Ejercicios sobre patrones. GoF 1.

Ejercicio 1.

Se desea implementar un juego de estrategia en el que existen diversos **tipos de soldados** (zapador, que pone bombas, y artillero, que dispara) y varias **familias de ejército** (los Atreides y los Harkonen). Los jugadores pueden ser humanos o máquina, los cuales heredan de Jugador, que posee el método abstracto jugar(). Un Jugador tiene un ejército de 2 artilleros y de 2 zapadores inicial (una lista), que crea en el constructor (además podría crear más instancias de soldados en fases posteriores del juego, dentro del método jugar(), por ejemplo). Si el jugador es de la familia Atreides, deberá crear siempre soldados Atreides. Lo mismo si es de la familia Harkonen.

Se dispone de una [implementación del juego](https://moovi.uvigo.gal/pluginfile.php/946233/mod_assign/intro/juegoestrategia.zip). La clase principal del programa es JuegoEstrategia.

Se pide:

1. Analizar el código fuente proporcionado (se recomienda hacer un diagrama de clases para entenderlo).
2. Mejorar esta implementación con un patrón GoF, para que sea más fácil añadir una nueva familia de soldados y no afecte a la clase Jugador, evitando que sean los jugadores los que tengan que escoger la implementación al instanciar (Atreides, Harkonen, etc), además de facilitar que en un mismo Jugador no se mezclen soldados de distintas familias.

Entregar:

1. Diagrama de clases sólo de la versión ya refactorizada.
2. Código fuente refactorizado.

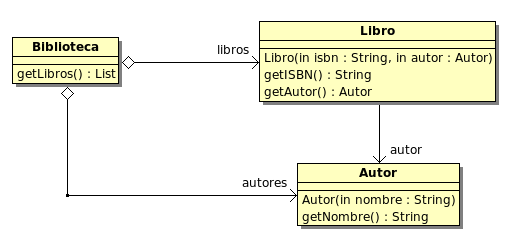
Ejercicio 2.

Se desea crear una aplicación que sea capaz de importar una serie de libros a partir de un fichero en el siguiente formato:

LIBRO<TABULADOR>AUTOR<TABULADOR>ISBN

Un ejemplo de entrada esta [aquí](https://moovi.uvigo.gal/pluginfile.php/946233/mod_assign/intro/libros.txt)

Los ficheros deberán ser importados dentro de la aplicación en una estructura de objetos que se detalla en el diagrama a continuación:



También se contempla la posibilidad de transformar ese fichero de texto en otra representación, consistente en un fichero xml de la forma:

<libros>

<libro>

<titulo>El quijote</titulo>

<autor>Cervantes</autor>

<isbn>2222</isbn>

</libro>

</libros>

Se pide:

1. Diseña el sistema con un diagrama de clases aplicando el/los patrón/es de diseño GoF que consideres más adecuado/s, teniendo en cuenta que se desean construir dos tipos de representaciones distintas y que el algoritmo de creación de las mismas (una iteración añadiendo libros) es la misma, independiente de la representación final. **NOTA: Se pueden añadir métodos a las clases del diagrama (sobre todo la clase Biblioteca)**.
2. Implementa el sistema en Java. El programa, en su método main y sin interacción con el usuario, deberá leer el fichero "libros.txt", crear la estructura biblioteca en memoria y crear el fichero "libros.xml".

