Diseño y Programación Orientados a Objetos
1er. Semestre 2022
**Tarea 2: Simulador Gráfico de Dispositivos Domóticos**

Lea detenidamente la tarea. **Si algo no lo entiende, consulte. Si es preciso, se incorporarán aclaraciones**. Esta interacción se asemeja a la interacción entre desarrolladores y clientes cuando algo no está del todo especificado.

**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".

## 1.- Descripción General

Esta tarea busca adaptar y extender la tarea 1 para simular gráficamente el modelo de un sistema domótico. En esta tarea usted modelará y programará un tipo de cortina motorizada (aquí un ejemplo) y un tipo de lámpara. Por completitud de la tarea, se pasa a describir una cortina motorizada y una lámpara controlada.

### Modelo de una cortina "roller" motorizada

Aquí, el modelo de la cortina es simple, incluye un motor capaz de girar en ambas direcciones y una tela, Figura 1. Cada cortina está asociada a un canal de trabajo. Para fijar la apertura de la cortina se dispone de un control con botones para enviar mensajes "up", "down" y "stop". Cada control remoto de cortina tiene asociado un canal y a través de éste se define las cortinas que puede controlar. El canal de cada control se muestra en su parte central y coincide con su botón de detención. Para cambiar de canal se debe presionar los botones laterales. Al ser instaladas (al "nacer") cada cortina parte talmente enrollada; es decir, 100% abierta.


Figura 1: Modelo simple de cortina motorizada y su control remoto

Cuando se acciona una vez el botón para subir la cortina (botón superior del control o tecla con flecha "UP" se presiona), el motor comienza a girar en sentido correspondiente y como resultado de esto la tela extendida se reduce. Para detener el giro del motor se debe presionar el botón central del control (o la tecla con fecha "UP" es soltada). Un control basta para controlar todas las cortinas de un mismo canal. Este tipo de cortinas tiene sensores "de fin de carrera" que detienen el motor cuando la tela está completamente enrollada y cuando está completamente extendida. Para no complicar el modelo, parte del estado de la tela será su nivel de apertura y será la tela la que detenga el giro del motor según cuando no pueda seguir abriéndose o cerrándose.

La rapidez angular del motor es un parámetro de éste (alfa). Su radio es fijo e igual a 2 [cm] para todos los motores de cortinas.

El largo total de la tela ("Length" o "L") y el ancho (Width o W) son parámetros y pueden variar de cortina a cortina.
Por simplicidad cada giro del motor extenderá o reducirá la tela un mismo valor ($2\pi r$). Notar que éste debería depender del radio efectivo de la tela enrollada el cual cambia según cuanta tela esté enrollada.
Detrás de la cortina habrá una ventana modelada como un rectángulo en el cual se podrá observar una vista cambiante del fondo (video repetitivo). Note que surge un nuevo objeto a modelar: la ventana. Ésta puede ser modelada como un rectángulo que muestra un video repetitivo en su interior, luego es posible ubicar una cortina sobre la ventana.

**Modelo de una lámpara controlada**
La lámpara de esta tarea permite ser encendida y su luz cambia de color a través de un control remoto, Figura 2. La luz se compone de tres colores básicos rojo, verde y azul, los cuales pueden ser ajustados para generar la percepción de otros colores. Cada color básico es representado por un valor que varia de 0 a 255. La intensidad es lograda variando el nivel de cada uno de los colores. Cuando una lámpara es instalada ("nace"), parte apagada y con intensidades 255 para cada color.

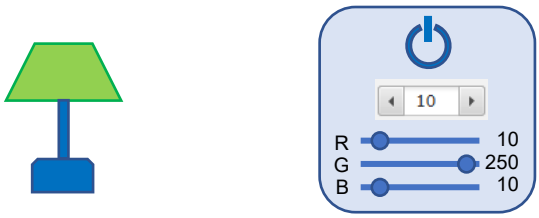

Figura 2: Modelo de una lámpara domótica y su control remoto

Cada lámpara tiene un canal asociado el cual no se mezcla con los canales de las cortinas. Un control de lámpara asociado a un canal controlará todas las lámparas de ese canal. Cada control permite cambiar su canal y así con un control controlar múltiples lámparas (use un Spinner). En Figura 2 se aprecia cómo bajo el botón de power se muestra el canal actual el cual se puede aumentar reducir los controles laterales.
Al encender una lámpara (botón "power" o tecla espacio)[1], ésta ilumina con la intensidad y color que tenía previo a ser apagada. Se apaga con mismo botón ("power" o tecla espacio).
Para cambiar el nivel de cada color, se usa los controles mostrados (Slider). No existen intensidades inferiores a 0 o superiores a 255 para cada color.

**"La Nube"**
La nube no corresponde a un objeto físico sino a una aplicación corriendo en algún servidor remoto. Cada aparato controlable domóticamente se registra en la nube y puede ser controlado por el control de ese canal. Así, para cambiar un dispositivo domótico, un control envía mensajes a la nube y ésta los envía a cada dispositivo domótico.

**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 3. No se pide dar flexibilidad al número de elementos que la componen.

[1] Cuando los controles tienen botón único de encendido y apagado, éste funciona conmutando el estado actual.

