Tabla de Contenidos

Java Diseño y Objetos . ¿Cómo podemos mejorar nuestro diseño orientado a objeto ? . En muchas ocasiones me he encontrado con que los conceptos de programación orientada a objeto son conceptos que todos conocemos y que todos usamos normalmente en nuestro día a día . Creamos clases , instanciamos objetos aplicamos herencia y composición . Pero en muchos casos nuestros diseños se nos quedan un poco cortos .Esto es debido a que no pensamos lo suficiente en “Objetos” y “Clases” como conceptos fundamentales a la hora de desarrollar en nuestro día a día. Vamos a ver un ejemplo sencillo de lo que comento . Para ello usaremos el concepto de Caja que tiene un ancho , un alto y un largo .

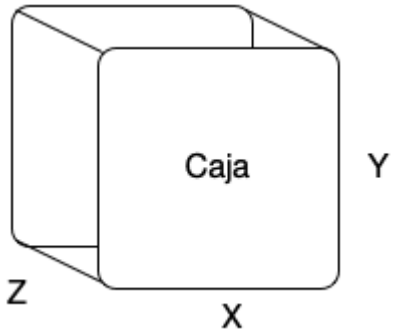
```
package com.arquitecturajava;

public class Caja {

	private double x;
	private double y;
	private double z;
	private double peso;
	public double getX() {
		return x;
	}
	public void setX(double x) {
		this.x = x;
	}
```

```
        public double getY() {
                return y;
        }
        public void setY(double y) {
                this.y = y;
        }
        public double getZ() {
                return z;
        }
        public void setZ(double z) {
                this.z = z;
        }
        public double getPeso() {
                return peso;
        }
        public void setPeso(double peso) {
                this.peso = peso;
        }
        public Caja(double x, double y, double z, double peso) {
                super();
                this.x = x;
                this.y = y;
                this.z = z;
                this.peso = peso;
        }
        public double getVolumen() {
                return this.x*this.y*this.z;
        }
}
```

Esta clase define el concepto de Caja con coordenadas x,y,z que definen su volumen y una propiedad peso que define el peso que la Caja tiene .

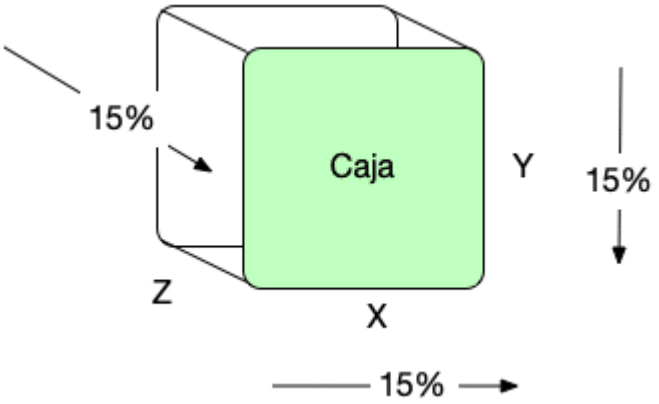


Java Diseño y Objetos y sus opciones

Hasta aquí todo bastante razonable. ¿Ahora bien imaginémonos que necesitamos el volumen del hueco mínimo en el que podemos ubicar la Caja?. En principio se trata de algo sencillo ya que con tener un volumen un 15 % mayor al de la Caja es suficiente para ubicar el hueco. Mucha gente suele resolver este problema modificando el código y realizando una sencilla operación de multiplicación.

```
public double getVolumenHueco() {
                return this.x*this.y*this.z*1.15;
        }
```

Todo el mundo se queda contento en un principio . Hasta que después de pensar un rato nos damos cuenta que no vale solamente con tener un volumen un 15% mayor sino que este volumen debe ser un 15% mayor en cada una de las coordenadas .



Algo que es evidente para todos y que no es tan sencillo de implementar. Muchas personas

me comentan que bueno que habría que tener un control de los métodos set , que rehacer los cálculos etc.

Orientación a Objeto

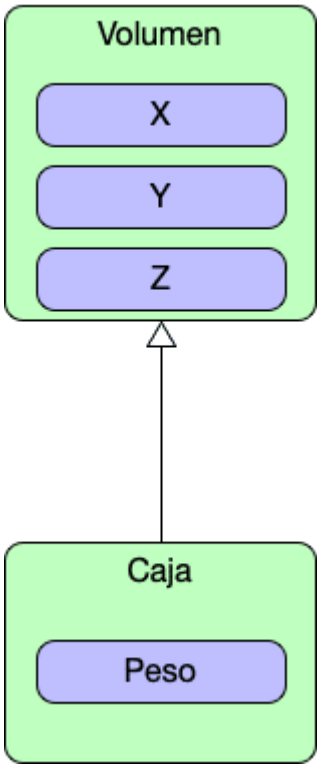
Muchas veces las soluciones pasan por tener un enfoque más orientado a objeto. Por ejemplo podemos hacer que el volumen que se retorna en vez de ser un volumen sea una Caja en si , la cual contiene el volumen.

```
        public Caja getVolumenHueco() {
                return new
Caja(this.x*1.15,this.y*1.15,this.z*1.15,this.peso);
        }
```

Esta solución es más orientada a objeto ya que al devolver una nueva Caja con el nuevo volumen esperado podemos acceder a cada una de las Coordenadas x,y ,z .

Java Diseño y Objetos (Herencia)

Muchas veces las cosas se pueden afinar más cuando nos encontramos con problemas conceptuales . En este caso el volumen que obtenemos contiene un peso que quizás no sea la característica que queremos en nuestro código. Podemos diseñar una clases superior Volumen. Esta clase define el concepto de Volumen que cualquier objeto ocupa.




```
package com.arquitecturajava;

public class Volumen {

	protected double x;
	protected double y;
	protected double z;

	public double getX() {
		return x;
	}

	public void setX(double x) {
		this.x = x;
	}
```

```
	public double getY() {
		return y;
	}

	public void setY(double y) {
		this.y = y;
	}

	public double getZ() {
		return z;
	}

	public void setZ(double z) {
		this.z = z;
	}

	public Volumen(double x, double y, double z) {
		super();
		this.x = x;
		this.y = y;
		this.z = z;
	}

}
```

Una vez tenemos esta clase creada . La clase Caja queda mucho más sencilla y puede devolver de forma más elegante el volumen mínimo que contiene.

```
package com.arquitecturajava;

public class Caja extends Volumen {
```

```
        private double peso;
        public double getPeso() {
                return peso;
        }
        public void setPeso(double peso) {
                this.peso = peso;
        }
        public Caja(double x, double y, double z, double peso) {
                super(x,y,z);
                this.peso = peso;
        }
        public double getVolumen() {
                return this.x*this.y*this.z;
        }
        public Volumen getVolumenHueco() {
                return new
Volumen(this.x*1.15,this.y*1.15,this.z*1.15);
        }
}
```

Conclusiones

Al hablar de Java Diseño y Objetos muchas veces se nos olvida que la programación orientada a objeto , implica un mayor diseño a nivel conceptual de “Clases” las cuales mejoran lo construido anteriormente.

Otros artículos relacionados

- Java new String y la creación de objetos
- ¿Que es un Java Bean?
- ¿Que es un Java Optional?