CFGS

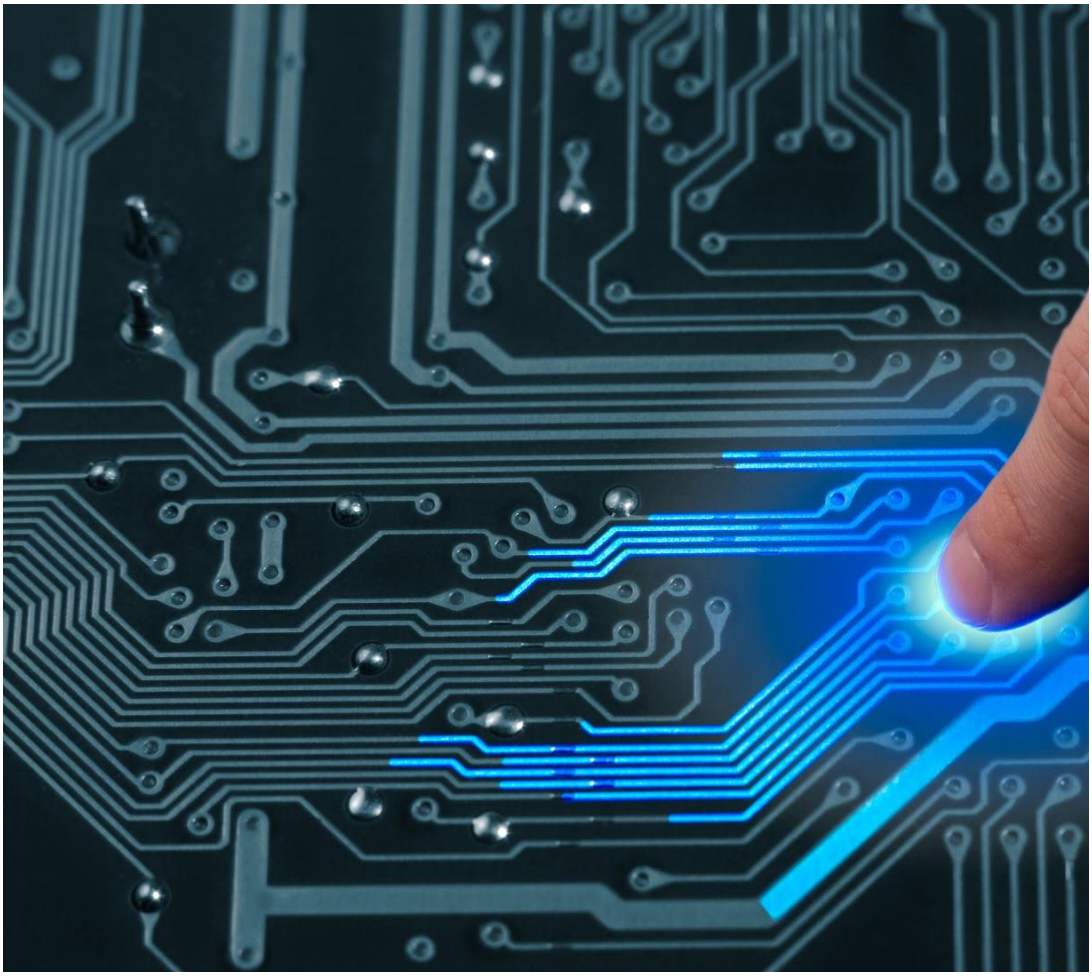
DESARROLLO DE APLICACIONES MULTIPLATAFORMA

DESARROLLO DE APLICACIONES WEB

ENTORNOS DE DESARROLLO

OPTIMIZACIÓN DE SOFTWARE

PAC 2: Optimización y documentación



PAC 2: Optimización y documentación

INFORMACIÓN
Para responder a las siguientes cuestiones deberás ayudarte del material didáctico y consultar internet.

Requisitos varios que deben cumplirse en vuestros trabajos:

• En los ejercicios, si se requieren de cálculos, estos deben aparecer en la respuesta que planteéis.
• Siempre que utilicéis información de Internet para responder / resolver alguna pregunta, tenéis que citar la fuente (la página web) de dónde habéis sacado aquella información.
• Siempre que utilicéis información del libro digital para responder / resolver alguna pregunta, tenéis que citar el tema y la página de dónde habéis sacado aquella información.
• No se aceptarán respuestas sacadas de Internet utilizando la metodología de copiar y pegar. Podéis utilizar Internet para localizar información, pero el redactado de las respuestas ha de ser vuestro.
• Las respuestas a las preguntas deben estar bien argumentadas, no se admiten respuestas escuetas o monosílabas.
• Se valorará la presentación, ortografía y gramática de vuestro trabajo hasta con un punto y medio de la nota final.

