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

**2 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.

**3 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.

**4 Mover la clase**: Si es necesario, se puede mover una 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.

**5 Borrado seguro:** Se debe comprobar, que cuándo un elemento del código ya no es necesario, se han borrado todas las referencias a él que había en cualquier parte del proyecto.

**6 Cambiar los parámetros del proyecto:** Nos permite añadir nuevos parámetros a un método y cambiar los modificadores de acceso.

**7 Extraer la interfaz**: Crea un nueva interfaz de los métodos public non‐static seleccionados en una clase o interfaz.

**8 Mover del interior a otro nivel**: Consiste en mover una clase interna a un nivel superior en la jerarquía.

En los que nos centramos nosotros van a ser los siguientes:

**1 Rename:** Este patrón nos indica que debemos cambiar el nombre de un paquete, clase, método o campo, por un nombre más significativo.

Este caso se lleva a cabo de la siguiente manera, clicamos sobre el archivo a renombrar> Refactor>Rename> y a veces por defecto te da un nombre diferente, si no nos gusta, podemos cambiarlo.

**2 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.

Este patrón sustituye un trozo de código por un método. Para ello seleccionamos las líneas de código que queremos sustituir > Refactor > Method > Escribimos el nombre del método y podemos elegir el acceso, que por defecto viene private, pero también podemos elegir public, protected y default. Y genera el método. Sirve para ahorrar código.

**4 Mover la clase**: Si es necesario, se puede mover una 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.