

Refactorización

TEMA 6

Introducción

- ▶ Consiste en aplicar transformaciones en el código del programa para mejorar su estructura sin que cambie el comportamiento
- ▶ Se aplica sobre software que funciona correctamente y su objetivo es hacer más entendible el código.
- ▶ Algunas de las técnicas de refactorización que se utilizan son:
 - ▶ Extraer métodos
 - ▶ Renombrar
 - ▶ Mover método
 - ▶ Encapsulado de campos

Extraer métodos

- Sustituye un código de bloque por un método. Veamos un ejemplo

Partimos de un bloque de código

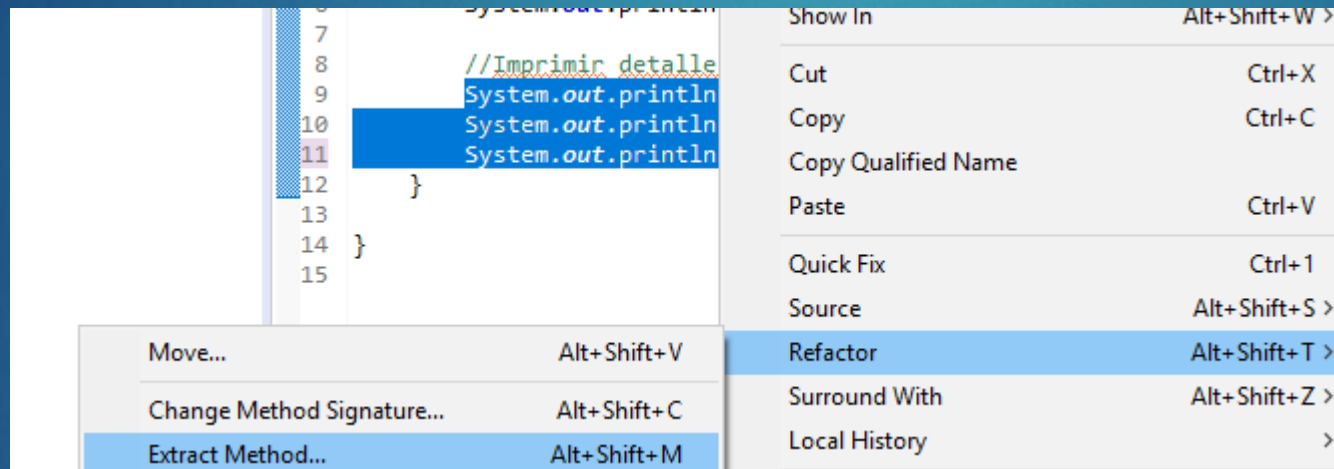
```
public class ExtraerMetodo {  
  
    public void Imprimir() {  
        //Imprimir cabecera  
        System.out.println("---CABECERA---");  
  
        //Imprimir detalles  
        System.out.println("Nombre");  
        System.out.println("Apellidos");  
        System.out.println("Dirección");  
    }  
}
```

Deseamos crear métodos con algunas líneas de ese código. Veamos como quedaría

```
public class ExtraerMetodo {  
  
    public void Imprimir() {  
        //Imprimir cabecera  
        imprimirCabecera();  
  
        //Imprimir detalles  
        imprimirDetalles();  
    }  
  
    private void imprimirDetalles() {  
        System.out.println("Nombre");  
        System.out.println("Apellidos");  
        System.out.println("Dirección");  
    }  
  
    private void imprimirCabecera() {  
        System.out.println("---CABECERA---");  
    }  
}
```

