Refactorización

Refactorizar código fuente es modificar el código sin que se modifque el comportamiento del programa. Entonces, ¿para qué sirven estas modificaciones? Para mejorar su estructura, su legibilidad o su eficiencia.

El nombre viene del mundo matemático, de la refactorización de polinomios. Así, resulta que $(x + 1)(x - 1)$ se puede expresar como $x^2 - 1$ sin que se altere su sentido.

Ejemplos de refactorización:

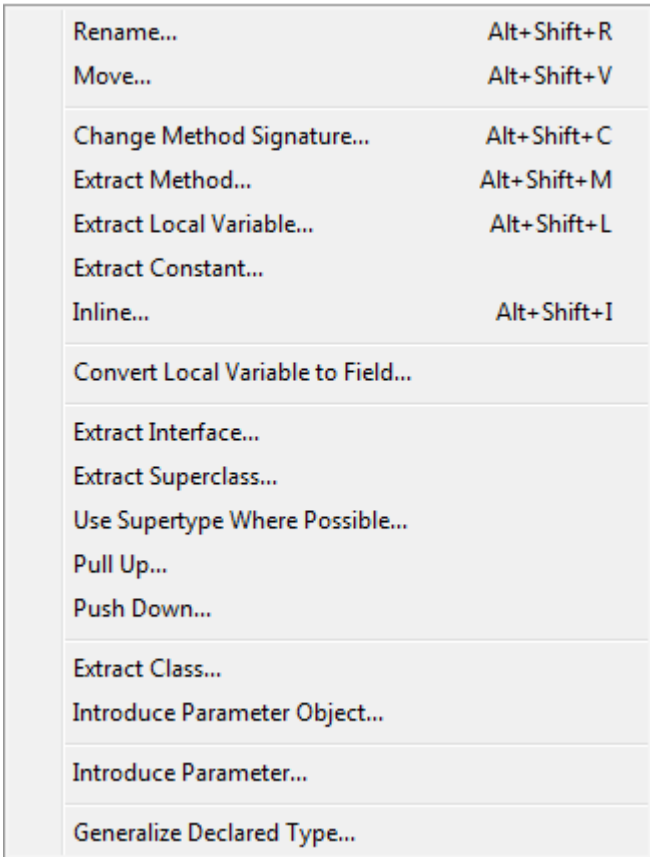
- Cambiar de nombre variables, clases o métodos con el fin de clarificar código
- Hacer un código más extenso si se gana en claridad
- Hacer un código menos extenso sólo si con eso se gana eficiencia
- Reorganización de código condicional complejo (varios if o condiciones anidadas o complejas)
- Dividir métodos o clases demasiado largos
- Creación de código común (en una clase o método) para evitar el código repetido

Algunas pistas que nos pueden indicar la necesidad de refactorizar son:

- Código duplicado
- Métodos demasiado largos
- Clases muy grandes o con demasiados métodos
- Métodos más interesados en los datos de otra clase que en los de la propia
- Grupos de datos que suelen aparecer juntos y parecen más una clase que datos sueltos
- Clases con pocas llamadas o que se usan muy poco
- Exceso de comentarios explicando el código

Refactorización en Eclipse

Hacer clic con el botón secundario sobre un fragmento de código, y escoger Refactor. Aparece un menú con muchas opciones, de las que estudiaremos algunas.



Rename

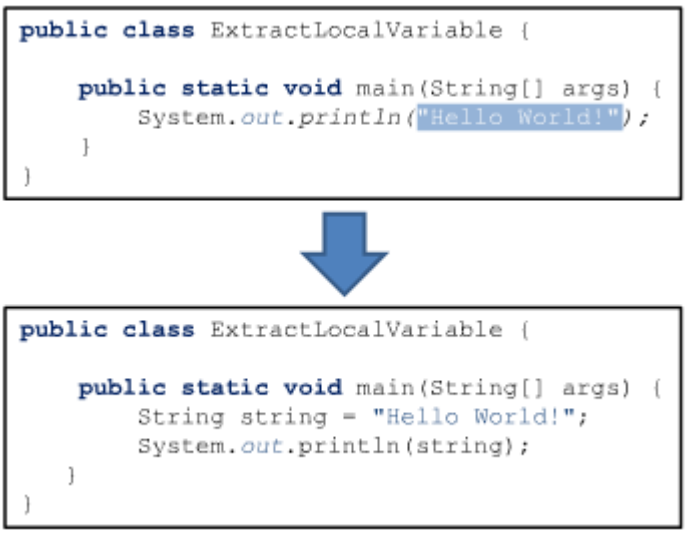
La opción más común: cambia el nombre a cualquier elemento (variable, atributo, método, clase…) y hace los cambios necesarios en las referencias que haya a dicho elemento en todo el proyecto.

