UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

DEPARTAMENTO
DE INFORMÁTICA

Tarea 1

ELO329 - Diseño y Programación Orientada a Objetos

Bruno Vega

201854051-k

Constanza Alvarado

201973521-7

Sebastián Martínez

201873519-1

Luis González

201892004-5

1. Solución del problema

La solución propuesta consiste en un desarrollo iterativo-incremental para representar un sistema de alarma domiciliaria. En particular se busca resolver la problemática mediante el uso de clases que representan los diferentes elementos con los que cuenta el sistema de seguridad, tales como sensores magnéticos para las puertas y ventanas, sensores infrarrojos pasivos para detección de movimiento, una central que toma como entradas la información provista por estos sensores, una sirena conectada a la central que se gatilla según se cumplen condiciones específicas, y por último, se tienen clases Recta y Círculo que se usan para calcular el área de detección de los detectores PIR.

Para identificar las relaciones entre las clases desarrolladas y cómo interactúan, se hace referencia a la figura 1. Existen clases para sensores, puntos de entrada y salida del domicilio, para la central del sistema de seguridad, mecanismos de alerta y de personas que simulan recorrer el domicilio.

Clase Stage

Esta clase representa la actividad Main, en esta rutina se carga el archivo de configuración indicado por el usuario, la cual incluye cantidad de sensores, el posicionamiento de estos, y el sonido de alarma para la sirena. Al tomar la configuración, crea tantos objetos de clase Door, Window y PIR como sean indicados, además instancia un objeto Central y uno Siren.

También toma los inputs para navegar y accionar el menú de opciones.

Clase Person

Esta clase indica la información de posición de la persona que circulará en el domicilio, cuenta con atributos para las coordenadas y métodos para desplazarse y obtener su estado actual.

Clase Siren

La clase Siren es instanciada cuando se crea la central, representa el dispositivo que alerta cuando es abierto un sensor o se detecta movimiento mientras el sistema está “armado”. Esta alerta se genera mediante un llamado de la clase Central.

Clase Central

La clase Central interactúa con la clase Sirena, con el fin de reproducir un sonido de alarma cada vez que sea necesario según las condiciones de armado indicadas. Interactúa con la clase Sensor para determinar cuáles son las zonas a proteger y por cuantos y cuales sensores está conformada. Además, interactúa con la clase Persona para obtener cuántas personas están circulando en el domicilio y accionar la alarma si es que estas cruzan por un sensor PIR.

Clase Sensor

La clase Sensor es la clase padre de Magnetic Sensor y PIR Detector. Esta clase tiene métodos para obtener el estado actual del sensor y un atributo para identificar si el sensor está en estado OPEN o CLOSE. Además, la clase Sensor cuenta con clases que heredan de esta, las cuales son Magnetic Sensor y PIR Detector.

PIR Detector

La clase PIR Detector hereda los atributos y métodos de Sensor. Esta interactúa directamente con la clase Central, Círculo y Recta. Central llama al método constructor de PIR para crear tantos sensores como sean indicados en la configuración. PIR hace uso de las clases Círculo y Recta para calcular el área de detección que se genera según el rango y ángulo de detección.

Magnetic Sensor

Esta clase hereda de la clase Sensor y además se le agregan métodos que interactúan con Door y Window. Estos métodos simulan la acción de juntar o separar los imanes que indican si una puerta o ventana está cerrada o abierta. Los elementos que interactúan con Magnetic Sensor y hacen uso de los métodos son las clases Door y Window.

Door y Window

Estas clases representan a las puertas y ventanas que pueden ser abiertas o cerradas dentro del domicilio. Interactúa con la clase Magnetic Sensor específicamente, ya que indica los cambios de estado cuando son abiertas o cerradas.

Clases complementarias

AePlayWave

Esta clase viene implementada para dar funcionamiento a la reproducción de sonidos desde el PC, con el fin de simular el accionamiento de la Sirena cuando se activa la alarma de un sistema armado.

