Teoría UT4.-Optimización y Documentación

**Refactorización:**

1. Describe brevemente qué es la refactorización.

Pequeños cambios en el software para mejorar su legibilidad y eficacia sin modificar el comportamiento.

1. Describe dos **limitaciones** que se puede tener en el proceso de refactorización.

En bases de datos, pues cualquier modificación, por mínima que sea, supone un gran coste.

Otra limitación es con las interfaces, ya que aunque las modificaciones internas no afectan de manera directa a la interfaz, sí se ha de cambiar las referencias que se hacen a los métodos cambiados.

1. Indica 4 de los **patrones** más utilizados en la refactorización y explícalos brevemente.

* Renombrado: se cambia el nombre de una clase, paquete, método o campo por alguno más significativo.
* Mover la clase: así no se duplica código que ya se haya generado, necesitando actualizar todas las referencias a esa clase para llamarlas correctamente.
* Borrado seguro: cuando un elemento ya no es necesario, hay que borrar todas las referencias a él.
* Campos encapsulados: Se aconseja crear métodos getter y setter, (de asignación y de consulta) para cada campo que se defina en una clase. Cuando sea necesario acceder o modificar el valor de un campo, basta con invocar al método getter o setter según convenga.

1. ¿Qué es un **analizador de código**?

Es la herramienta que se encarga de obtener información que nos permita mejorar la base del código sin modificar su semántica.

1. Explica brevemente dos de los analizadores de código para Java (PMD y CPD)

PMD: Esta herramienta basa su funcionamiento en detectar patrones, que son posibles errores en tiempo de ejecución, código que no se puede ejecutar nunca porque no se puede llegar a él, código que puede ser optimizado, expresiones lógicas que pueden ser simplificadas, malos usos del lenguaje, etc.

CPD: Forma parte del PMD. Su función es encontrar código duplicado.

1. ¿Qué es el TDD y en qué consiste?

Desarrollo Guiado por Pruebas, que propone agilizar el ciclo de escritura de código y realización de pruebas de unidad. El programador realiza las pruebas de unidad en su propio código, e implementa esas pruebas antes de escribir el código a ser probado.

**Control de versiones:**

1. Define lo que es una versión.

Desde el punto de vista de la evolución, se define como la forma particular de un objeto en un instante o contexto dado.

1. Define los **componentes** que forman parte de un **sistema de control de versiones**:

A Repositorio: Es el lugar de almacenamiento de los datos de los proyectos. Suele ser un directorio en algún ordenador. 9 Módulo: En un directorio especifico del repositorio. Puede identificar una parte del proyecto o ser el proyecto por completo. 9 Revisión: Es cada una de las versiones parciales o cambios en los archivos o repositorio completo. La evolución del sistema se mide en revisiones. Cada cambio se considera incremental. 9 Etiqueta: Información textual que se añada a un conjunto de archivos o a un módulo completo para indicar alguna información importante. 9 Rama: Revisiones paralelas de un módulo para efectuar cambios sin tocar la evolución principal. Se suele emplear para pruebas o para mantener los cambios en versiones antiguas.

1. Define las **operaciones** que se pueden realizar en un **sistema de control de versiones:**
   1. Chekout

Comprueba la rama

* 1. Update

Actualiza la rama

* 1. Commit

Crea una transacción con ID referenciando a los cambios en los ficheros

* 1. Abort

Pausa un UPDATE

1. Describe las siguientes **herramientas** de **control de versiones**:
   1. CVS

**Concurrent Versions System**

* 1. SourceSafe

Es una herramienta que forma parte de Visual Studio

* 1. Darcs

Es un sistema de gestión de versiones distribuido. Algunas de sus características son:

La posibilidad de hacer commits locales(sinconexión),cada repositorio es una rama ensímisma, independencia de un servidor central, posibilidad de renombrar ficheros,variosmétodos de acceso como local, ssh, http y ftp etc..

* 1. Git

Esta herramienta decontrol deversiones, diseñada por Linus Torvalds

* 1. Mercurial

1. ¿Qué es GCS?

GestióndeConfiguracionesdeSoftware

1. Indicar si las siguientes herramientas CASE para GCS son libres o propietarias:
   1. Bugzilla LIBRE
   2. REM ([Enlace](http://www.lsi.us.es/descargas/descarga_programas.php?id=3))) GRATUITA

**Documentación:**

1. ¿Qué es JavaDoc?

Utilidad para generar documentación en formato HTML

1. Describir brevemente cómo comentar código y utilizar las siguientes etiquetas para que **JavaDoc** genere automáticamente documentación.
   1. **Comentarios**:

///

* 1. **Etiquetas (tags):**
     1. **@autor:** nombre\_autor
     2. **@versión:** versión
     3. **@see:**referencia

(#método();clase#método();

paquete.clase;paquete.clase#método()).

* + 1. **@since:** indicativo numerico
    2. **@param:** nombre\_parametro descripción
    3. **@return:** descripción
    4. **@thows:** nombre\_clase descripción
    5. **@deprecated:** descripción