Práctica 3

**Ejercicio 1**

Tenemos 3 clases (*Profesional*, *Expediente* y *Paciente*), un enumerado (*TipoAcceso*), 3 relaciones (1 entre *Profesional* y *Expediente* y 2 entre *Expediente* y *Paciente*) y una clase de asociación (*Acceso*).

El enumerado *TipoAcceso* quedaría implementado así:

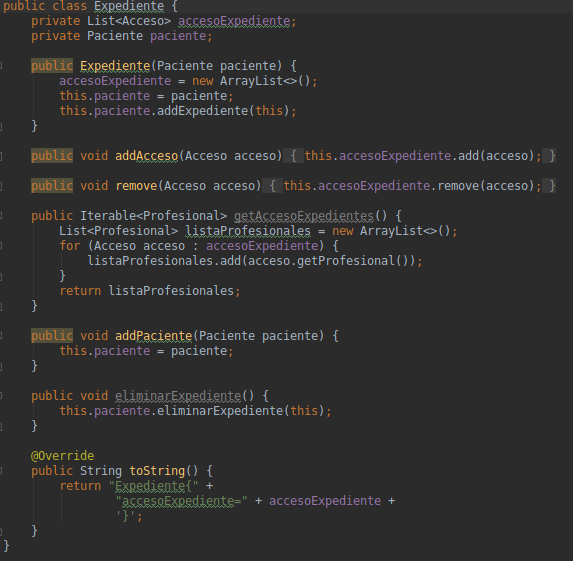


Las clases *Profesional*, *Expediente* y *Paciente* quedarían así:

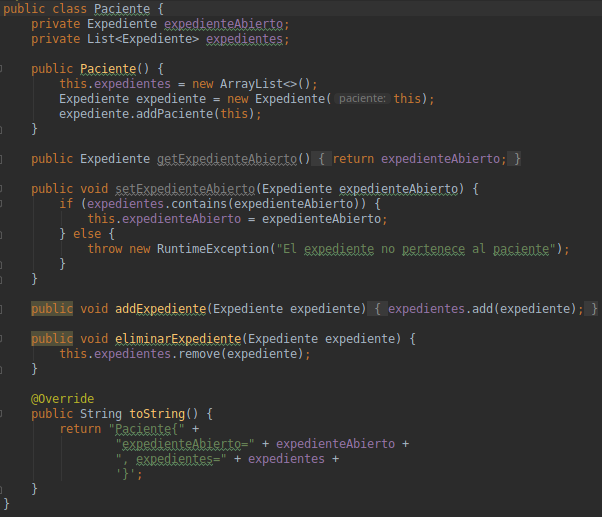
* *Profesional*.



* *Expediente*.



* *Paciente*.



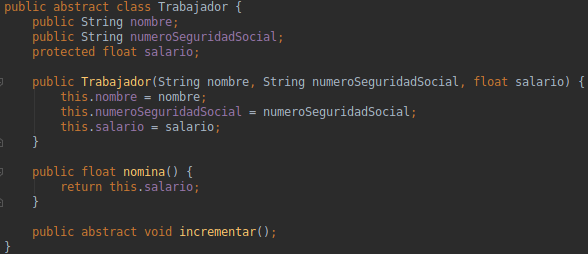
Las relaciones se implementan de la siguiente forma:

* **Relaciones *Expediente - Paciente***: hay 2 relaciones, la relación de 1 paciente a 1..\* expedientes se ha implementado con una lista de *Expedientes* en la clase *Paciente* (List<Expediente> expedientes) )y con un *Paciente* en la clase *Expediente* (Paciente paciente). Por otro lado, la otra relación entre ambas clase se ha implementado añadiendo un atributo de *Expediente* a la clase *Paciente* (Expediente expedienteAbierto)
* **Relación *Profesional - Expediente***: la única relación entre ambas clases es una relación de asociación, cuya clase de asociación es Acceso. Para esto solo hay una posible solución, implementando la clase Acceso como una clase aparte y haciendo que se encargue de la consistencia entre Profesional y Expediente, ya que, como indica el enunciado, “[...] *se puedan conocer los diferentes accesos a cada expediente y también conocer los expedientes a los que ha accedido un determinado trabajador del hospital* [..]*”*, es decir, existe bidireccionalidad*.*

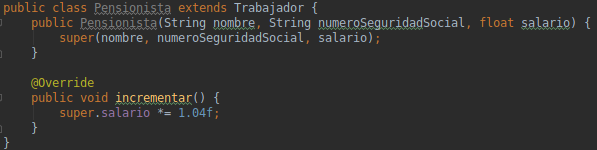
**Ejercicio 2**

1. Las clases no se pueden implementar directamente en Java puesto que no existe herencia múltiple.
2. La solución propuesta es una clase MedioPensionista que herede de Trabajador implementando el método abstracto y teniendo dos variables, una variable de la clase Activo con un salario y otra de la clase Pensionista, con otro salario distinto. el cálculo del salario es la suma de ambos, y al incrementarlo se incrementa cada salario por su porcentaje.
3. Implementación:

