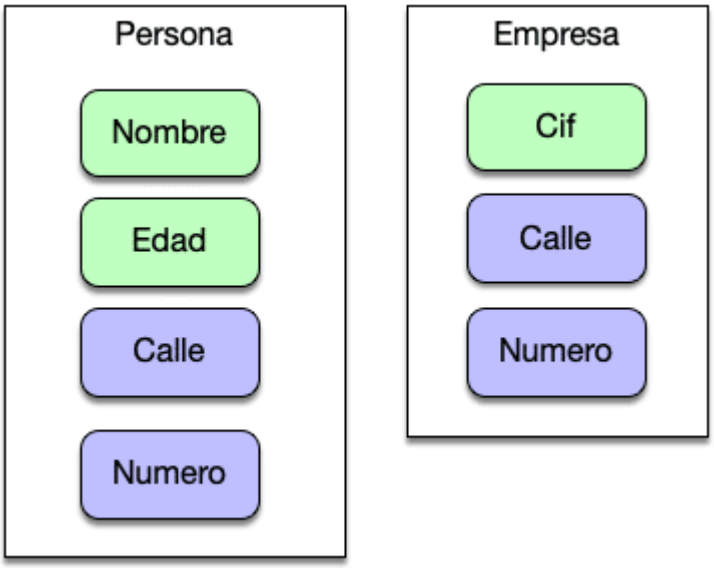
Tabla de Contenidos

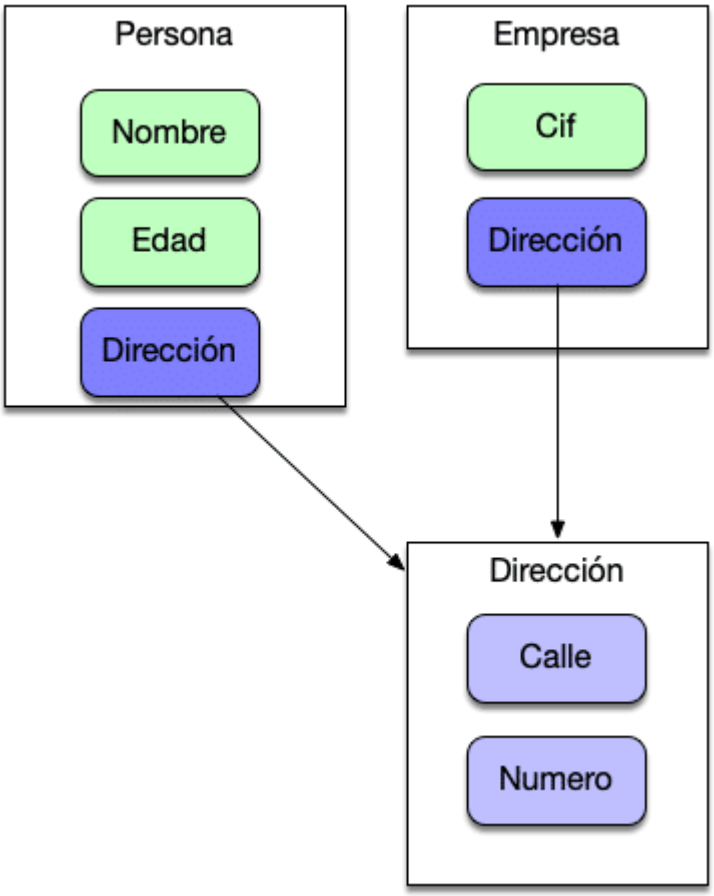
- Java Composicion y su código
- Composicion y Main
- Conclusión
- Otros artículos relacionados

El uso de Java Composicion en el trabajo diario es muy habitual . Cuando uno comienza a programar siempre parece que la herencia es la característica de un lenguaje de programación orientado a objeto que aportar mayor reutilización . Sin embargo esto no es cierto ya que la herencia se usa normalmente para categorizar es decir por ejemplo Persona y Deportista un Deportista es una Persona . Fuera de este tipo de opciones el uso de herencia no suele encajar y se opta por un enfoque de composición una clase usa otras clases. Un ejemplo cotidiano podría ser el diseñar las Clases Persona , Empresa y Dirección . En un ejemplo muy básico tanto la Persona como la Empresa contienen ellas mismas la información de la dirección como propiedades individuales.



Este enfoque no favorece la reutilización ya que ambas clases comparten las propiedades y nos encontramos ante una situación sencilla de repetición de código . Para solventar este problema debemos apostar por un enfoque de Java Composicion . En donde tanto Persona como Empresa se apoyan en la clase Dirección para gestionar el concepto de calle y

numero.



Java Composicion y su código

Vamos a ver este diagrama implementado en código Java:

```
package com.arquitecturajava;

public class Persona {

        private String nombre;
        private int edad;
        private Direccion dirección;
        public Persona(String nombre, int edad) {
```

```
                super();
                this.nombre = nombre;
                this.edad = edad;
        }
        public Direccion getDirección() {
                return dirección;
        }
        public void setDirección(Direccion dirección) {
                this.dirección = dirección;
        }
        public String getNombre() {
                return nombre;
        }
        public void setNombre(String nombre) {
                this.nombre = nombre;
        }
        public int getEdad() {
                return edad;
        }
        public void setEdad(int edad) {
                this.edad = edad;
        }
}
```

```
package com.arquitecturajava;

public class Empresa {

        private String cif;

        private Direccion dirección;
        public Direccion getDirección() {
```

```
                return dirección;
        }

        public void setDirección(Direccion dirección) {
                this.dirección = dirección;
        }

        public String getCif() {
                return cif;
        }

        public void setCif(String cif) {
                this.cif = cif;
        }
}

public class Direccion {

        private String calle;
        private int numero;
        public String getCalle() {
                return calle;
        }
        public void setCalle(String calle) {
                this.calle = calle;
        }
        public int getNumero() {
                return numero;
        }
        public void setNumero(int numero) {
                this.numero = numero;
        }
```

```
	public Direccion(String calle, int numero) {
		super();
		this.calle = calle;
		this.numero = numero;
	}
}
```

De esta manera es muy sencillo compartir a través de Java Composición el concepto de Dirección entre ambas clases sin la necesidad de utilizar herencia.

Composicion y Main

Es momento de construir un programa main que se encargue de construir un objeto Empresa y otro objeto Persona cada uno de ellos asociado a una dirección diferente,

```
package com.arquitecturajava;

public class Principal {

	public static void main(String[] args) {
		Direccion d1= new Direccion("calle A",3);
		Direccion d2= new Direccion("calle B",7);

		Persona p= new Persona("pepe",20);
		p.setDirección(d1);
		Empresa e= new Empresa();
		e.setCif("1A");
		e.setDirección(d2);
		System.out.println(p.getDirección().getCalle());
		System.out.println(e.getDirección().getCalle());
	}
```

```
}
```

Conclusión

Acabamos de usar composición para añadir tanto a la Persona como a la Empresa el concepto de Dirección de tal forma que este pueda estar ubicado en una clase diferente y tanto Persona como Empresa lo reutilicen.

Otros artículos relacionados

- Java Herencia
- Java Interfaces
- Java APIs Diseño y Homogeneidad
- Curso gratuito Java y Herencia