Programación de aplicaciones. Diseño basado en componentes

Ejercicios

Requisitos previos

- Tema 1.Diseño basado en componentes
- Anexo I. Definición de interfaces Java
- Conocimientos de programación orientada a objetos

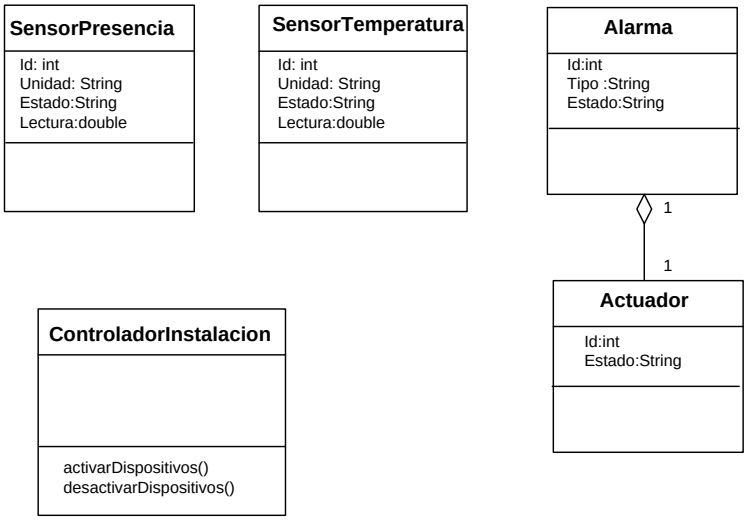
Objetivos de la práctica

- Conocer cómo tratar abstracciones
- Introducir al alumno en el diseño de colaboraciones entre clases usando interfaces

Índice

EJERCICIO PRÁCTICO 1

Dado un sistema Domótico, donde existe una clase denominada **ControladorInstalación** que controla los distintos dispositivos de la instalación (sensores, alarmas, actuadores, etc).



Se pide :

a) Realice un diseño **basado en el uso de interfaces** de forma que la clase **ControladorInstalacion** pueda activar o desactivar **a cualquier tipo de dispositivo** de la instalación existente y que en un futuro se pueda incorporar a la instalación
b) Realice el diagrama UML estructural del diseño anterior

Cuestiones:

1.-Donde se encuentra el punto de variabilidad en este diseño

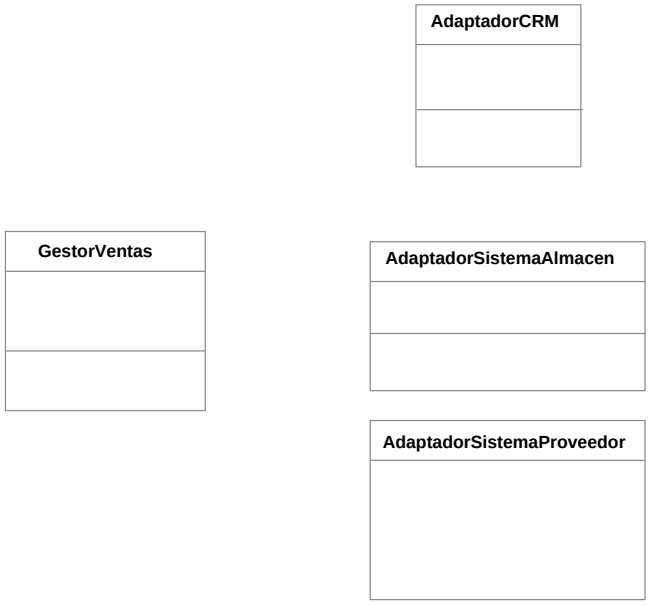
2.-¿Qué efectos provocaría en el diseño del **ControladorInstalacion** la introducción de un nuevo tipo de dispositivo?

3.- ¿ Se está haciendo uso en esta práctica del polimorfismo?. Explique de qué forma y donde

4.- Realice un diagrama del comportamiento del sistema cuando el operador da la orden al sistema de activar dispositivo.

EJERCICIO PRÁCTICO 2

Se desea proporcionar una manera **de integrar un Sistema central (GestorVentas)** con una **serie de sistemas externos**. Los sistema externos son: un sistema CRM (Custom Relationship Customer) que se encarga de la gestión de la información de todos los clientes de la compañía, otro es un sistema de registros de proveedores y el último es un sistema de gestión de almacén. La siguiente figura muestra las abstracciones realizadas. Se debe tener en cuenta que los sistemas anteriores no pueden ser modificados.



Para todos ellos existe una lógica común que usa el sistema GestorVentas y que consiste en las siguientes operaciones

- Como todos son sistemas independientes para cada uno se debe desarrollar la operación *conectar()*
- Para todos ellos existe la operación **obtenerInfo (String consulta)** donde consulta es una expresión que indica la información a obtener de sistema remoto
- Para todos se debe poder ejecutar la operación *cerrar()*

Se pide

Dadas las clases denominadas G**estorVentas, AdaptadorCRM, AdaptadorSistemaAlmacen y AdaptadorSistemaProvedor**

a) Proporcione solución de diseño de forma que el sistema GestorVentas integre los sistemas externos
b) Realice un diagrama UML del diseño obtenido
c) Estudie el acoplamiento entre el sistema y los sistemas internos.