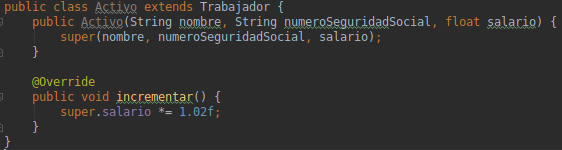
* Clase Trabajador:



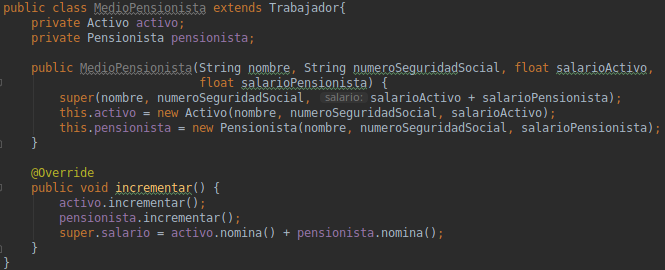
* Clase Pensionista:



* Clase Activo:



* Clase MedioPensionista:



**Ejercicio 3**

Las 3 formas de realizar el código son:

* **Implementación condicional.**

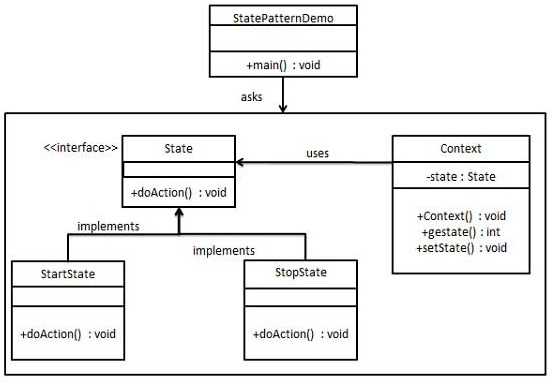
Realiza comprobaciones de las guardas dentro de la operación mediante bloques if - else.

* **Implementación mediante tabla de estado.**

Implementa una tabla mostrando cual es el siguiente estado dependiendo del evento entrante.

* **Implementación mediante el patrón de diseño Estado.**

En el patrón de Estado, el comportamiento de una clase cambia según su estado. Este tipo de patrón de diseño viene bajo patrón de comportamiento. En este patrón creamos objetos que representan varios estados y un objeto de contexto cuyo comportamiento varía a medida que cambia su objeto de estado. Para implementarlo, creamos una interfaz de Estado que define una acción y después creamos las clases que implementan dicha interfaz como en el siguiente modelo:

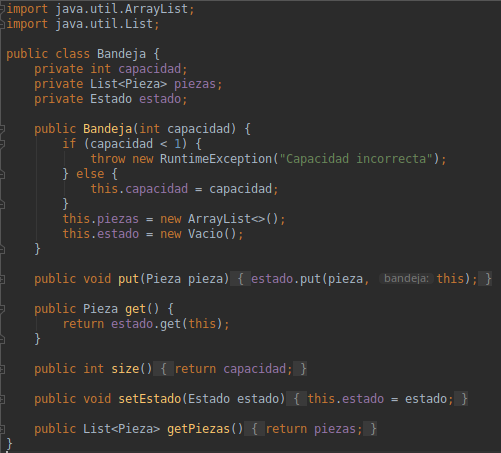


El código sería el siguiente:

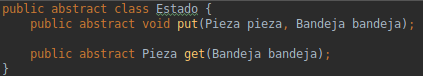
* ***Pieza***.



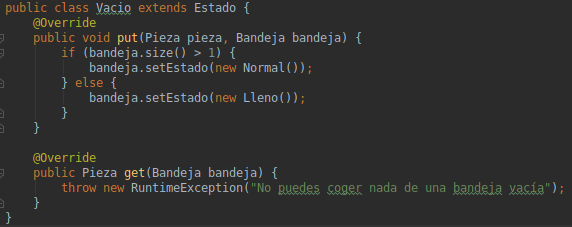
* ***Bandeja***.



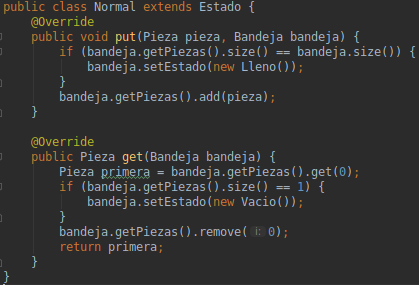
* ***Estado***.



* ***Vacio* hereda de *Estado***.



* ***Normal* hereda de *Estado***.



* ***Lleno* hereda de *Estado***.