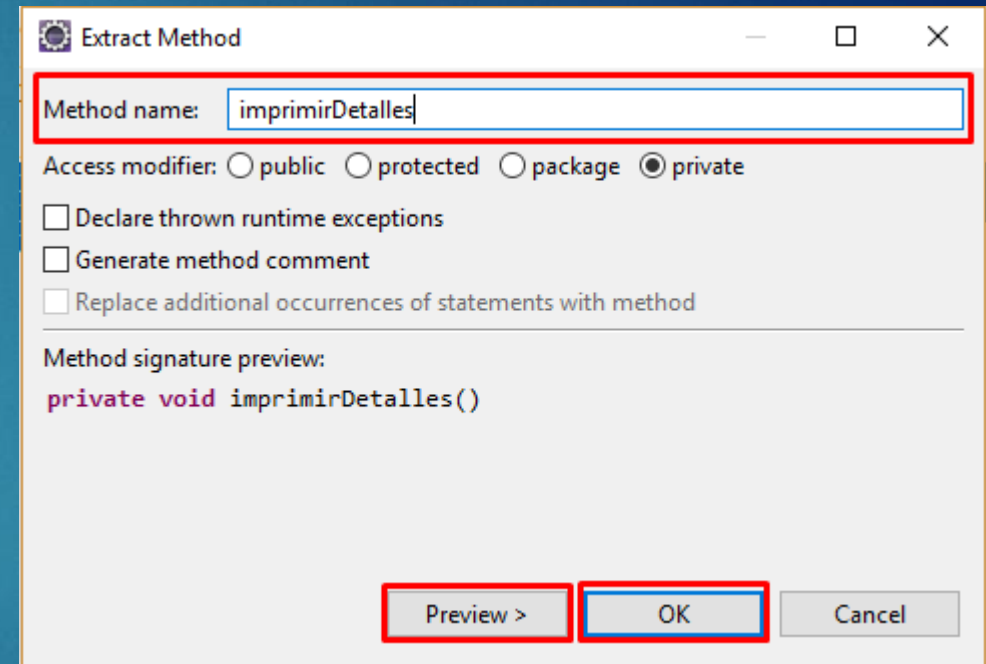
Extraer métodos

- ▶ Eclipse nos facilita la refactorización para extraer métodos.
- ▶ Primero seleccionaremos las líneas de código que deseamos extraer y hacemos botón derecho y buscamos Refactorizar->Extraer método



Extraer métodos

- ▶ En el cuadro de diálogo que nos aparece debemos:
 - ▶ Establecer el nombre del método
 - ▶ La visibilidad (público, protegido, privado)
 - ▶ Ok: Procederá a realizar los cambios
 - ▶ Preview: Permitirá ver los cambios antes de su aplicación



Renombrar

- ▶ Nos ayuda a cambiar el nombre de clases, métodos, parámetros, variables y hacer más entendible el código. Veamos un ejemplo:
- ▶ Partimos del siguiente código
- ▶ Es más entendible si se utilizan nombres más descriptivos

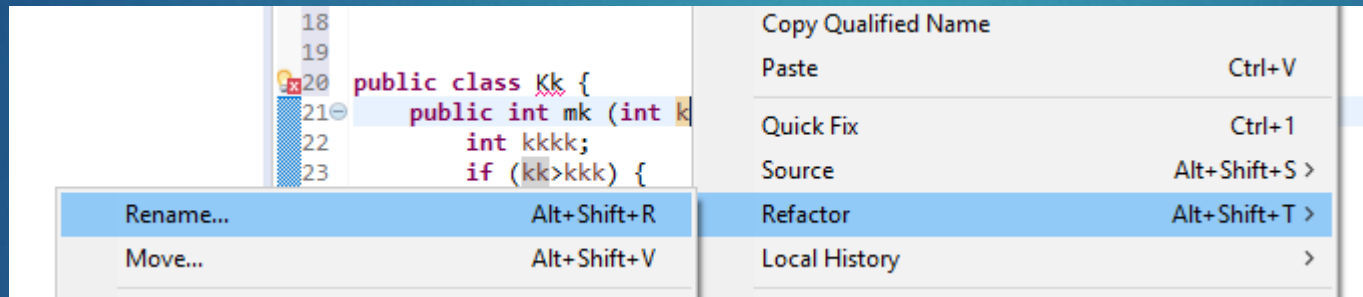
```
public class Kk {  
    public int mk (int kk,int kkk) {  
        int kkkk;  
        if (kk>kkk) {  
            kkkk=kk;  
        }  
        else {  
            kkkk=kkk;  
        }  
        return kkkk;  
    }  
}
```



```
public class Renombrar {  
    public int mayor (int valor1,int valor2) {  
        int resultado;  
        if (valor1>valor2) {  
            resultado=valor1;  
        }  
        else {  
            resultado=valor2;  
        }  
        return resultado;  
    }  
}
```

Renombrar

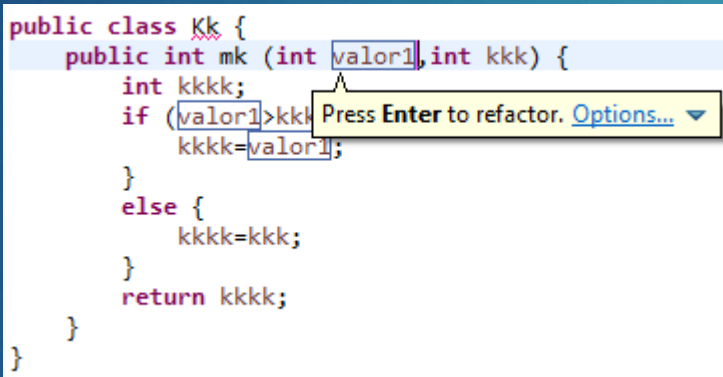
- ▶ Eclipse nos va a facilitar el renombrado.
- ▶ Para ello nos situaremos encima de la clase, método,... que deseemos renombrar y haremos botón derecho Refactorizar->Renombrar



