Java Composite Pattern es uno de los patrones más utilizados de programación ya que afecta a la construcción de estructuras complejas y aporta una solución elegante a problemas clásicos. Este patrón es uno de los que cuesta entender. Vamos a ver un par de ejemplos que nos ayuden a clarificar como y cuando utilizarlo

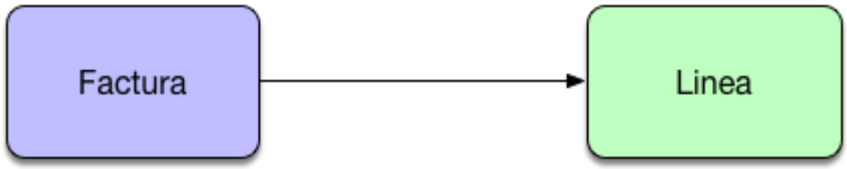
El concepto de Java Composite Pattern

Para entender el uso de este patrón de diseño nos vamos a apoyar inicialmente en el concepto de Factura y LineaFactura. Son dos conceptos sencillos a nivel de programación.

```
package com.arquitecturajava.composite2;

import java.util.ArrayList;
import java.util.List;

public class Factura {

	private String concepto;
	private List<LineaFactura> lineas= new
ArrayList<LineaFactura>();
	public String getConcepto() {
		return concepto;
	}

	public void setConcepto(String concepto) {
		this.concepto = concepto;
	}
```

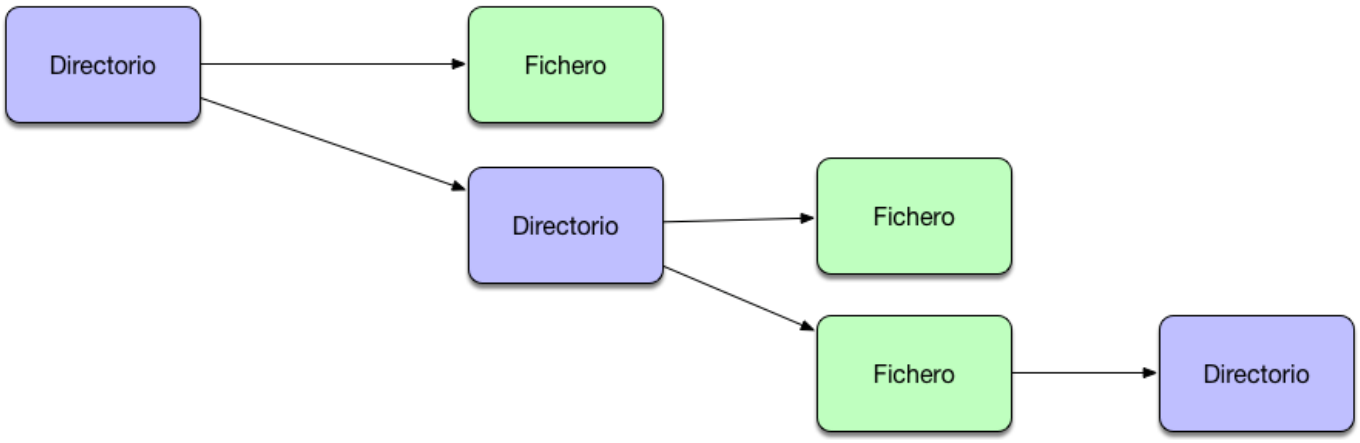
```
        public List&lt;LineaFactura&gt; getLineas() {
                return lineas;
        }

        public void setLineas(List<LineaFactura> lineas) {
                this.lineas = lineas;
        }

}
```

¿Qué relación existe entre ambos a nivel de Clases? . Es una pregunta sencilla , la relación es de composición una Factura incluye varias LineasFactura.

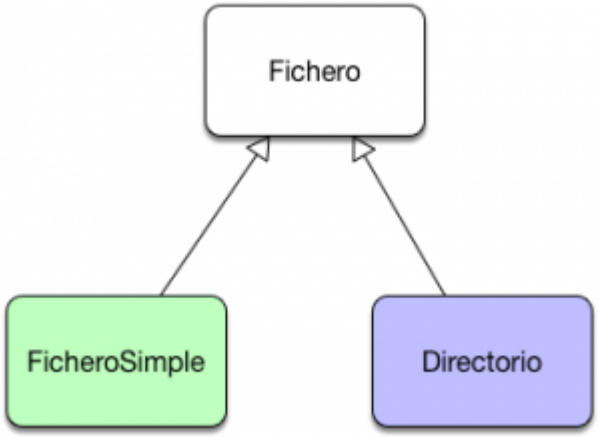


Hasta aquí todo perfecto . Veamos otra relación que en principio parece idéntica un Directorio contiene varios ficheros. La relación vuelve a ser de composición y en principio parece idéntica a la anterior.

