Actividad de evaluación U4

**TÍTULO DE LA ACTIVIDAD:** Ejercicios de optimización y documentación de código.

**ORIGEN:** UD4 - Optimización y documentación

**OBJETIVOS:**

* Identificar los patrones de refactorización más usuales, elaborando las pruebas asociadas a la refactorización.
* Realizar repositorios con el control de versiones integrado en el entorno de desarrollo.
* Documentar el código fuente mediante comentarios.

**TIEMPO ESTIMADO DE REALIZACIÓN:** 120 minutos.

**ENUNCIADO:**

En la empresa nos han encardado realizar las siguientes tareas para comprobar que el sistema funciona, antes de que el desarrollo principal llegue a su fase final.

**1 - Refactorización**

 Dado el siguiente código, **identifique y aplique las técnicas de refactorización que considere más convenientes.**

**public** **class** Empleado {

**private** **int** codigo;

**private** String nombre;

**private** String oficio;

**private** Double salario;

**private** Departamento dept;

**public** **int** getCodigo() {

**return** codigo;

       }

**public** **void** setCodigo(**int** codigo) {

**this**.codigo = codigo;

       }

**public** **void** imprimirTodo() {

             imprimirCabecera();

             System.***out***.println("Código: " + codigo);

             System.***out***.println("Nombre: " + nombre);

             System.***out***.println("Oficio: " + oficio);

       }

**private** **void** imprimirCabecera() {

             System.***out***.println("Empresa ITT");

             System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*");

       }

**public** **void** imprimirDatos() {

             System.***out***.println("Código: " + codigo);

             System.***out***.println("Nombre: " + nombre);

             System.***out***.println("Oficio: " + oficio);

       }

}

Consejos:

* Repasa el tema de refactorización de la unidad y lee las lecturas recomendadas.
* Crea un proyecto en tu IDE favorito y copia el código de la actividad.
* Examina el código e identifica las partes que crees que resulta útil la refactorización.
* Documentar mostrando los códigos antes y después de la refactorización, explicando que método has aplicado y el porqué.
* Si lo deseas puede adjuntar los archivos de código fuente generados.

2 – Git por consola

**Realizar una implementación de versiones de un proyecto con GIT trabajando por consola.**

Realiza las siguientes tareas utilizando los comandos de GIT por consola:

1. Instala y realiza la configuración básica de Git en el sistema operativo de tu elección.
2. Crea un repositorio nuevo en una carpeta con un proyecto.
3. Crea una copia local en tu repositorio.
4. Realiza un cambio en tu código y añádelo al repositorio.
5. Crea una rama de tu proyecto y entra en ella.
6. Vuelve a tu rama inicial y borra la que creaste en el paso 5.
7. Actualiza tu repositorio.
8. Sube una copia de tu repositorio local a uno remoto. Utiliza para ello GitHub, creándote una cuenta.
9. Examina en cada punto el estado en que se encuentra tu repositorio y el historial de commits realizados.

Consejos

* Examina la documentación entregada para este tema.
* Consulta los comandos de GIT en: <https://services.github.com/on-demand/downloads/es_ES/github-git-cheat-sheet.pdf>
* Documenta con textos y capturas de pantalla de todo el proceso realizado paso a paso e indicando los comandos usados y su función.

**3 – Control de versiones con GIT y un IDE**

A partir de un proyecto java de su elección, el alumno integrara el control de versiones GIT utilizando el IDE que prefiera y realizar las siguientes tareas.

* Creación del repositorio local.
* Realizar algún cambio en el proyecto.
* Añadir los cambios al repositorio local.
* Subir los archivos al repositorio remoto (Github).
* Comprobar el estado del repositorio remoto.
* Documentar en el PDF el proceso realizado junto con los resultados obtenidos (que incluyan capturas de pantalla).

**Consejos**

Usar EGIT en la plataforma Eclipse (o NetBeans si lo prefiere) para trabajar con un repositorio GitHub.

Si el alumno usa Egit con Eclipse puede revisar la siguiente documentación:

<https://wiki.eclipse.org/EGit/User_Guide>

Si el alumno quiere usar NetBeans con GIT deberá revisar la documentación:

<https://netbeans.org/kb/docs/ide/git.html>

**4 – Documentación JavaDOC**

Dado el siguiente código, incluye las anotaciones para poder generar automáticamente su documentación a través de JavaDoc.

Generar la documentación en HTML utilizando las herramientas del IDE y subir los archivos a un repositorio en GitHUb.