Renombrar

- Según cambiemos el nombre se cambiará en todos los uso que se haga de esa clase, método,... dentro del código. Para finalizar pulsaremos Enter

```
public class Kk {  
    public int mk (int valor1, int kkk) {  
        int kkkk;  
        if (valor1 > kkk) {  
            kkkk = valor1;  
        }  
        else {  
            kkkk = kkk;  
        }  
        return kkkk;  
    }  
}
```




Mover método

- ▶ Nos permite mover métodos de unas clases a otras donde por su encapsulación están mejor ubicadas.
- ▶ Partimos de dos clases con sus respectivos métodos y atributos
- ▶ Se considera que el método aplicarDto debería formar parte de la clase Pedido

```
public class Pedido {  
  
    public double ajusteTotal = 0;  
  
    public double getTotal() {  
        return 100;  
    }  
  
}
```

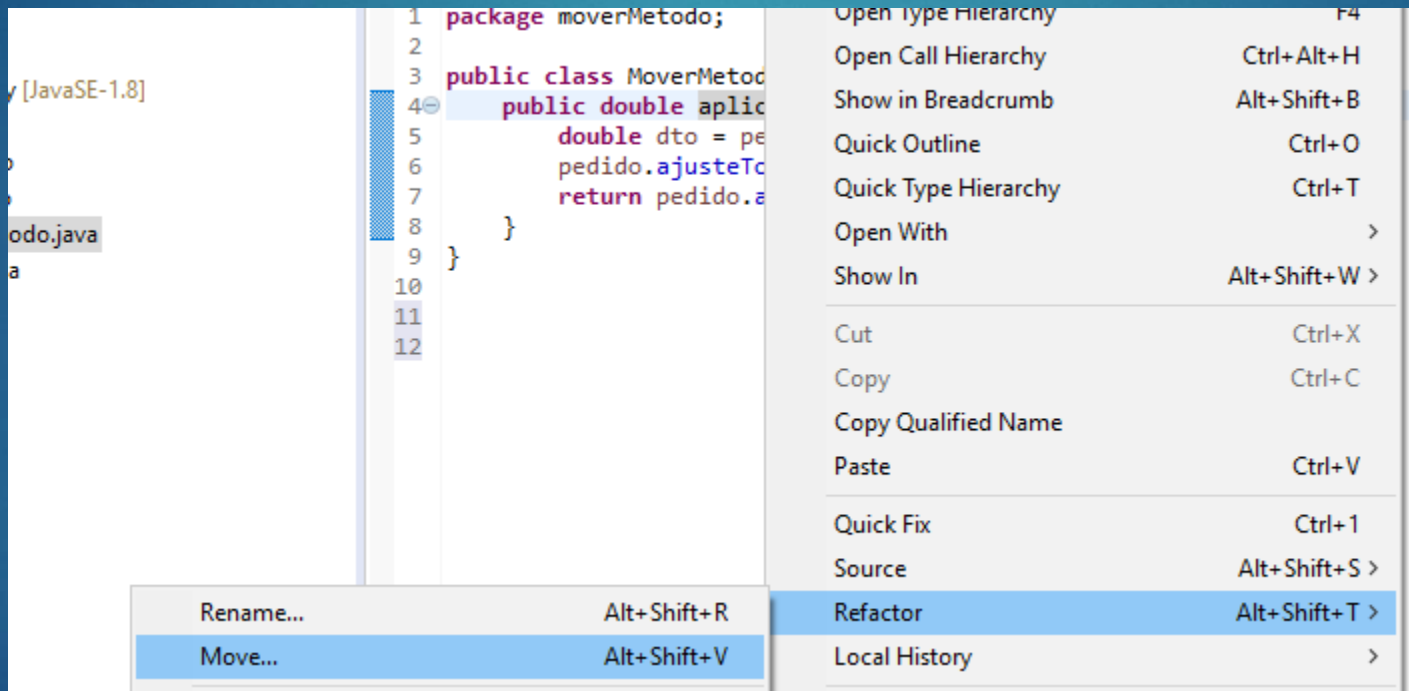
```
public class MoverMetodo {  
    public double aplicarDto(Pedido pedido, double porcentaje) {  
        double dto = pedido.getTotal() * porcentaje;  
        pedido.ajusteTotal = pedido.getTotal() - dto;  
        return pedido.ajusteTotal;  
    }  
}
```

```
public class Pedido {  
  
    public double ajusteTotal = 0;  
  
    public double getTotal() {  
        return 100;  
    }  
  
    public double aplicarDto(MoverMetodo moverMetodo, double porcentaje) {  
        double dto = getTotal() * porcentaje;  
        ajusteTotal = getTotal() - dto;  
        return ajusteTotal;  
    }  
  
}
```