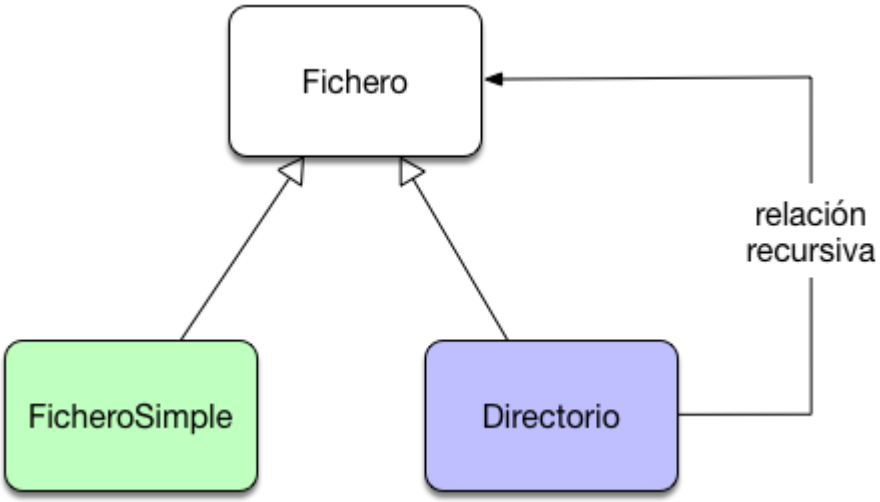
Lamentablemente no es así porque un Directorio puede contener otros directorios que contendrán otros ficheros. De esta forma la relación puede ampliarse hasta el infinito. ¿Cómo podemos organizar esto?



Se trata de una relación recursiva. El concepto que tenemos que entender es que un Directorio también es un un tipo de Fichero. Por lo tanto podríamos diseñar las relaciones de la siguiente forma.



La peculiaridad que esto tiene es que ademas un directorio contiene ficheros que pueden ser ficheros sencillos o otros directorios. Por lo tanto acabamos de definir una relación recursiva.



Este es el concepto de Java Composite Pattern.Vamos a ver un ejemplo en código:

```
package com.arquitecturajava.composite1;

import java.util.List;

public abstract class Fichero {

	private String nombre;
	public String getNombre() {
		return nombre;
	}

	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
```

```
        public Fichero(String nombre) {
                super();
                this.nombre = nombre;
        }
        public abstract boolean esDirectorio() ;
        public abstract List<Fichero> getFicheros();
}
```

Acabamos de construir la clase padre Fichero . La hemos definido como clase abstracta que contiene los métodos esDirectorio() y getFicheros(). Métodos que serán implementados por las clases hijas para dar una funcionalidad u otra.

```
public class FicheroSimple extends Fichero {

        public FicheroSimple(String nombre) {
                super(nombre);
        }

        @Override
        public boolean esDirectorio() {
                return false;
        }

        @Override
        public List<Fichero> getFicheros() {
                throw new RuntimeException(" un fichero no contiene
directorios");
```

```
        }

}
```

```
package com.arquitecturajava.composite1;

import java.util.ArrayList;
import java.util.List;

public class Directorio extends Fichero{

        public Directorio(String nombre) {
                super(nombre);
        }

        private List<Fichero> ficheros= new ArrayList<>();

        public List<Fichero> getFicheros() {
                return ficheros;
        }

        public void setFicheros(List<Fichero> ficheros) {
                this.ficheros = ficheros;
        }
        public void addFichero (Fichero f) {
                ficheros.add(f);
        }
```

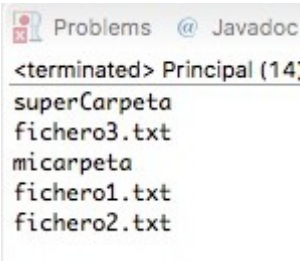
```
        @Override
        public boolean esDirectorio() {

                return true;
        }
}
```

Una vez tenemos la estructura de clases construida , nos queda construir un programa principal que acceda de forma recursiva a ella.

```
package com.arquitecturajava.composite1;

public class Principal {

        public static void main(String[] args) {
                Fichero f1= new FicheroSimple("fichero1.txt");
                Fichero f2= new FicheroSimple("fichero2.txt");
                Directorio d1= new Directorio("micarpeta");
                d1.addFichero(f1);
                d1.addFichero(f2);
                Directorio d2= new Directorio("superCarpeta");
                d2.addFichero(new FicheroSimple("fichero3.txt"));

                d2.addFichero(d1);
                recorrerFicheros(d2);
        }
        public static void recorrerFicheros(Fichero fichero ) {
                        System.out.println(fichero.getNombre());
                        if(fichero.esDirectorio()) {
```

```
                                    for (Fichero f:fichero.getFicheros())
{
                                            recorrerFicheros(f);
                                    }
                            }

        }
}
```

Como se puede ver accedemos de forma recursiva a las diferentes carpetas y ficheros. El resultado nos mostrará la estructura por completo.



Otros artículos relacionados

1. Java 8 Factory Pattern y su implementación
2. El concepto de Java Proxy Pattern
3. Command Pattern en Java y la gestion de tareas

Externos

Command Pattern