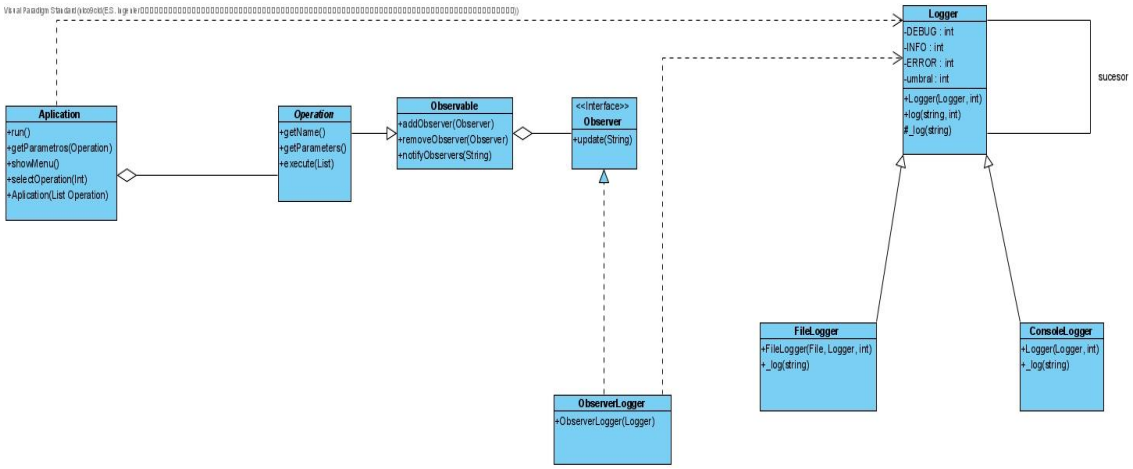
Prácticas entregables sobre patrones. GoF 3.

EJERCICIO 1:

Diagrama de clases:



Patrones utilizados : Chain of Responsability, Template Method, Command y Observer.

Chain of Responsibility: Se aplica en el sistema de log del framework. Nos permite pasar las peticiones a los distintos receptores (en nuestro caso serían ConsoleLogger y FileLogger) permitiendo que mas de un receptor pueda manejar la petición.

Template Method: Usamos este patrón en Logger, ya que nos permite definir el esqueleto de la operación log que delegará unos pasos concretos en las subclases (FileLogger y ConsoleLogger) que usarán el método log en las situaciones que lo pidan. Esto nos permitirá reutilizar código de manera que no se repita en las clases hijas de Logger.

Command: Este patrón se aplica en la clase abstracta Operation, que se pasa por parámetro a Aplicación. Las distintas clases que extiendan de Operation también se podrán pasar como parámetro. Esto nos permite desacoplar al que dispara la aplicación (Aplicación) del que tiene conocimiento de ejecutarla, en nuestro caso las distintas clases que extienden de Operation. Además, nos permite añadir fácilmente otras operaciones(comandos).

Observer: Usamos este patrón para poder notificar cuando una Operation avanza y así poder indicar el porcentaje que se lleva de esta,esto nos permite que las Operation no estén acopladas con los observadores.

Manual de Usuario:

El usuario puede crear distintas operaciones, para ello solo tiene que extender de la clase abstracta Operation e implementar los métodos abstractos de esta getName(), getParameters y execute().

El usuario puede crear una clase Aplication a la que se le tendrá que pasar una lista de Operations, una vez hecho esto si llama al método run se mostrará un menú con las distintas operaciones. Se podrá seleccionar cada una de las operaciones las cuales pedirán el número de parámetros que necesiten, además a medida que se ejecuta la aplicación se notificará al usuario del porcentaje de realización de esta, se enviarán distintos mensajes de log para notificar el estado de la operación.

El usuario también puede acceder al sistema de log del framework para ello solo tiene que inicializar una clase Logger a la que se le pasará un mensaje que tendrá un nivel que puede ser DEBUG,INFO Y ERROR. Las distintas opciones que da el framework es mostrar el mensaje por línea de comando (ConsoleLogger) y por fichero (FileLogger) , estas clases se inicializarán de manera que hay que pasarle el Logger y el nivel del mensaje.

EJERCICIO 2:

Utilizamos PROXY :Utilizamos este patrón ya que hay clases que queremos que se tenga acceso a ellas si se cumple una condición. En este caso el proxy se aplicaría en la clase Raiz teniendo en cuenta la clase LicenseManager que tiene el método para comprobar si la licencia es de pago. Para ello creamos una clase Operation (ProtectedRaiz) la cual a partir una clase “real” de raíz que se le pasa y comprueba la condición de checkIsFullVersion, de manera que dependiendo de lo que devuelva la condición se deje al usuario realizar la operación o no.