Diseño y Programación Orientados a Objetos
1er. Semestre 2023
Tarea 2: Simulador Gráfico de una Alarma Domiciliaria

Lea detenidamente la tarea. **Si algo no lo entiende, consulte. Si es preciso, se entregarán aclaraciones vía correo**. 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 Java.
- Ejercitar la compilación y ejecución de programas JavaFX desde una consola de comandos.
- Ejercitar la configuración de un ambiente de trabajo para desarrollar aplicaciones JavaFX. Se pide trabajar con un IDE. Si bien IntelliJ es el sugerido, su grupo puede trabajar con otro IDE.
- Manejar proyectos vía GIT.
- Ejercitar la preparación y entrega de resultados de software (creación de makefiles, readme, documentación).
- Familiarización con desarrollos "iterativos" e "incrementales".

2. Descripción General

Esta tarea busca adaptar y extender la tarea 1 para simular gráficamente, usando JavaFX, el modelo de una alarma domiciliaria. Consideraremos una **casa** que tendrá una lista de personas, un conjunto de puertas, ventanas, detectores de movimiento, una central y una sirena. En esta tarea usted modelará y programará los elementos sensores (magnético e infrarrojos) y de alerta (sirena) presentes en una alarma simple. Por completitud de la tarea, se pasa a describir el modelo a usar para un sensor magnético, un detector de movimiento PIR (Passive InfraRed), una sirena y una central.

Figura 1: Modelo simplificado de puerta, ventana y sensores PIR

2.1. Sensor magnético

Un sensor magnético es usado normalmente para detectar la apertura de puertas y ventanas. En este uso, su aspecto se ve aquí. Cuando el imán (en verde en Figura 1) se aleja del interruptor (en rojo en Figura 2), éste abre el circuito entre sus terminales. Cuando está suficientemente cercano, el interruptor cierra el circuito. En su instalación en puertas y ventanas, el imán está adosado a la parte movible y el interruptor a la parte fija. Así, al abrir la puerta o ventana, éstas alejan el imán del sensor, el cual cambia el estado del interruptor de cerrado a abierto. En esta tarea, este sensor será usado en puertas y ventanas, de manera que, ante la apertura de una puerta o ventana por un usuario, ésta cambiará el estado del sensor. Lo haremos así pues es más complejo modelar la apertura del interruptor como consecuencia de la ausencia del campo magnético debido al alejamiento del imán cuando un usuario abre la puerta o ventana.

2.2. Detector de movimiento

Un sensor infrarrojo pasivo (PIR, por Passive Infrared) es usado normalmente para detectar movimiento. Este sensor cuenta con un sistema detector de infrarrojo y un interruptor el cual se abre cuando detecta un cambio no gradual en la señal infrarroja recibida[1]. Como resultado, permite

[1] Todo cuerpo emite radiación la cual es mayor si su temperatura es mayor. Esto explica por qué al calentar un objeto se pone de color rojo.

detectar el movimiento de personas dentro de un cierto ángulo y distancia (Ver área de color naranja en Figura 1). Normalmente son instalados en las esquinas de las habitaciones. En esta tarea, cada detector PIR rastreará cada 300 [ms] si hay alguna persona en su área de detección. Para esto el sensor PIR tendrá acceso a la instancia de Casa que lo contiene.

2.3. Sirena

Una sirena es utilizada para alertar vía un sonido a las personas y vecinos cuando el sistema de **seguridad** ha sido transgredido. Ésta está ubicada normalmente en la parte exterior de la casa. La sirena suena a petición de la central una vez que ésta identifica que el circuito de alguna de sus zonas ha sido abierto cuando la alarma está activada (o "armada") para esas zonas. En esta tarea junto con reproducir el sonido de la sirena, ésta conmutará entre verde y otro color de su elección. Al desarmar la alarma, ésta vuelve a su color verde (ver Figura 2, sector derecho).

2.4. Central y teclado

La central es la unidad de control de la alarma. Normalmente dispone de un teclado y un display alfanumérico. En esta tarea, la interfaz presenta tres botones para armar y desarmar la alarma. El armado permite activar sólo los sensores perimetrales (puertas y ventanas) o la totalidad de los sensores (puertas, ventanas y detectores de movimiento). Notar que, para armar una alarma, todas sus zonas a armar deben estar cerradas. Al armar la alarma estando abierta la zona deseada, la central mostrará en el Display los números de las zonas no cerradas.

Una central típicamente puede manejar entre 4 y 8 zonas. Así un conjunto de sensores se conecta de manera serial a una zona y la central los distingue separadamente de otra zona. Esto permite que, por la noche, se active sólo las puertas y ventanas de manera de no disparar la alarma cuando un detector PIR detecte la circulación de ocupantes de la casa. En esta tarea consideraremos tres zonas (etapa 4 de la tarea) puerta principal, otras puertas y ventanas perimetrales, y detectores PIR.