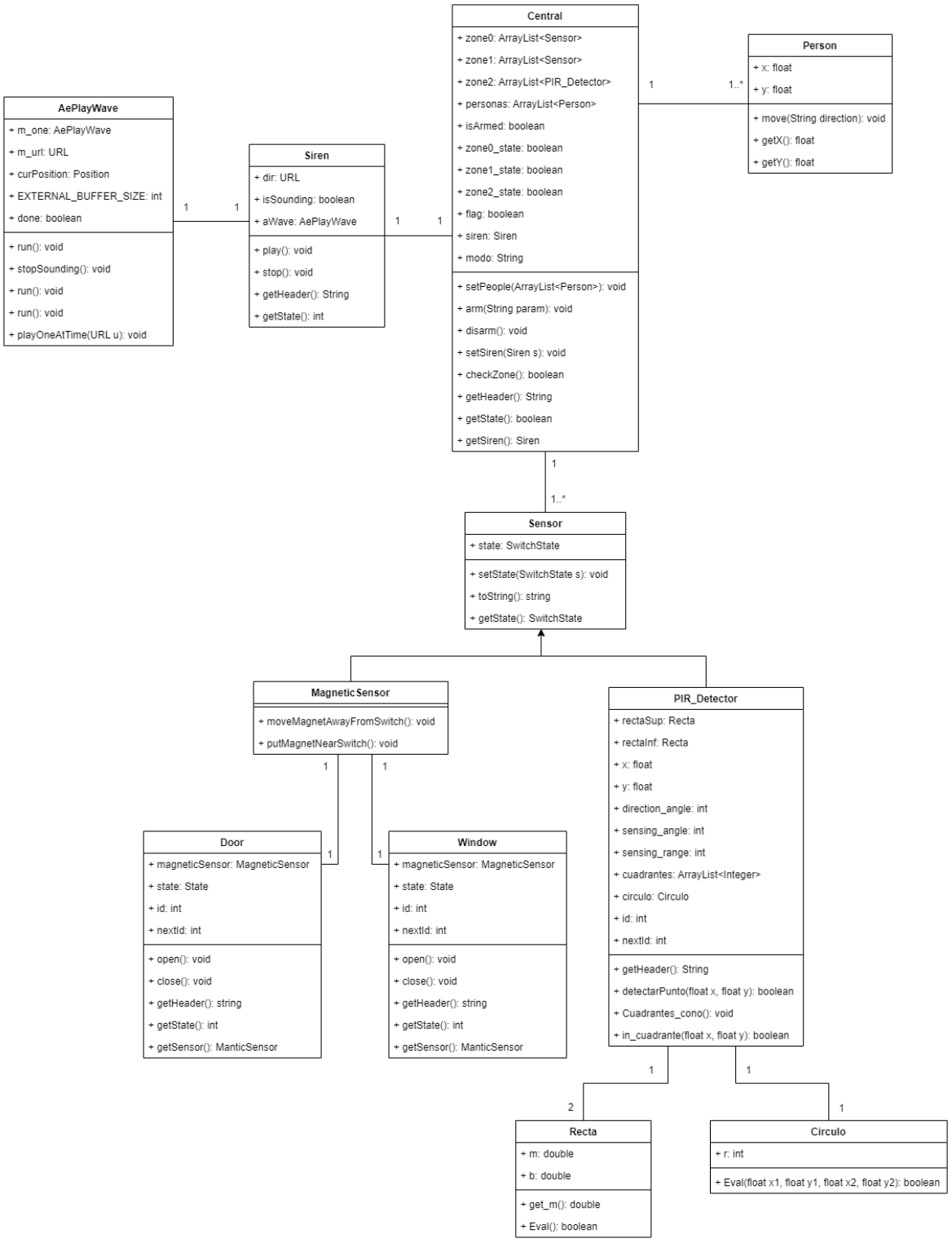
Recta

Se usa la clase recta para poder calcular el área de detección que tiene un Sensor PIR, esta clase solo interactúa con PIR ya que es el único que requiere de cálculos que involucran rectas.

Círculo

Se usa la clase Círculo en conjunto con la clase Recta para poder determinar el área de detección que tiene un Sensor PIR. Esta clase solo interactúa con la clase PIR Detector.

2. Diagrama UML de Clases Implementada



3. Dificultades Enfrentadas Durante el Desarrollo

El proceso de desarrollo implicó analizar de forma conjunta la mejor forma para poder representar el sistema de seguridad de manera que fuese certero en cuanto al comportamiento y funcionalidades. Sin embargo, se produjeron situaciones donde el avance se vio estancado. Los principales problemas fueron:

1. Al momento de empezar a desarrollar, fue problemático no haber establecido firmemente un plano de referencia para poder saber con certeza si el mapa del domicilio tenía origen en el centro, o en el extremo inferior izquierdo, por lo que ciertos métodos calculaban de manera errónea en la primera iteración implementada.

   Por ejemplo, los sensores PIR podían detectar movimiento en dos zonas opuestas, para corregirlo se procedió a establecer que el origen del espacio sería en el centro de este, por si era necesario que se pudiera manejar entradas de coordenadas negativas.

2. La implementación por clases se llevó a cabo por separado entre los miembros, por lo que al momento de trabajar con una clase o método que desarrolló otro compañero, hubo fallos en algunos casos en entender que necesitaba el método para operar, que realizaba y que retornaba.

   Para arreglar cada parte del código donde había esta clase de malentendidos, se comunicaba con los compañeros para pedir una explicación de cómo funcionaban los métodos en cuestión y se procedía a arreglar el error inicial.

3. La realización de los makefiles fue más complicada de lo esperado ya que había que investigar precisamente como se trabaja para compilar las clases y las clases que tienen dependencias.

   Para este problema se buscaron videos de ayuda y documentación oficial para generar Makefiles, con lo que se pudo entregar un orden específico para que se completara exitosamente el proceso de compilado.