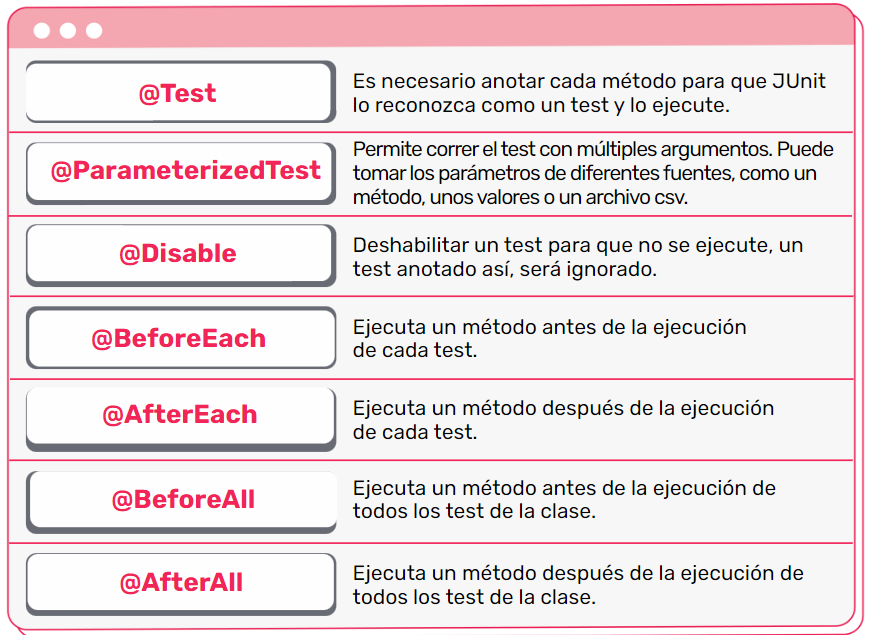
SEMANA 1

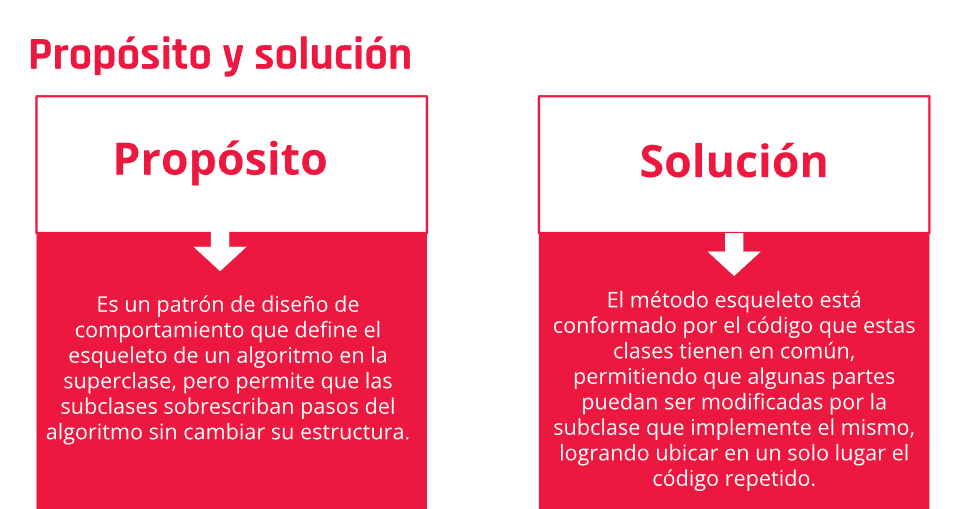
JUnit (http://junit.org) es el framework open source de testing para Java más utilizado. Nos permite escribir y ejecutar tests automatizados. Es soportado por todas las IDEs (Eclipse, IntelliJ IDEA), build tools (Maven, Gradle) y por frameworks como Spring.

Test en Junit:



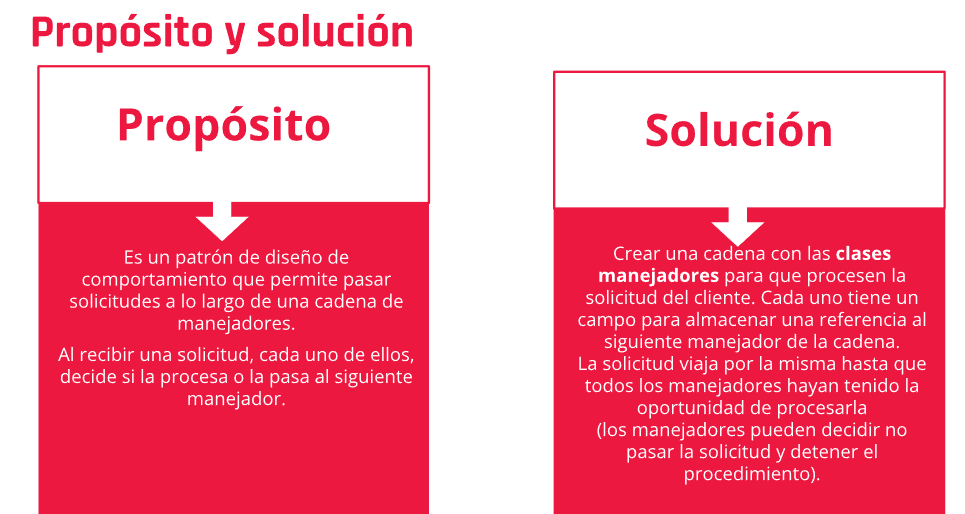
**PATRONES DE DISEÑO:**

PATRON TEMPLATE METHOD:



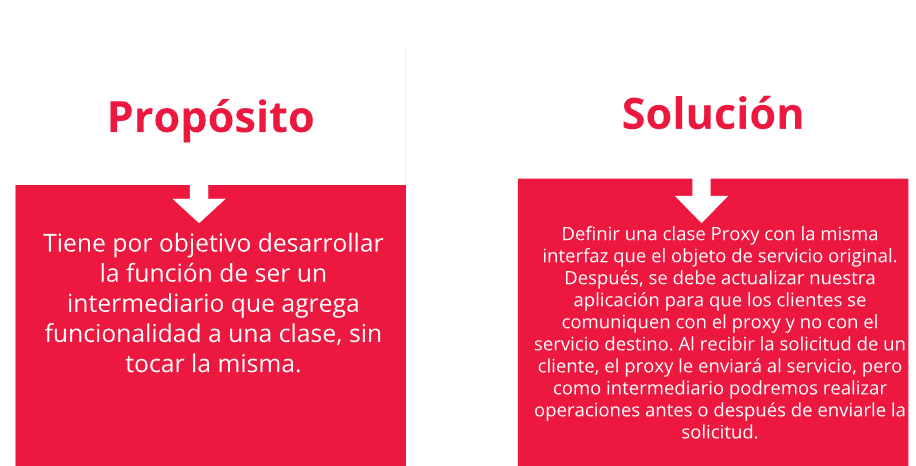
¿para qué nos sirve el patrón template method? Porque al eliminar el código repetido, nuestro código será más eficiente, legible y mantenible. Esto hará que sea más fácil de extender y mejorar.

PATRON CADENA DE RESPONSABILIDAD:



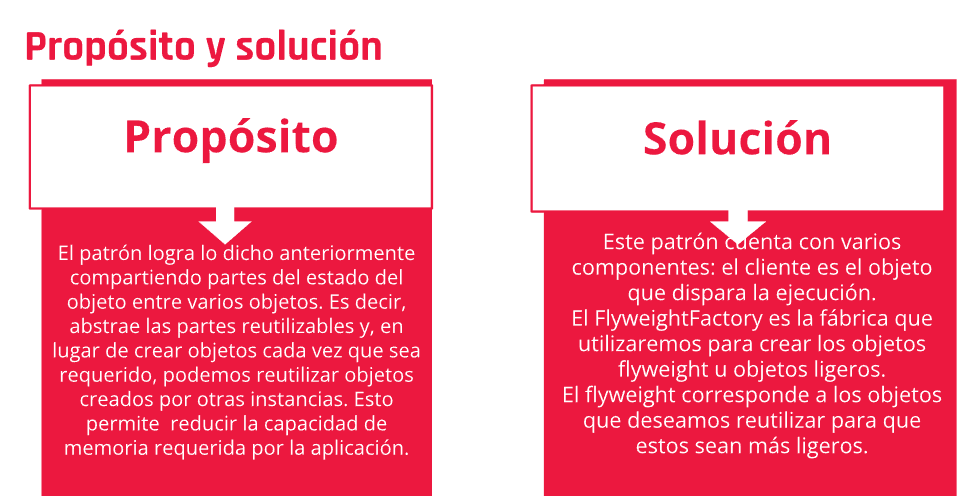
Entonces, podemos concluir que es beneficioso utilizar el patrón cadena de responsabilidad cuando esperamos que nuestra aplicación procese diferentes tipos de solicitudes y responda con diferentes resultados, pero no conocemos de antemano cuáles son estas solicitudes.