Estando armada, la central revisará el estado de las zonas armadas cada 200 [ms] y según esto disparará la sirena.

3. Interfaz Gráfica

En la primera tarea usted ya aprendió a leer datos desde un archivo. En esta tarea su programa debe generar una interfaz definida por su código similar a lo mostrado en Figura 2. La apariencia final de las ventanas y puertas las puede elegir usted. Notar que ahora el archivo de entrada incluye además algunos parámetros asociados a la ubicación y orientación de puertas y ventanas.

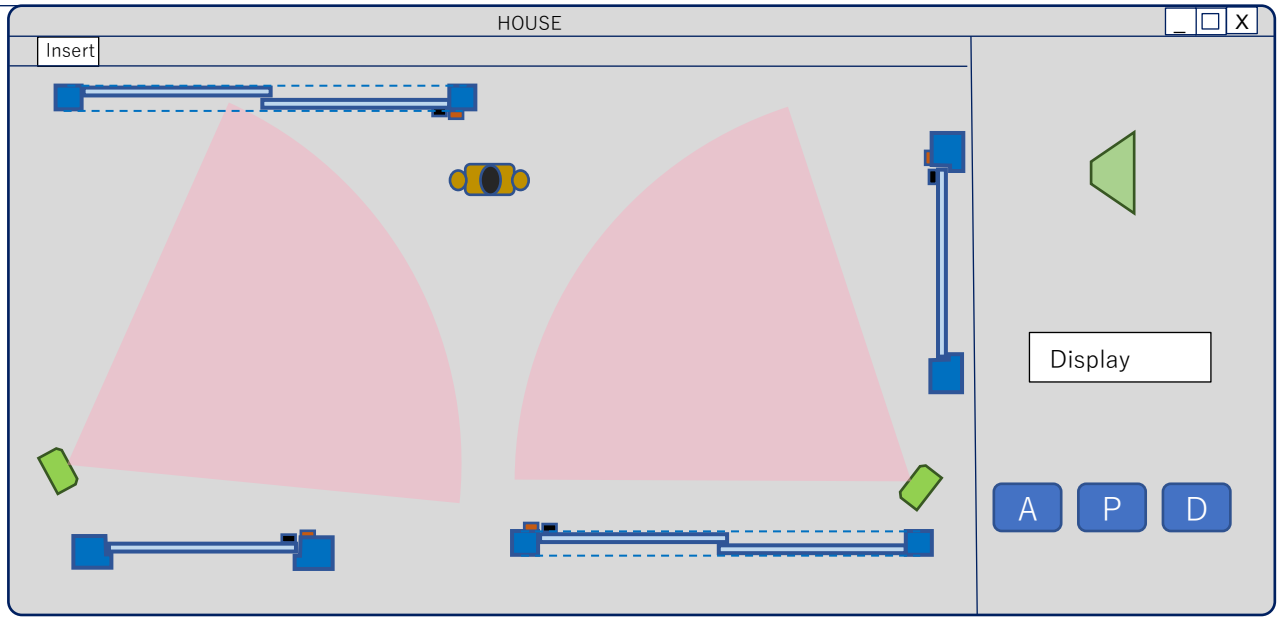

Figura 2: Apariencia pedida para la interfaz gráfica de usuario

En el sector derecho de la interfaz y separado por un separador (Separator de JavaFX) se muestra la interfaz de la central. Tiene dos botones para su activación: armado total “A”, armado perimetral “P” (sólo zonas 0 y 1) y desarmado “D”. En “Display” se muestra información de interés, por ejemplo, si al armar hay alguna ventana abierta, se indica la zona que no está cerrada. Cuando la alarma se dispara, Display debe indicar el número de la o las zonas abiertas. Para desarmar la alarma normalmente se debe ingresar una clave, por simplicidad esta característica no se pedirá en esta tarea.
La apertura y cierre de puertas y ventanas se logra con un click en botón izquierdo del mouse.
El Menú Insert permite agregar una persona en la casa, usted es libre de elegir dónde se ubica inicialmente. Presionando el botón izquierdo del mouse sobre una persona y arrastrando el mouse es posible mover a la persona por la casa. Un click del botón derecho sobre una persona presenta un menú de contexto que permite eliminarla.

