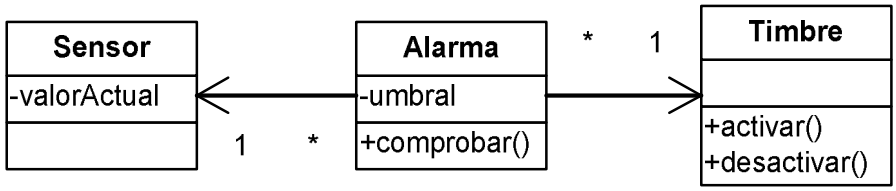
Programación orientada a objetos
Relación de ejercicios

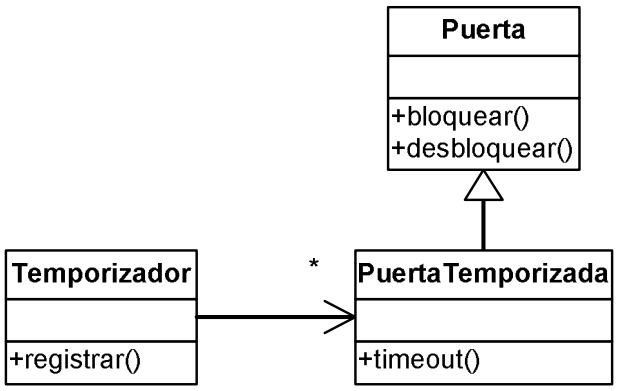
1. En las jerarquías de clases que haya creado hasta ahora, analice detenidamente si alguna de sus clases ha de ser abstracta. En particular, fíjese en la existencia de métodos que no deberían implementarse en una clase base y habrían de declararse, por tanto, como métodos abstractos. ¿Alguna de las clases que obtiene es completamente abstracta y debería convertirse en una interfaz?

2. Al estudiar clases y objetos, vimos cómo diseñar una alarma que conectábamos a un sensor y activaba un timbre:



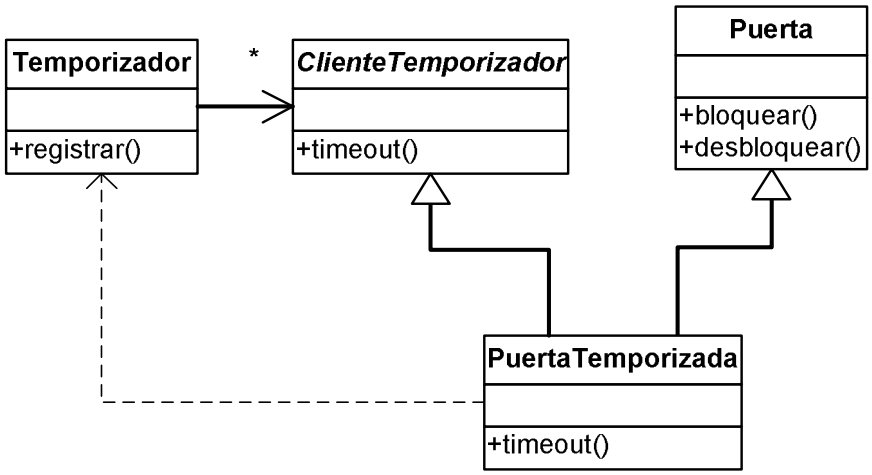
   Si queríamos que la alarma también encendiese una señal luminosa, lo que hacíamos era crear una subclase de `Alarma`. La herencia y el polimorfismo nos permite crear distintos tipos de alarmas que comparten parte de su comportamiento con la clase base `Alarma`. Ahora bien, si queremos añadir nuevos dispositivos conectados a la alarma (p.ej. un teléfono que automáticamente llama a la policía) y queremos que la alarma se pueda configurar de forma flexible, ¿se le ocurre alguna forma de hacerlo usando interfaces? Implemente la solución en Java.

3. Implemente en Java el sistema de seguridad asociado a una puerta con temporizador, tal como se muestra en la siguiente figura:

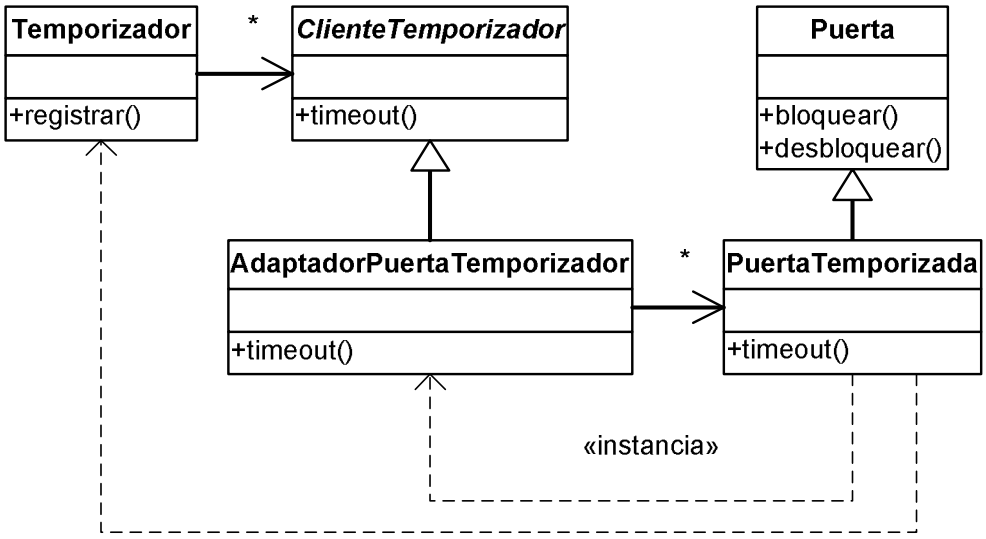


4. Generalice el diseño anterior para que el temporizador pueda activar distintos dispositivos aplicando el principio de segregación de interfaces:

   a. Usando herencia múltiple (de interfaces)



   b. Usando delegación (mediante un adaptador)



Implemente en Java todo el código asociado a los dos diseños propuestos.

A continuación, implemente en Java, utilizando las dos variantes descritas, el código necesario para que, por ejemplo, el temporizador cierre active automáticamente el sistema de riego del césped y, simultáneamente, cierre las persianas de la casa que dan al jardín…