**Entregar:**

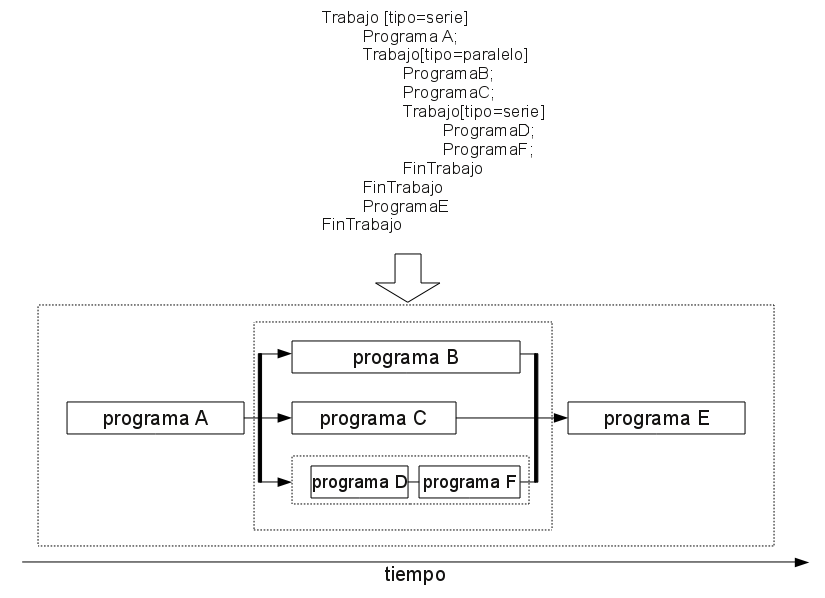
1. Diagrama de clases y justificación del patrón brevemente (PDF).
2. Código fuente (carpeta de eclipse).

Ejercicio 3.

Se desea crear un sistema genérico para la ejecución de trabajos en un cluster de computación (entendemos cluster simplemente un equipo que tiene un número considerable de procesadores y donde es factible lanzar muchos hilos simultáneamente). Los trabajos están compuestos de programas que son los que ejecutan tareas atómicas. Además, los trabajos pueden estar, a su vez, dividos en trabajos de forma recursiva. Los trabajos pueden ejecutar sus subtrabajos en serie o en paralelo, es decir:

* Trabajo en serie: Ejecuta cada uno de sus subtrabajos uno tras otro.
* Trabajo en paralelo: Lanza todos los subtrabajos en un hilo separado. Para que el trabajo termine, todos sus subtrabajos deberían haber terminado.

Por ejemplo, supongamos que hay implementados 6 programas básicos (A,B,C,D,E,F). El usuario desea ejecutarlos en el cluster siguiendo la siguiente disposición.



Como se puede observar en la figura anterior, un trabajo se puede ver como un árbol. De hecho, la descripción textual del trabajo podría ser un buen fichero de configuración que el usuario aportaría al sistema (no hay que implementarlo).

Los distintos programas están implementados en distintas clases que implementan la interfaz Runnable. Por ejemplo, el programa A está definido así:

public class ProgramaA implements Runnable{

public void run(){

try{

System.out.println("[INICIO] ProgramaA"); Thread.sleep(10); System.out.println("[FIN] ProgramaA");

}catch(InterruptedException e){ throw new RuntimeException(e);}

}

}

Información Java útil:

* Para ejecutar un Runnable en nuestro propio hilo, simplemente llamamos a su método run().
* Para lanzar un nuevo hilo en segundo plano que ejecute un Runnable:

Thread miRunnableThread = new Thread(miRunnable); miRunnableThread.start();

* Y para esperar a que termine:

miRunnableThread.join();

Se pide:

1. Diseñar el sistema empleando el/los patron/es GoF que consideres adecuados teniendo en cuenta que se podrán combinar los trabajos de forma distinta (según el fichero de configuración) y que la ejecución de un trabajo así como de un programa se debe realizar invocando el método run().
2. Implementar el sistema en Java y un método main que cree en memoria la estructura del trabajo de la figura y la ejecute. Deberá salir por pantalla la salida de cada programa. Una posible salida sería:

[INICIO] Programa A

[FIN] ProgramaA

[INICIO] ProgramaB

[INICIO] ProgramaC

[INICIO] ProgramaD

[FIN] ProgramaD

[INICIO] ProgramaF

[FIN] ProgramaF

[FIN] ProgramaC

[FIN] ProgramaB

[INICIO] ProgramaE

[FIN] ProgramaE

Según el trabajo propuesto por la figura, se debe cumplir:

1. ProgramaA debe finalizar antes de que empiece ningún otro.
2. ProgramaF debe comenzar después de que finalice ProgramaD.
3. ProgramaE debe comenzar después de finalizar los programas B,C,D,F

Entregar:

1. Diagrama de clases y breve justificación del patrón (PDF).
2. Código fuente (carpeta del proyecto eclipse).