Move

Cambia una clase de un paquete a otro, afectando a su declaración “package” y a su localización en el disco.

Extract local variable

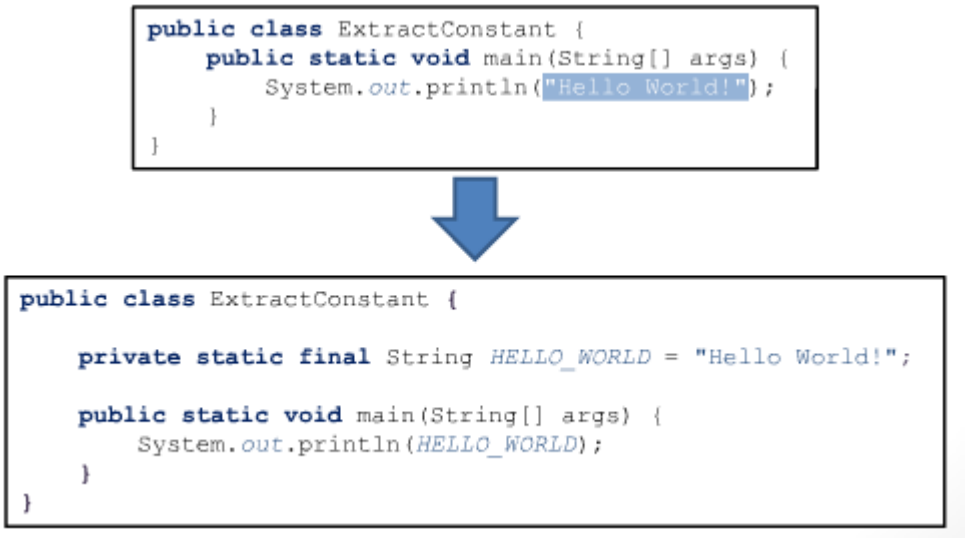
Cualquier expresión (número, String…) se asigna a una variable local. Las referencias a esa expresión se modifican por una referencia a la variable.



Sólo afecta al ámbito actual, es decir, si la expresión existe por ejemplo en otro método, no se hará ningún cambio en ese otro método.

Extract constant

Exactamente igual que el anterior, pero genera una constante con la expresión seleccionada.



Convert local variable to field

A veces definimos una variable local dentro de un método pero luego nos damos cuenta de que esa variable es más relevante de lo que parecía y debe ser accedida desde otros métodos: por tanto debe ser considerada un atributo de la clase. Esta opción realiza el trabajo por nosotros.

```java
public class ConvertLocalVariable {

    public static void main(String[] args) {
        String msg = "Hello World!";
        System.out.println(msg);
    }
}
```

```java
public class ConvertLocalVariable {

    private static String msg;

    public static void main(String[] args) {
        msg = "Hello World!";
        System.out.println(msg);
    }
}
```

Extract method

Convierte el código seleccionado en un método: haremos esto cuando veamos que nos interesa reutilizar ese código. También puede servir para descargar un método que es demasiado largo.

Eclipse sólo nos preguntará el nombre del método, pero descubrirá automáticamente los parámetros y tipo de retorno necesarios.

Observa las diferencias:

```java
int base = 34;
int altura = 54;
double area = base * altura / 2.0;
System.out.println(area);
```

```java
double area = calcularArea();
System.out.println(area);
```

```java
private double calcularArea() {
    int base = 34;
    int altura = 54;
    double area = base * altura / 2.0;
    return area;
}
```

```java
int base = 34;
int altura = 54;
double area = base * altura / 2.0;
System.out.println(area);
```

```java
int base = 34;
int altura = 54;
double area = calcularArea(base, altura);
System.out.println(area);
```

```java
private double calcularArea(int base, int altura) {
    double area = base * altura / 2.0;
    return area;
}
```

Inline

Hace lo contrario que los “Extract”: elimina una declaración de variable, método o constante y coloca su valor (en el caso de variables y constantes) o su código (en el caso de métodos) en aquellos lugares en que se referenciaba a esa variable, método o constante que ya no existirán.

Es muy útil cuando se comprueba que el contenido de una variable o constante sólo se va a usar una sola vez y por tanto no merece la pena almacenarlo, sino que queda más limpio el código en una línea.

También cuando vemos que un método sólo se llama una o dos veces y no merece la pena por tanto aislar ese código en un método.

En los ejemplos de Extract anteriores, Inline haría el camino inverso (desde abajo hacia arriba).

Change method signature

Modifica la “firma” o cabecera del método. Si el método ha sido ya usado, cambiar el número o tipo de parámetros (así como el tipo de valor devuelto) provocará fallos de compilación.

Es útil para cambiar el nombre de los parámetros, o su orden (Eclipse modificará también el orden de entrada de los parámetros en todas las llamadas al método)