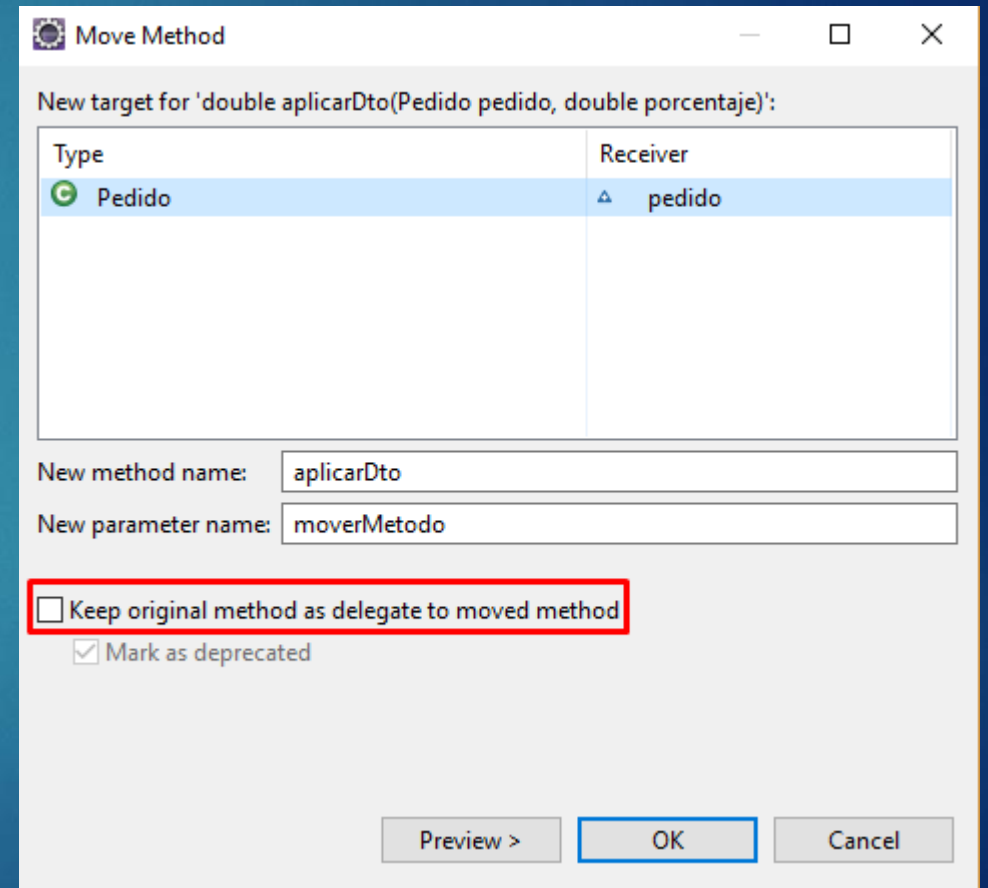
Mover método

- Para mover un método en Eclipse nos situaremos en el nombre del método y haremos botón derecho Refactorizar->Mover...



Mover método

- En el cuadro de diálogo que nos aparece seleccionaremos el nombre del método, la clase dónde moveremos y le indicaremos si deseamos mantener el método en la clase original como delegado.



