REFACTORIZACIÓN

Refactorizar es la acción de modificar el código fuente de una aplicación sin alterar su comportamiento con el objetivo de mejorarlo. Esta acción no viene provocada por la aparición de bugs, sino por la posibilidad de mejorar su calidad mediante la modificación o sustitución del código en el que se han utilizado prácticas no recomendables.

```
// antes:
void imprimeFactura() {
   imprimeEncabezado();
   //imprime los detalles
   System.out.println ("Nombre:        " +  nombre );
   System.out.println ("Cantidad       " + getCantidad());
}
```

```
// después
void imprimeFactura() {
   imprimeEncabezado();
   imprimeDetalles(getCantidad());
}
void imprimeDetalles (double cantidad) {
   System.out.println ("Nombre:        " +  nombre );
   System.out.println ("Cantidad       " + cantidad );  // corregido
}
```

DOCUMENTACIÓN

Documentar el código de un programa es añadir suficiente información como para explicar lo que hace, punto por punto, de forma que no sólo los ordenadores sepan qué hacer, sino que además los humanos entiendan qué están haciendo y por qué. Porque entre lo que tiene que hacer un programa y cómo lo hace hay una distancia impresionante: todas las horas que el programador ha dedicado a pergeñar una solución y escribirla en el lenguaje que corresponda para que el ordenador la ejecute ciegamente.

Generar Javadoc

1.- Click derecho sobre el proyecto
2.- Click en Export
3.- Seleccionar Java
4.- Seleccionar Javadoc
5.-Seleccionar proyecto del que queremos generar JavaDoc
6.- En Destination elegir la ruta en la que queremos que se genere la documentación. Por dafault se hará en el mismo proyecto en una nueva carpeta llamada "doc"

Patrones de diseño

Los patrones de diseño son soluciones habituales a problemas que ocurren con frecuencia en el diseño de software. Son como planos prefabricados que se pueden personalizar para resolver un problema de diseño recurrente en tu código.

No se puede elegir un patrón y copiarlo en el programa como si se tratara de funciones o bibliotecas ya preparadas. El patrón no es una porción específica de código, sino un concepto general para resolver un problema particular. Puedes seguir los detalles del patrón e implementar una solución que encaje con las realidades de tu propio programa.

CLASIFICACIÓN:

Los patrones creacionales proporcionan mecanismos de creación de objetos que incrementan la flexibilidad y la reutilización de código existente.

Singleton, Builder, Prototype...

Los patrones estructurales explican cómo ensamblar objetos y clases en estructuras más grandes a la vez que se mantiene la flexibilidad y eficiencia de la estructura.

Adapter, Bridge, Decorator...

Los patrones de comportamiento se encargan de una comunicación efectiva y la asignación de responsabilidades entre objetos.

Observer, State, Visitor...

GIT

Git, que presenta una arquitectura distribuida, es un ejemplo de DVCS (sistema de control de versiones distribuido, por sus siglas en inglés). En lugar de tener un único espacio para todo el historial de versiones del software, como sucede de manera habitual en los sistemas de control de versiones antaño populares, como CVS o Subversion (también conocido como SVN), en Git, la copia de trabajo del código de cada desarrollador es también un repositorio que puede albergar el historial completo de todos los cambios.