PATRON PROXY:



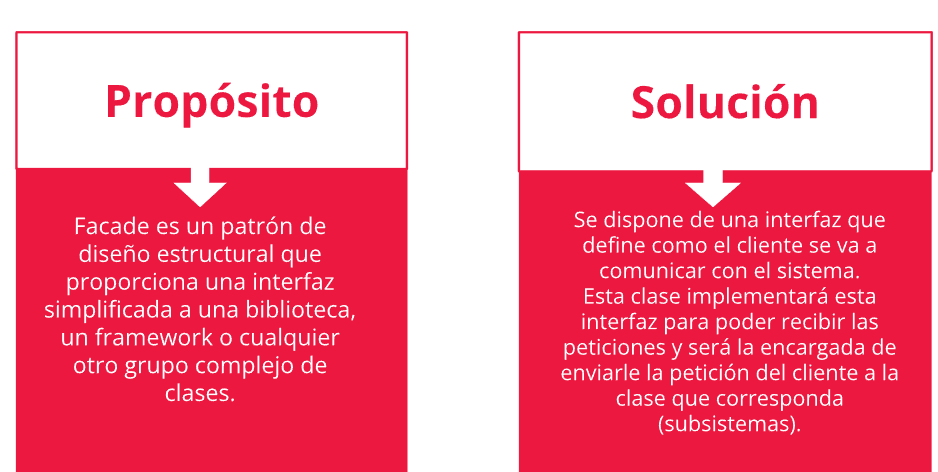
Entonces, podemos mencionar que es beneficioso utilizar el patrón proxy cuando queremos agregar funcionalidad adicional sin modificar los servicios actuales.

PATRON FLYWEIGHT:



Este patrón es utilizado cuando la optimización de los recursos es algo primordial, ya que elimina la redundancia de objetos con propiedades idénticas.

PATRON FACADE:



El patrón facade nos ayuda a reducir la complejidad de interactuar con un conjunto de subsistemas, actuando de intermediario entre el cliente y los subsistemas, permitiéndonos tener una única entrada, facilitando la comunicación entre estos.

SEMANA 2:

LOG4J: Es una librería desarrollada en Java por la **Apache Software Foundation** que permite a los desarrolladores elegir la salida y el nivel de granularidad de los mensajes o logs en tiempo de ejecución. En otras palabras, es utilizada para generar mensajes de **logging** de una forma limpia, sencilla, permitiendo filtrarlos por importancia y pudiendo configurar su **salida** tanto por **consola**, **fichero** u otras diferentes.

VENTAJA:

Permite tener un registro de lo que está pasando en nuestros sistemas, lo que nos posibilita entender mejor los errores.

DESVENTAJA:

La única desventaja es que a veces los archivos se hacen muy grandes y ocupan mucho espacio. Es por ello que debemos elegir bien qué tipo de información queremos almacenar

Por defecto Log4j tiene niveles de prioridad para los mensajes, entre ellos se encuentran:

* OFF: Este es el nivel de mínimo detalle, deshabilita todos los logs.
* FATAL: Se utiliza para mensajes críticos del sistema, generalmente después de guardar el mensaje, el programa se cierra.
* ERROR: Indica eventos de error que aún podrían permitir que la aplicación continúe ejecutándose.
* WARN: Se utiliza para mensajes de alerta sobre eventos.
* INFO: Se refiere a mensajes informativos que resaltan el progreso de la aplicación en un nivel aproximado.
* DEBUG: Designa los eventos informativos detallados más útiles para depurar una aplicación.
* TRACE: Se utiliza para mostrar mensajes con un mayor nivel de detalle que debug.
* ALL: Es el nivel de máximo detalle, habilita todos los logs.

ACCESO A BASE DE DATOS

JDBC: Por las siglas de Java DataBase Conectivity, es un framework que consiste en múltiples interfaces y solo algunas clases de soporte. Esto se debe a que la idea detrás de JDBC es que cualquiera pueda crear su propia implementación del framework y adaptarla a sus necesidades. Dado que se trata de un conjunto de interfaces, cualquier código que interactúe con el framework no se verá afectado si se altera la implementación.  
  
Es por esto que JDBC define interfaces que **solo** declaran el comportamiento que debe llevarse a cabo para conectarse e interactuar con una base de datos. Así, nos encontramos con interfaces tales como: Connection (abstracción del comportamiento de una conexión), Statement (define el comportamiento para realizar sentencias contra una base de datos, sean queries y otras instrucciones), ResultSet (que abstrae el comportamiento para extraer resultados de las consultas), entre otras. Todas estas clases e interfaces están dentro del paquete java.sql.\*. Entonces, para interactuar con los diferentes motores de base de datos, debemos tener una implementación de estas y otras interfaces, es decir, una clase concreta que implemente cada interfaz.

H2: Es una base de datos open source escrita en Java que permite integrar aplicaciones en Java o ejecutarse en modo cliente-servidor. Principalmente, se puede configurar para que se ejecute como una base de datos en memoria. Entonces, los datos no persistirán en el disco, debido a que la base de datos no se utiliza para el desarrollo de producción, sino principalmente para el desarrollo y las pruebas.