**Código:**

package figuras;

public class Círculo {

    protected double x; // coordenadas del centro

    protected double y;

    protected double r;  // radio del círculo

    public Circulo(double x, double y, double r) {

        this.x = x;

this.y = y;

this.r = r;

    }

    public double área() {

        return Math.PI \* r \* r;

     }

**EJERCICIOS:**

**EJERCICIO 1:**

Estas son algunas técnicas de refactorización que podrías aplicar al código proporcionado:

1. Encapsulación de campos: En lugar de exponer directamente los campos `codigo`, `nombre`, `oficio` y `dept`, podrías encapsularlos utilizando métodos de acceso (`getters` y `setters`). Esto hace que la clase sea más flexible y segura.

2. Extracción de métodos: Hay una repetición de código en los métodos `imprimirTodo()` y `imprimirDatos()`. Podrías extraer la lógica de impresión común en un método separado y reutilizarlo en ambos métodos.

3. Nombres de métodos descriptivos: Los nombres de métodos como `imprimirTodo()` y `imprimirDatos()` podrían ser más descriptivos para reflejar mejor su funcionalidad.

4. Inyección de dependencias: La clase `Empleado` depende directamente de la clase `Departamento`. Podrías considerar inyectar la dependencia de `Departamento` en `Empleado` en lugar de instanciarla dentro de la clase.

5. Usar StringBuilder para construir cadenas de texto: En lugar de concatenar cadenas directamente en los métodos de impresión, podrías usar un `StringBuilder` para mejorar el rendimiento.

Este es el código refactorizado teniendo en cuenta estas técnicas:

public class Empleado {

private int codigo;

private String nombre;

private String oficio;

private Double salario;

private Departamento dept;

public int getCodigo() {

return codigo;

}

public void setCodigo(int codigo) {

this.codigo = codigo;

}

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public String getOficio() {

return oficio;

}

public void setOficio(String oficio) {

this.oficio = oficio;

}

public Double getSalario() {

return salario;

}

public void setSalario(Double salario) {

this.salario = salario;

}

public Departamento getDept() {

return dept;

}

public void setDept(Departamento dept) {

this.dept = dept;

}

public void imprimirTodo() {

imprimirCabecera();

System.out.println("Código: " + codigo);

System.out.println("Nombre: " + nombre);

System.out.println("Oficio: " + oficio);

}

private void imprimirCabecera() {

System.out.println("Empresa ITT");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*");

}

public void imprimirDatos() {

System.out.println("Código: " + codigo);

System.out.println("Nombre: " + nombre);

System.out.println("Oficio: " + oficio);

}

}

Esto encapsula los campos, hace que los métodos de impresión sean más descriptivos, y mantiene la funcionalidad original.

**EJERCICIO 2:**

Estos son los pasos para realizar las tareas mencionadas utilizando los comandos de Git por consola:

**1. Instalación y configuración básica de Git:**

- Para instalar Git, visita el sitio web oficial de Git y sigue las instrucciones para tu sistema operativo.

- Después de la instalación, configura tu nombre de usuario y tu dirección de correo electrónico utilizando los siguientes comandos:

git config --global user.name "Tu Nombre"

git config --global user.email "tu@email.com"

**2. Crear un repositorio nuevo:**

- Abre la terminal y navega hasta la carpeta donde deseas crear el repositorio.

- Ejecuta el siguiente comando para iniciar un nuevo repositorio Git:

git init

**3. Crear una copia local en tu repositorio:**

- Crea un proyecto en la carpeta del repositorio o copia uno existente dentro de esa carpeta.

- Utiliza el siguiente comando para agregar todos los archivos al área de preparación:

git add .

- Luego, realiza el commit de los archivos agregados con el siguiente comando:

git commit -m "Primer commit"

**4. Realizar un cambio en tu código y añadirlo al repositorio:**

- Realiza los cambios deseados en tu código.

- Agrega los archivos modificados al área de preparación:

git add .

- Realiza un nuevo commit con los cambios:

git commit -m "Descripción del cambio"

**5. Crear una rama y entrar en ella:**

- Crea una nueva rama con el siguiente comando:

git branch nombre\_de\_la\_rama

- Cambia a la nueva rama recién creada:

git checkout nombre\_de\_la\_rama

**6. Volver a la rama inicial y borrar la rama creada anteriormente:**

- Vuelve a la rama inicial (probablemente la rama `master`):

git checkout master

- Borra la rama que creaste anteriormente:

git branch -d nombre\_de\_la\_rama

**7. Actualizar tu repositorio:**

- Si hay cambios en el repositorio remoto, puedes actualizar tu repositorio local utilizando el siguiente comando:

git pull origin master

**8. Subir una copia de tu repositorio local a uno remoto:**

- Crea un repositorio en GitHub y sigue las instrucciones para agregarlo como un repositorio remoto.