4. Funcionamiento de la Aplicación

En esta tarea su programa leerá la configuración del sistema (puertas, ventanas, sensores, sirena, central, etc.) desde un archivo y a través de la interacción con la interfaz el usuario simulará el ingreso y movimientos de personas en la propiedad. Esta tarea no genera archivo de salida.
El formato para el archivo de entrada es rígido y el nombre del archivo es pasado como un argumento al ejecutar el programa. La primera línea define el número de puertas, ventanas y detectores de movimiento. Todas las puertas y ventanas están cerradas al crearse. Las siguientes líneas entregan los atributos para cada puerta y cada ventana creada. La coordenada (x,y) de ventanas y puertas corresponde al extremo superior izquierdo de cada una. Todas las puertas y ventanas son de tamaño fijo, su orientación está entre 0 y 360 º, e incluyen un sensor magnético asociado a la zona señalada en el archivo. Luego se lista la ubicación, orientación y rango de cada uno de los detectores PIR. Finalmente se define el nombre del archivo con el sonido de la sirena.
Su formato es:

```
<#_doors> <#_windows> <#_PIRs>
<x> <y> <direction_angle> <zone>
….
<x> <y> <direction_angle> <sensing_angle> <sensing_range> <zone>
...
 <siren_sound_file>
```

La primera línea es única. Luego vienen tantas líneas como puertas, ventanas y detectores PIR. Para cada uno de estos elementos se indica la posición de su centro geométrico y orientación en grados [º], para los sensores PIR le sigue el ángulo del cono sensor [º] y el rango de detección [**cm**]. Finalmente, todos los sensores especifican su zona (entre 0 y 2). La ubicación de todos estos elementos está referida a un plano R^2 usando **centímetros** como unidad de cada eje de ordenadas. La puerta principal debe ser asociada a la zona 0 y podría haber más de una. La última línea contiene el nombre del archivo a reproducir por la sirena en señal de alerta.

Un ejemplo para el contenido del archivo de configuración es:

```
2       1       3
20      400     0       0
500     200     270     1
200     400     180     1
0.0     0.0     45      60      500     2
1000    400     225     90      400     2
500     500     135     30      500     2
siren.wav
```

Llamando a su programa GraphicAlarmTest.java, la ejecución de su tarea sería del tipo (esto se correrá desde IntelliJ):
$ java GraphicAlarmTest config.txt

Una gran diferencia con la Tarea 1 la genera el nuevo paradigma usado en Tarea 2: “programación conducida por eventos”. En la Tarea 1, el método main activaba todos los cambios del programa. En esta tarea los eventos del usuario y los que usted programe gobernarán la ejecución del programa.

5. Desarrollo en Etapas

Para llegar al resultado final de esta tarea usted debe aplicar una metodología de desarrollo "Iterativo y creciente (o incremental)" para desarrollar software. Usted y su equipo irán desarrollando etapas donde los requerimientos del sistema final son abordados gradualmente. En cada etapa usted y su equipo obtendrá una solución que funciona para un subconjunto de los requerimientos finales o bien es un avance hacia ellos. Esto tiene por objetivo familiarizarse con la metodología de desarrollo iterativo e incremental.

5.1. Primera Etapa: Una ventana (sin central ni sirena)

En esta primera etapa se deben crear las clases Sensor, MagneticSensor, Window, MagneticSensorView y WindowView. Cree el programa Stage1.java para crear una ventana en la interfaz a partir de la definición hecha en el archivo de configuración (con valor 0 para el número puertas y detectores PIR). La última línea del archivo de entrada se puede omitir (no hay sirena).
Verifique que la interacción del usuario con la ventana permite su apertura y cierre usando una animación vía una subclase de Transition.
Entregue las clases de esta etapa y el archivo de entrada usado.

Use y/o complete el código inicial para esta etapa. Aquí está el diagrama de clases para este diseño. Usted no está obligado(a) a usar estos códigos. Están para su conveniencia y en caso de que le resulten útiles. Otras formas de estructurar el resultado hasta llegar a la etapa 4 también son válidos.

5.2. Segunda Etapa: Propiedad con dos puertas, dos ventanas, central y sirena

A las clases de la etapa previa incorporar las clases Door, DoorView, Central, CentralView, Siren, SirenView, las cuales siguen la lógica descrita para ellas previamente. Como archivo de sonido de la sirena, puede usar éste. Cree el programa Stage2.java para esta etapa y verifique que la interacción con el usuario genera los resultados esperados (apertura de puerta, armado/desarme de alarma, sonido de alarma, cambio de color de sirena).
Entregue todas las clases de esta etapa y el archivo de entrada usado.
Use y/o complete el código inicial para esta etapa.

5.3. Tercera Etapa: Agrega Detectores PIR y una persona movible arrastrando el mouse

Cree la clase PIR_Detector, PIR_DetectorView, Person y PersonView. En esta etapa la central sólo permite el armado total (A). Una Persona tiene una posición (la del centro de su cabeza) y una vista como la mostrada en Figura 2. Un detector PIR detecta a una persona cuando su centro ingresa a su región de detección.
Cree el programa Stage3.java y verifique que la interacción con el usuario genera resultados acordes (movimiento de personas, detectores PIR operan correctamente, etc.).
Entregue todas las clases de esta etapa y el archivo de entrada usado.