1. ¿Qué es la refactorización?

Es una técnica de la ingeniería de software que permite la optimización de un código previamente escrito.

- Limpia el código, mejorando la consistencia y la claridad.
- Mantiene el código, no corrige errores ni añade funciones nuevas.
- Va a permitir facilitar la realización de cambios en el código.
- Se obtiene un código limpio y altamente modularizado.

2. ¿Cuándo consideras que es necesario refactorizar?

- Cuando detectamos el mismo código en distintos lugares.
- Cuando los métodos son muy largos.
- Cuando una clase tiene muchos métodos, atributos e instancias.
- Cuando se pasan muchos parámetros.
- Cuando una clase es frecuentemente modificada.
- Cuando se deben realizar cambios para compatibilizar con otros cambios en otras clases.
- Cuando un método utiliza más elementos de otra clase que de la suya propia.
- Cuando tenemos clases que sólo tienen atributos y métodos de acceso a ellos.
- Cuando una subclase utiliza pocas características de sus superclases.

3. ¿Qué son los bad smells?

Son los síntomas para refactorizar:
- Código duplicado.
- Métodos muy largos.
- Clases muy grandes.
- Lista de parámetros extensa.
- Cambio divergente.
- Cirugía a tiro pistola.
- Clase de solo datos.
- Legado rechazado.

4. ¿Cuáles son las partes de un software más conflictivas a la hora de refactorizar?

Las bases de datos y las interfaces.

5. Nombra los métodos de refactorización que podemos encontrar en Eclipse.

- Rename
- Move
- Extract Constant
- Extract Local Variable
- Convert Local Variable to Field
- Extract Method
- Change Methos Signature
- Inline
- Member Type to Top Level
- Extract Interface
- Extract Superclass
- Convert Anonymous Class to Nested

6. ¿Qué es el control de versiones?

La capacidad de recordar los cambios realizados tanto en la estructura de directorios como en el contenido de los archivos.

7. Describe en qué consisten los siguientes términos del control de versiones:
   a. Repositorio: Lugar donde se almacenan todos los datos y los cambios.
   b. Revisión: Es una versión concreta de los datos almacenados.
   c. Etiquetar: Identificar puntos del proyecto fácilmente.
   d. Trunk: Línea principal del desarrollo del proyecto.
   e. Crear un Branch: son copias del proyecto, por lo que se crea una bifurcación entre el proyecto y su copia, y se puede modificar cualquiera de las dos versiones.
   f. Checkout: crear una copia de trabajo del proyecto, o de archivos y carpetas del repositorio en el equipo local.
   g. Commit: se confirman los cambios realizados en local para integrarlos al repositorio.
   h. Importación: subir carpetas y archivos del equipo local al repositorio.

i. Update: se integran los cambios realizados en el repositorio en la copia de trabajo local.
j. Merge: se unen los cambios de varias ramas a la vez.
k. Conflicto: Explicar claramente lo que hace el programa, de esta manera todo el equipo de desarrollo sabrá lo que se está haciendo y por qué.

8. ¿Qué es subversion? ¿Cómo funciona el ciclo de vida de esta herramienta?

Es una herramienta multiplataforma de código abierto para el control de versiones.

El ciclo de vida de subversion parte del comienzo del software, se crea el tronco del proyecto (trunk) y después se van añadiendo funcionalidades a este desarrollo mediante las ramas, para que no dañen el proyecto. Cada vez que se tienen listas (sin bugs) estas funcionalidades se fusionan con la rama principal. Por último, cada vez que hay una nueva versión del software, se añade una etiqueta para poder identificar estas partes fácilmente.

9. ¿Qué tipo de documentación se realizan de un proyecto?

- Documentación de las especificaciones.
- Documentación del diseño.
- Documentación del código fuente.
- Documentación de usuario final.

10. ¿Qué es JavaDoc? ¿Para qué se utiliza?

Es una utilidad de Java que se utiliza para extraer y generar documentación directamente del código en formato HTML.

¡Buen trabajo!