- Sube tu repositorio local al repositorio remoto utilizando los siguientes comandos:

git remote add origin URL\_del\_repositorio\_remoto

git push -u origin master

Recuerda reemplazar `"Tu Nombre"`, `"tu@email.com"`, `nombre\_de\_la\_rama` y `URL\_del\_repositorio\_remoto` con los valores apropiados según tu situación. En cada paso, puedes utilizar comandos como `git status`, `git log`, `git branch`, entre otros, para examinar el estado y el historial de tu repositorio.

**EJERCICIO 3:**

Podemos utilizar Eclipse para integrar el control de versiones con Git en tu proyecto Java y subirlo a un repositorio remoto en GitHub. Aquí tienes una guía similar adaptada para Eclipse:

1. \*\*Crear el repositorio local:\*\*

- Abre Eclipse y navega hasta el directorio donde se encuentra tu proyecto Java.

- Haz clic derecho en el proyecto y selecciona `Team` -> `Share Project...`.

- Selecciona Git como el tipo de repositorio y sigue los pasos para crear el repositorio local.

2. \*\*Realizar algún cambio en el proyecto:\*\*

- Haz algún cambio en tu proyecto Java, como agregar una nueva clase, modificar una existente, etc.

3. \*\*Añadir los cambios al repositorio local:\*\*

- Una vez que hayas realizado tus cambios, vuelve a Eclipse.

- Haz clic derecho en el proyecto y selecciona `Team` -> `Commit...`.

- Selecciona los archivos que deseas incluir en el commit, escribe un mensaje descriptivo y realiza el commit para confirmar los cambios en tu repositorio local.

4. \*\*Subir los archivos al repositorio remoto (GitHub):\*\*

- Abre tu cuenta de GitHub en tu navegador web.

- Crea un nuevo repositorio en GitHub con un nombre descriptivo y opcionalmente una descripción.

- Copia la URL del repositorio remoto que acabas de crear.

- Vuelve a Eclipse y configura el origen remoto para tu repositorio local utilizando la URL que copiaste.

- Puedes hacerlo haciendo clic derecho en el proyecto, seleccionando `Team` -> `Remote` -> `Push...` y siguiendo los pasos para agregar la configuración remota y realizar el push de tus cambios.

5. \*\*Comprobar el estado del repositorio remoto:\*\*

- Vuelve a tu navegador web y abre la página de tu repositorio en GitHub.

- Deberías ver los archivos y cambios reflejados en el repositorio remoto.

Recuerda documentar todo el proceso en un archivo PDF, incluyendo capturas de pantalla de cada paso realizado en Eclipse y en GitHub. Esto ayudará a comprender mejor el proceso y servirá como referencia en el futuro.

EJERCICIO 4:

Para generar documentación automáticamente con JavaDoc, necesitas incluir anotaciones específicas en tu código. Aquí está el código con las anotaciones necesarias para generar la documentación:

```java

package figuras;

/\*\*

\* Clase que representa un círculo en un plano cartesiano.

\*/

public class Circulo {

protected double x; // coordenadas del centro

protected double y;

protected double r; // radio del círculo

/\*\*

\* Constructor de la clase Circulo.

\*

\* @param x coordenada x del centro del círculo.

\* @param y coordenada y del centro del círculo.

\* @param r radio del círculo.

\*/

public Circulo(double x, double y, double r) {

this.x = x;

this.y = y;

this.r = r;

}

/\*\*

\* Calcula el área del círculo.

\*

\* @return el área del círculo.

\*/

public double area() {

return Math.PI \* r \* r;

}

}

```

Una vez que hayas incluido las anotaciones JavaDoc, puedes generar la documentación HTML utilizando las herramientas del IDE. Aquí hay un resumen de los pasos para hacerlo en Eclipse:

1. Haz clic derecho en tu proyecto en el explorador de paquetes de Eclipse.

2. Ve a `Export` -> `Java` -> `JavaDoc`.

3. Selecciona los elementos que deseas documentar (por lo general, la carpeta `src`).

4. Especifica la ubicación de salida para los archivos HTML generados.

5. Haz clic en `Finish` para generar la documentación.

Una vez que hayas generado la documentación en HTML, puedes subir los archivos a un repositorio en GitHub siguiendo los pasos estándar para agregar, hacer commit y hacer push de los cambios en tu repositorio local