5.4. Cuarta Etapa: Cumplimiento de las funcionalidades de la alarma según sección 4

En esta se da cumplimiento a la totalidad de la especificación de la tarea, por ejemplo, incluyendo el menú de la ventana principal para agregar personas, menú de contexto para eliminar personas, armado perimetral (nocturno, sin zonas sólo asociadas a sensores PIR), etc.
En esta etapa su grupo debe documentar usando notación compatible con utilitario javadoc las clases Sensor y PIR_Detector.

5.5. Extra-crédito: Permitir la salida y entrada con armado y disparo retardado

Opción A

- A1(5 puntos extras): Modifique las clases necesarias para que, al posicionar el mouse sobre una ventana o puerta, ésta cambie de color. Así el usuario sabrá que está en condiciones de cerrarla o abrirla.
- A2 (5 puntos extras): Agregue un menú de contexto a cada sensor para permitir cambiar la zona a la que pertenece.

Opción B

(10 puntos) Enviar un correo ante el disparo de la alarma. Para esto usted debe revisar el ejemplo disponible aquí. Programe su aplicación para que ésta lea desde el archivo de configuración, después del archivo de sonido de sirena, la cuenta de correo a usar para el envío del correo y en la línea próxima el correo destino de la alerta. Cuando se lea el archivo de entrada, la aplicación pedirá, vía un campo de texto tipo PasswordField, la clave del correo para enviar la alerta.

Extra-crédito es voluntario, su desarrollo otorga 10 puntos adicionales (la nota final se satura en 100). Si desarrolla esta parte, indique en su readme qué opción tomó.

6. Elementos a considerar en su entrega

En AULA habrá un buzón para subir el readme y la documentación. En el readme indique todas las etapas desarrolladas y el proyecto Git para acceder al código de su entrega. **Su equipo deberá entregar una solución para cada una de las etapas** aun cuando la última integre las primeras. **El readme y archivo de documentación deben ser preparados sólo para la última etapa desarrollada. En su readme indique cómo generar la documentación con javadoc.** Independiente del IDE utilizado, considere un directorio por etapa desarrollada, ponga en él los archivos *.java, config.txt usado y su makefile.
Prepare un archivo makefile para compilar y ejecutar su tarea con rótulo “run”. Defina una variable para registrar el directorio donde está la carpeta lib de JavaFX la cual es requerida para para compilar y ejecutar programas JavaFX. La idea es que, cambiando el valor de ese parámetro en la línea de comando, la o el ayudante pueda compilar su tarea según la ubicación donde ella o él tiene JavaFX. Además, incluya rótulos "clean" para borrar todos los .class generados. Los comandos a usar en cada caso son:

```
$ make    /* para compilar*/
$ make JFX_OPTIONS=”/path/to/javafx/lib” /* compilación cambiando el valor de variable */
$ make run  /* para ejecutar la tarea */
$ make clean  /* para borrar archivos .class */
```

En su archivo de documentación (pdf o html) incorpore el diagrama de clases de la aplicación (última etapa)
Revise lo indicado en “Normas de Entrega de Tareas”, en caso de diferencias, lo señalado en este escrito tiene precedencia.

7. Sobre la arquitectura Modelo Vista Controlador

Para organizar interfaces gráficas una "solución de software general recomendada" (éstas son conocidas como patrones de diseño) es el patrón "modelo-vista-controlador".

Un **modelo** es una clase que caracteriza a un objeto y almacena los datos significativos de éste. Por ejemplo, en este caso la clase MagneticSensor maneja el modelo de un sensor magnético. Por otro lado, tenemos las **vistas**, estas clases indican cómo un objeto se muestra visualmente. En esta tarea la clase DoorView contiene los elementos gráficos para generar la apariencia visual de una puerta.
En otras situaciones puede ocurrir que un mismo modelo tenga varias vistas. Otro ejemplo, fuera de esta tarea, es un objeto termómetro con un modelo y tres vistas: columna de mercurio, número digital, o intensidad de color.
Finalmente tenemos el **controlador** del objeto gráfico. Las clases controladoras son aquellas que modifican los datos, por ejemplo, a través de las acciones del usuario en la interfaz. Generalmente las clases controladoras corresponden a los "handlers" o “listeners”, es decir, los manejadores de los eventos que usted estima de interés.
Una clase puede cumplir dos roles, por ejemplo, ser **modelo** y **vista**. Así como podemos tener varias vistas, es posible tener varias clases controladoras de un modelo. Recuerde que implícito en cada expresión lambda hay una clase anónima. Usted puede identificar clases controladoras observando la clase de los objetos que atienen los eventos de la interfaz.