ACTIVIDAD 3.3: REFACTORIZACIÓN

INSTRUCCIONES:

A). Edita un documento con las capturas de pantalla(si fuese necesario) de cada ejercicio y respuestas, donde sea necesario.

B) Una vez terminado, convertir el archivo a pdf y subir a Classroom en el apartado de la actividad con: **apellidonombre1_apellidonombre2_Actividad_3.3**

C) Elabora un documento .pdf explicando el proceso seguido para realizar las refactorizaciones pedidas o algunas otras que os parezcan convenientes. Se deben poner capturas de pantalla previas y una vez realizadas las modificaciones.
Se Pide enviar:

- ✔ Documento explicativo del proceso con capturas de pantalla.

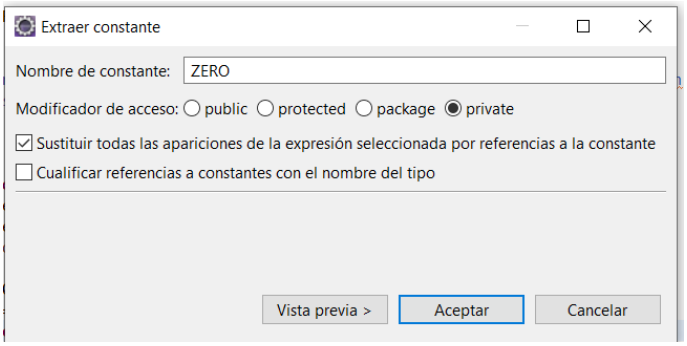
DE ALGUNAS DE LOS APARTADOS RECORDAD QUE TENEIS MIS VIDEOS DE LOOM.

En esta práctica se trata de probar los Patrones de refactorización más habituales:

1.- Renombrado (rename): Este patrón nos indica que debemos cambiar el nombre de un paquete, clase, método o campo, por un nombre más significativo.
Realizar las siguientes modificaciones:

- Cambiar el atributo tipoInterés para eliminar la tilde.
- Cambiar el nombre del método obtenerNombre() por getNombre.
- Cambiar el nombre del método obtenerCuenta por getCuenta.
- Cambiar el nombre del método getTipoInterés por getTipoInteres
- Cambiar el nombre del método setTipoInterés por setTipoInteres
- Cambiar el nombre del método estado por getSaldo.
- Cambiar el nombre de la clase CCuenta por CuentaCorriente. Observad como se cambia el nombre de la clase en todos los ficheros del proyecto.

2.- **Introducir CONSTANTE:** nos permite modificar el valor asignado a una variable de un método por una constante.
Botón derecho en el valor Refactor Extract constant



Por ejemplo, el 0 para la cantidad en CuentaCorriente y todos los valores 0 sustituirlos por la constante ZERO.

3.- **Sustituir bloques de código por un método**: Este patrón nos aconseja sustituir un bloque de código, por un método. De esta forma, cada vez que queramos acceder a ese bloque de código, bastaría con invocar al método.
Por ejemplo, añadir en la clase Main antes de terminar el programa crear un segundo objeto de la clase SCuenta, y las siguientes líneas para visualizar los datos de los 2 objetos creados:

```
System.out.println("Datos clientes");
System.out.println(cuenta1.toString());
System.out.println(cuenta2.toString());
```

Y luego sustituir dicho código por un método llamado visualizarDatosCuenta.

4.- **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.
Para ello se pulsa botón derecho sobre la clase source Generate Setters and Getters.
Por ejemplo, añadir todos los get y set de los atributos de la clase que aún no los tengan.

5.- **También se podría Mover la clase de un paquete a otro, o de un proyecto a otro. La idea es no duplicar código que ya se haya generado. Esto impone la actualización en todo el código fuente de las referencias a la clase en su nueva localización.**

6.- **Change Method signature. Este método permite cambiar la firma de un método. Es decir, el nombre del método y los parámetros que tiene. De forma automática se actualizarán todas las dependencias y llamadas al método dentro del proyecto.**

Por ejemplo, añadir en el método visualizarDatosCuenta un nuevo parámetro que sea un mensaje de bienvenida, por defecto tenga valor “Buenos días”
Observad cómo se modifican los comentarios JavaDoc.

7.- Extract Interface. Este método permite escoger los métodos de una clase para crear una interface (plantilla que define los métodos acerca de lo que puede hacer o no una clase, pero no los desarrolla. Las clases que implementan la Interface son las que desarrollan los métodos).

Eclipse permite visualizar el histórico de refactorizaciones realizadas sobre un proyecto. Para ello se abre el menú refactor=> History
También permite crear un script con todos los cambios realizados y guardarlo en un fichero .XML (menú refactor=> create script).

8.- Extract Superclass. Permite extraer una superclase
Eclipse permite visualizar el histórico de refactorizaciones realizadas sobre un proyecto. Para ello se abre el menú refactor=> History
También permite crear un script con todos los cambios realizados y guardarlo en un fichero .XML (menú refactor=> create script).

8.- Visualizar el histórico de refactorizaciones realizadas sobre el proyecto. Para ello se abre el menú refactor History