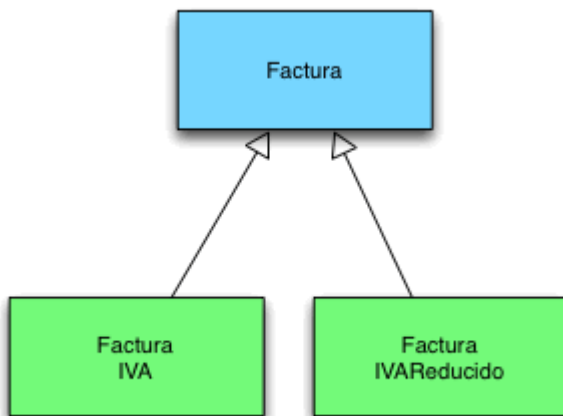


Uno de los patrones de diseño más utilizados en Java es el patron Factory que es un patrón de diseño creacional y que sirve para construir una jerarquía de clases. Sin embargo a veces a la gente le cuesta ver como usar este patrón en su código. Vamos a utilizar un ejemplo sencillo en el que tendremos una jerarquía de clases factura como se muestra a continuación.

CURSO Introducción Patrones Diseño
-75% BLACK FRIDAY
APUNTATE!!



El código será el siguiente:

```
package com.arquitecturajava;

public abstract class Factura {
```

```
private int id;
private double importe;
public int getId() {
    return id;
}
public void setId(int id) {
    this.id = id;
}
public double getImporte() {
    return importe;
}
public void setImporte(double importe) {
    this.importe = importe;
}

public abstract double getImporteIva();
}
```

```
package com.arquitecturajava;

public class FacturaIva extends Factura{

    @Override
    public double getImporteIva() {
        // TODO Auto-generated method stub
        return getImporte()*1.21;
    }
}
```

```
}
```

```
package com.arquitecturajava;

public class FacturaIvaReducido extends Factura{

@Override
    public double getImporteIva() {
        // TODO Auto-generated method stub
        return getImporte()*1.07;
    }

}
```

Como vemos la clase Factura es una clase abstracta de la cual heredan nuestras dos clases concretas que implementan el cálculo del IVA. Vamos a construir una Factoría para que se encargue de construir ambos objetos de la jerarquía.

```
package com.arquitecturajava;

public class FactoriaFacturas {

public static Factura getFactura(String tipo) {

    if (tipo.equals("iva")) {
```

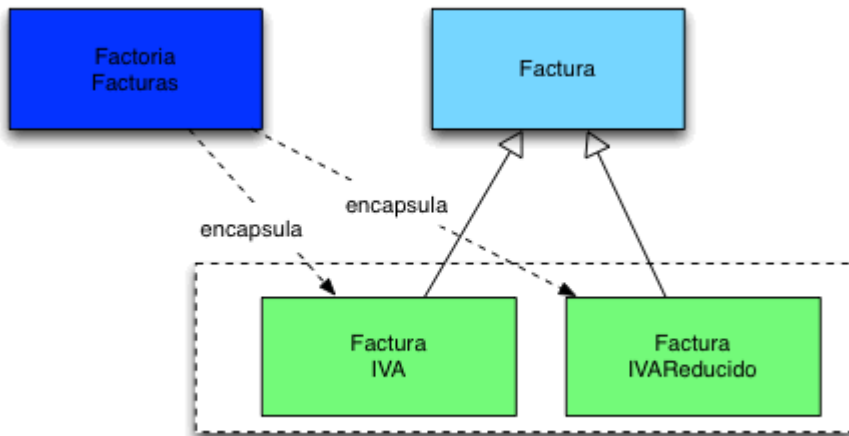
```
    return new FacturaIva();  
}  
else {  
    return new FacturaIvaReducido();  
}  
  
}  
}
```

Si nos fijamos la clase lo único que hace es instanciar un objeto u otro dependiendo del tipo que le solicitemos. Eso en un principio parece poco práctico. Pero vamos a ver como queda el programa main:

```
package com.arquitecturajava;  
  
public class Principal {  
  
    public static void main(String[] args) {  
  
        Factura f= FactoriaFacturas.getFactura("iva");  
        f.setId(1);  
        f.setImporte(100);  
        System.out.println(f.getImporteIva());  
    }  
  
}
```

Nos podemos dar cuenta que el programador ya solo tiene que tratar con el concepto de

Factura para el la clase FacturaIva y FacturaReducido no existen.



Esto permite una simplificación a la hora de trabajar clara. Es cierto que las Factorias se encargan de generar una jerarquía de clases pero su función fundamental es encapsular una jerarquía de objetos y reducir el conjunto de conceptos con los que trabajamos.

CURSO Introducción Patrones Diseño
-75% BLACK FRIDAY
APUNTATE!!

Otros artículos relacionados:

- [Singleton](#)
- [Delegación](#)
- [Adaptadores](#)

CURSO Diseño Orientado Objeto

-75% BLACK FRIDAY
APUNTATE!!