con tiempo el dónde poder ubicar el método connect() y además familiarizarse con el método para poder enlazar la interfaz con métodos de clases distintas.

2. El segundo problema fue poder entender cómo operar con la parte gráfica (agregar elementos gráficos y editar sus características como es el tamaño). Para poder abordar esta situación se debió entender que los elementos gráficos trabajan en contenedores. Por lo tanto ,para dividir la pantalla en 2 es necesario generar 2 Contenedores y así ingresar los elementos deseados dentro de los contenedores. Para poder ajustar los tamaños se debe de usar los espaciadores y ajustar su tamaño de acuerdo a las opciones dadas por estos mismos (Abarcado total, abarcado de pantalla o personalizado) de esta forma se pudo

y ajustar el tamaño por ejemplo de la ventana que muestra la sirena y dar espacio entre los botones.

3. La última dificultad detectada fue poder describir métodos setOpen() y setClose() ya que había desconocimiento sobre como determinar los puntos a los que la ventana se debía mover. Luego de poder estudiar los atributos de los objetos QGraphicsRectItem, se entontró la forma de poder retornar la posición del objeto (rect → pos()) y a su vez definir el final restando la posición del eje x con la función rect()→width que retorna el ancho del panel deslizante.

Desarrollo

La solución propuesta para el problema planteado consta de 17 clases las cuales son main, Central, Door, DoorView, HouseWindow, MagneticSensor, MagneticSensorView, Sensor, Siren, SirenView, Window, WindowView. 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 partes que componente la casa (puertas y ventanas) en él se crean los objetos de la sirena, la vista de Sirena, central, y la interfaz gráfica de la casa, utilizando las clases Siren, SirenView, Central y HouseWindow; dado que para esta tarea se realiza la utilización de parte gráfica, a Siren se le asocian las clases SirenView y los botones de la UI (user interface) a la clase Central. A la clase HouseWiendow se le entrega la configuración de las puerta, ventanas y PIR, en donde se crean los objetos de esto utilizando las clases Door y Window, adicionalmente se agrega la componente gráfica a partir de las clases DoorView y WindowView respectivamente; 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 cerrado al no estar asociado a una puerta o ventana y luego se asocian a estos (puertas y ventanas), además las puertas poseen un estado almacenado en la variable isClose que se representa mediante un booleano (TRUE para cerrado y FALSE para abierto). Por otro lado, debido a que MagneticSensor tiene algunas características en común de Sensor ya que de dicha clase es heredado. El estado de abierto y cerrado se asigna utilizado el método setClose(bool).
La animación de la Sirena se utiliza en la clase SirenView. Define si el estado (si la zona ha sido violada o no) por defecto está sin animación. La sirena hace uso de un timer de 100ms se ejecuta para cambiar el color de la sirena.
Finalmente, esta la Central la cual maneja el armado tanto perimetral como completo de la alarma, esta conecta un timer de 200ms con la ejecución de la función checkZones() para detectar si es que alguna zona ha sido abierta, además de generar las 2 zonas que existen las cuales corresponde a la zona 0, zona 1 que se define en el archivo de configuración. Además de armar y desarmar, la central verifica si algún sensor se encuentra activado, por lo que se utiliza el método checkZone ejecutada de manera periódica cada 200ms como se ha mencionado anteriormente, una vez que la central identifica que algún sensor se fue activo este no dejara de sonar hasta que sea desarmada. Por otro lado, el sistema también detecta si es que una zona está abierta antes de armar las zonas.