Práctica 3: Implementación de máquinas de estado

1. Objetivos

a) Practicar la adaptación de un modelo de diseño a un modelo de implementación aplicando patrones de diseño.

b) Aprender a implementar una máquina de estados aplicando el patrón State.

c) Practicar el uso de frameworks basados en inversión de control.

2. Actividades

El alumno deberá generar una implementación del sistema `Alarmas` cuya especificación se detalla a continuación. Para ello, deberá seguir los siguientes pasos:

- Completar el modelo de diseño detallado disponible en Moodle (archivo Alarmas`.mdzip`) finalizando el diagrama de estados de la clase `Alarmas` que se proporciona incompleto. El diagrama de estados debe ser correcto, completo y conforme al diagrama de clases incluido también en el modelo (el diagrama de clases, en un principio, no debería ser modificado).

- Elaborar el modelo de implementación del sistema, refinando el modelo de diseño detallado, de manera que refleje las características propias de la implementación a desarrollar:
  - técnica de implementación de máquinas de estado elegida, que en este caso será el patrón `State`,
  - características derivadas del lenguaje de programación elegido (Java),
  - opcionalmente se pueden incluir todas aquellas clases auxiliares necesarias para implementar el código, como timers, clases activas, etc.

  Hay dos modos de realizar este paso de manera sencilla y correcta:
  - Hacer una copia del archivo MagicDraw y usar el diagrama de clases del modelo de diseño detallado como punto de partida del modelo de implementación.
  - Dentro del propio archivo MagicDraw, realizar una copia del paquete que contiene el modelo de diseño detallado y renombrarlo como modelo de implementación.

- Generar el código de los esqueletos de las clases del sistema a partir del modelo de implementación.

  Nota: Este paso se puede realizar de manera automática utilizando la versión Enterprise de MagicDraw. Si alguien está interesado en hacerlo de ese modo, tenéis información sobre ello en la "Guía de instalación del software de prácticas" o podéis consultar al profesor, sino deberéis hacerlo a mano.

- Implementar la funcionalidad completa de las clases de la aplicación.

- Implementar una interfaz gráfica que permita probar la aplicación, para cuyo desarrollo deberá utilizarse el patrón MVC (en el anexo tenéis un ejemplo de cómo podría ser dicha interfaz).
- Probar el funcionamiento del sistema. Este apartado es necesario enseñárselo al profesor una vez terminada la aplicación.

3. Entrega y evaluación

Se deberá entregar un archivo comprimido con la siguiente información:

- Archivo MagicDraw con el modelo de diseño detallado completo (diagrama de clases y diagrama de estado).
- Archivo MagicDraw con el modelo de implementación (puede ser el mismo archivo que el anterior, pero con cada modelo en paquetes independientes).
- Proyecto Maven exportado con el código correspondiente a la aplicación. Para generarlo:

  `Botón derecho en el proyecto => Export => General => Archive File`
- Archivo .jar ejecutable correspondiente a la aplicación.

El nombre del archivo comprimido deberá ser `Practica1-<Nombre>-<Apellidos>`.

Se valorará:

(1) La completitud y coherencia del modelo de diseño detallado (1.5 puntos).
(2) La completitud y corrección del modelo de implementación, especialmente la aplicación del patrón State (1.5 puntos).
(3) La implementación desarrollada, incluyendo la completa funcionalidad, la coherencia con el modelo de implementación, la implementación del patrón State, la gestión de los eventos temporizados y el estilo general de programación (6 puntos).
(4) La correcta aplicación del patrón MVC (1 punto).

Parte opcional (sube nota):

(5) La utilización de la interfaz TimedState y el objeto TimedStateController para la gestión de eventos temporizados. Suma 0.5 puntos a la práctica.

El documento se entregará a través de la plataforma *Moodle* de acuerdo a las fechas establecidas en la planificación de las prácticas.

Patricia López Martínez

CASO DE ESTUDIO: ALARMAS

Se quiere desarrollar una aplicación que permita programar múltiples alarmas (parecida a la aplicación de alarmas de un móvil). La aplicación funciona del siguiente modo:

- Se pueden introducir cuántas alarmas se quieran (opción NuevaAlarma). Para introducir una nueva alarma se debe indicar la hora de la alarma y un nombre que la identifique. Cuando se introduce una nueva alarma, por defecto estará activada para sonar.
- Cualquier de las alarmas programadas se puede desactivar (seleccionándola y pulsando la opción AlarmaOff), pero sin borrarla del sistema, de manera que posteriormente se puede volver a activar (seleccionándola y pulsando la opción AlarmaOn).
- Existe también la posibilidad de eliminar de manera permanente cualquiera de las alarmas almacenadas en la aplicación (tanto si están activas como si no), a través de la opción EliminarAlarma.
- Cuando se alcanza la hora de la alarma más próxima, el móvil hace sonar una melodía predeterminada. Esta melodía suena durante un intervalo predeterminado pudiéndose parar primero pulsando el botón Apagar.
- Mientras la aplicación suena, el resto de opciones están deshabilitadas.

En la figura 1 se muestra un ejemplo de interfaz gráfica a través de la cual utilizar y probar nuestro sistema de alarmas.

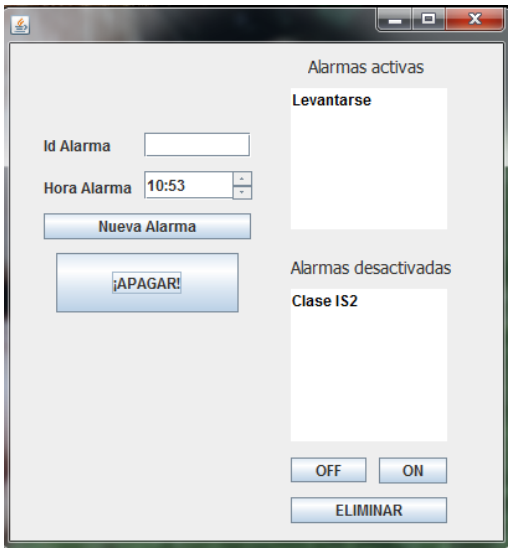

Figura 1. Esquema de la interfaz gráfica para la prueba de la aplicación a desarrollar

Aspectos a tener en cuenta en la implementación:

- En los métodos de negocio activarMelodia, desactivarMelodia de la clase Alarmas basta con escribir un mensaje por pantalla que indique el método invocado.
- En la interfaz gráfica se pueden hacer todas las modificaciones que se quieran para hacerla más rica o más estética, siempre que tenga la funcionalidad necesaria.