Encapsulado de campos

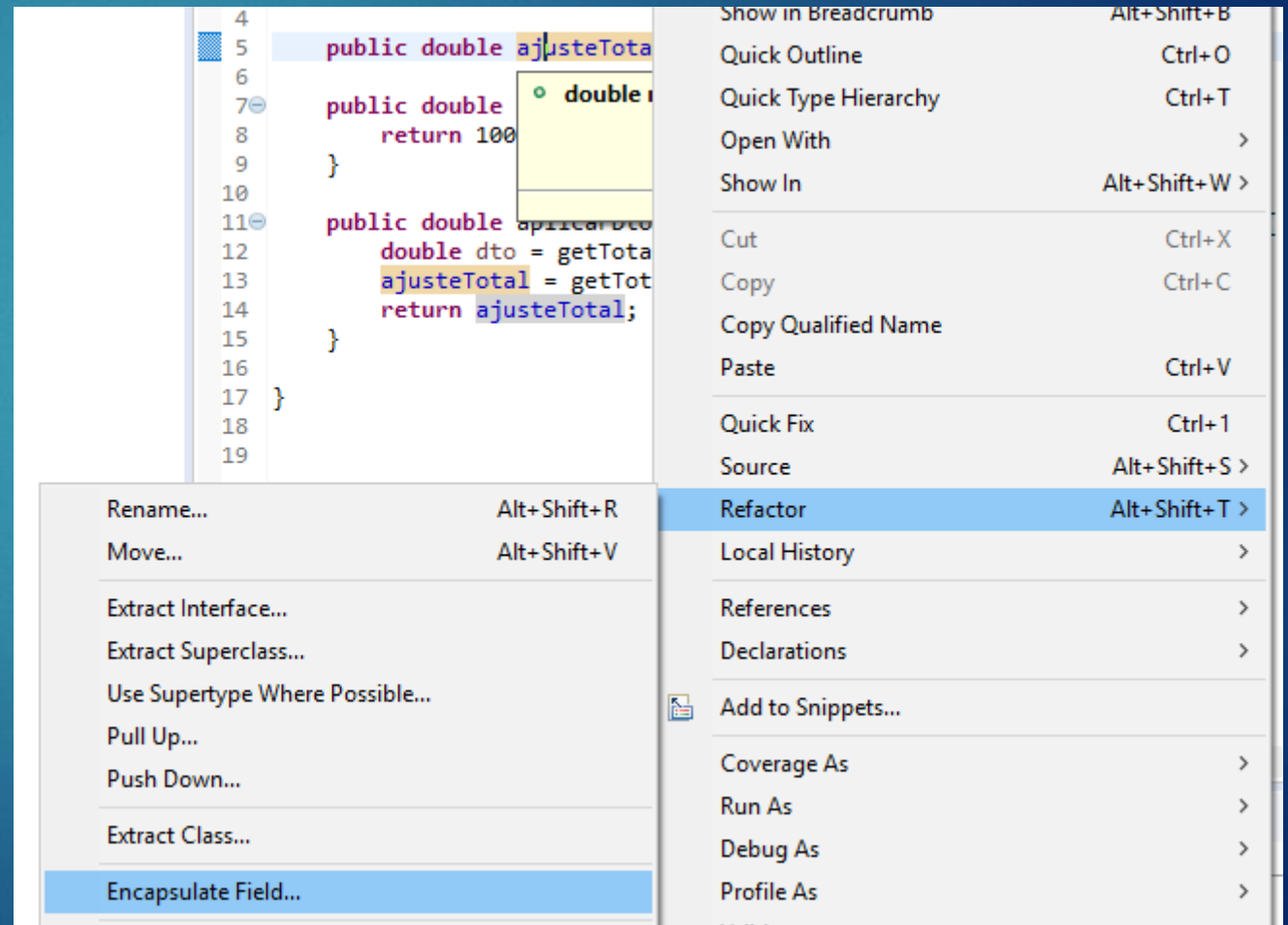
- ▶ Es aconsejable usar métodos getter y setter para cada atributo que se defina en la clase
- ▶ En el siguiente código el atributo `ajusteTotal` no tiene sus métodos getter y setter

```
public class Pedido {  
  
    public double ajusteTotal = 0;  
  
    public double getTotal() {  
        return 100;  
    }  
  
    public double aplicarDto(MoverMetodo moverMetodo, double porcentaje) {  
        double dto = getTotal() * porcentaje;  
        ajusteTotal = getTotal() - dto;  
        return ajusteTotal;  
    }  
}
```

```
public class Pedido {  
  
    private double ajusteTotal = 0;  
  
    public double getAjusteTotal() {  
        return ajusteTotal;  
    }  
  
    public void setAjusteTotal(double ajusteTotal) {  
        this.ajusteTotal = ajusteTotal;  
    }  
  
    public double getTotal() {  
        return 100;  
    }  
  
    public double aplicarDto(MoverMetodo moverMetodo, double porcentaje) {  
        double dto = getTotal() * porcentaje;  
        setAjusteTotal(getTotal() - dto);  
        return getAjusteTotal();  
    }  
}
```

Encapsulado de campos

- En Eclipse nos situaremos sobre el atributo que deseamos crear sus métodos getter y setter, haremos botón derecho, Refaztorizar->Encapsulado de campos



Encapsulado de campos

- ▶ En el cuadro de diálogo que nos aparece, indicaremos:
 - ▶ Nombre del método getter y setter
 - ▶ Donde deseas que se introduzca
 - ▶ Si quieres que incluya comentarios javadoc