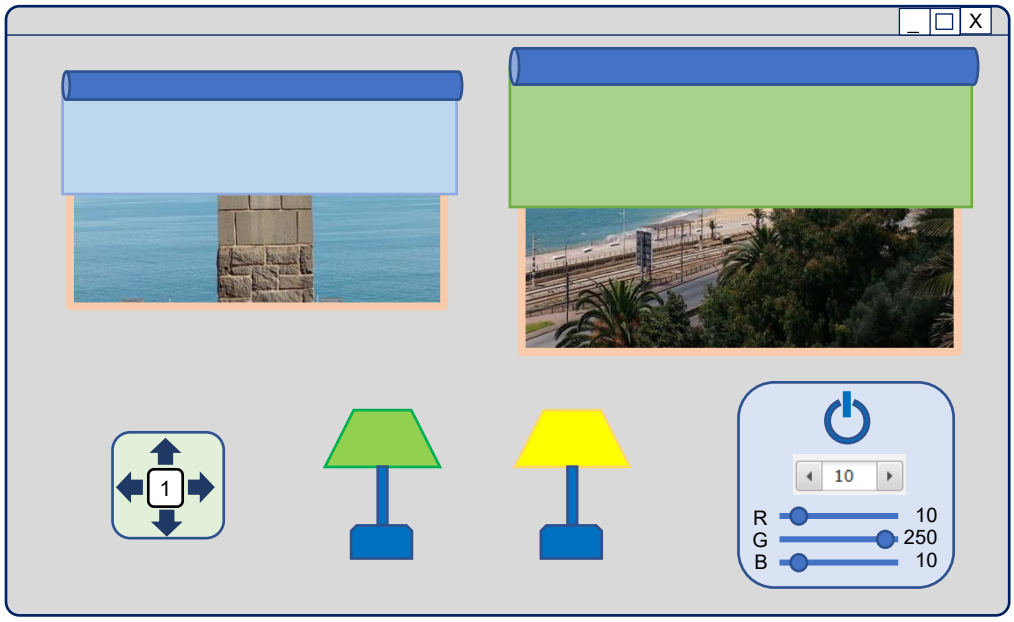


Figura 3: Apariencia pedida para la interfaz gráfica de usuario

Llamando a su programa GraphicDomoticsTest.java, la ejecución de su tarea sería:
$ java GraphicDomoticsTest

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 start activaba todos los cambios del programa. En esta tarea los eventos del usuario y los que usted programe gobernarán la ejecución del programa. En su solución trate que el programa emule lo que ocurre en la realidad.

**2.- Desarrollo en Etapas**
Para llegar al resultado final de esta tarea usted debe aplicar una metodología de tipo "Iterativa e Incremental" para desarrollo de 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. En AULA se dispondrá el recurso para subir la solución correspondiente a cada etapa del desarrollo. **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 solo para la última etapa. Prepare y entregue un makefile para cada una de las etapas.** Esto tiene por finalidad, educar en la metodología iterativa e incremental.

**2.1.- Primera Etapa: Una lámpara con encendido On/Off**
En esta etapa se pide crear una interfaz gráfica conteniendo sólo una lámpara y un control remoto que contendrá sólo el botón On/Off. El usuario podrá cambiar el color de la pantalla de la lámpara entre negro y algún color.
Use y/o complete las clases Lamp, LampView, LampControl, LampControlView, Cloud y Stage1 disponibles aquí. 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.
La clase Stage1 contiene el método main, en ésta está la lógica para crear la interfaz gráfica de usuario. Este método crear todas las instancias necesarias de esta etapa.
Entregue las clases de esta etapa con su correspondiente makefile.

### 2.2.- Segunda Etapa: Una cortina de apertura y cierre controlable

Extienda lo desarrollado en la etapa previa agregando en la parte superior una cortina y un control remoto de cortina. Vía programa ambos trabajan en el mismo canal. El control cuenta con los botones para subir, detener y bajar la tela de la cortina. Detrás de la cortina hay un rectángulo de color fijo elegido por usted.
Use y/o complete las clases RollerShade, RollerShadeView, Motor, ShadeControl, ShadeControlView y Stage2 disponibles aquí.
Entregue las clases de esta etapa con su correspondiente makefile.

### 2.3.- Tercera Etapa: Dos lámparas y su control con funcionalidad completa

Para esta etapa se pide completar las opciones del control de lámparas y ubicar dos lámparas en canales 2 y 3. El control remoto debe poder controlarlas ajustando el color de su pantalla según la manipulación del usuario sobre el control. La interfaz gráfica debe extender aquella de la segunda etapa. La funcionalidad de la etapa previa debe mantenerse en ésta.
Entregue las clases de esta etapa e incluya su makefile.

### 2.4.- Cuarta Etapa: Simulador gráfico de dispositivos domóticos

En esta última etapa se espera alcanzar la funcionalidad descrita en la sección 1. La interfaz gráfica de usuario debe asemejarse a la Figura 3. Para su comodidad, puede usar los siguientes dos videos como fondos de cada ventana (1:1, 16:19).

### 2.5.- Extra-crédito: Manejando dos vistas para una lámpara

Usted gana 10 puntos extras pero su tarea se satura en 100 si el total del puntaje fuera mayor.
Cree una segunda vista para una lámpara y agregue un menú de contexto para cambiar la vista de la lámpara entre la mostrada en esta tarea y otra que usted elija.

## 3.- Elementos a considerar en su documentación

Entregue todo lo indicado en “Normas de Entrega de Tareas”.
Prepare un archivo makefile para compilar y ejecutar su tarea con rótulo “run”. Defina una variable JavaFX 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, el ayudante pueda compilar su tarea según la ubicación donde 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 JavaFX=”/path/to/javafx/lib” /* compilación alternativa 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 (etapa 4).

## 4.- 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 Lamp maneja el modelo de una lámpara. Por otro lado, tenemos las **vistas**, estas clases indican cómo un objeto se muestra visualmente. En esta tarea la clase LampView contiene los elementos gráficos para generar la apariencia visual de una lámpara.
En otras situaciones puede ocurrir que un mismo modelo tenga varias vistas. Por ejemplo, como extra-crédito se pide crear otra vista para las lámparas. 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 (teclado y mouse).