Tabla de Contenidos

- Sencillez vs Simplicidad
- ¿Cómo podemos mejorar las cosas?
- Sencillez y APIS
- Usando Java VarArgs
- APIS y Homogeneidad
- Simplicidad vs Sencillez
- Otros artículos relacionados

¿Simplicidad vs Sencillez? . Esta es una pregunta que me hacen mucho cuando imparto formaciones. A veces son conceptos que parecen idénticos pero lamentablemente no lo son aunque muchas veces se intercambien . Vamos a explicarlos un poco más a detalle. Supongamos que disponemos de la clase Persona .[ihc-hide-content ihc_mb_type="show" ihc_mb_who="4" ihc_mb_template="1"]

```
package com.arquitecturajava;

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

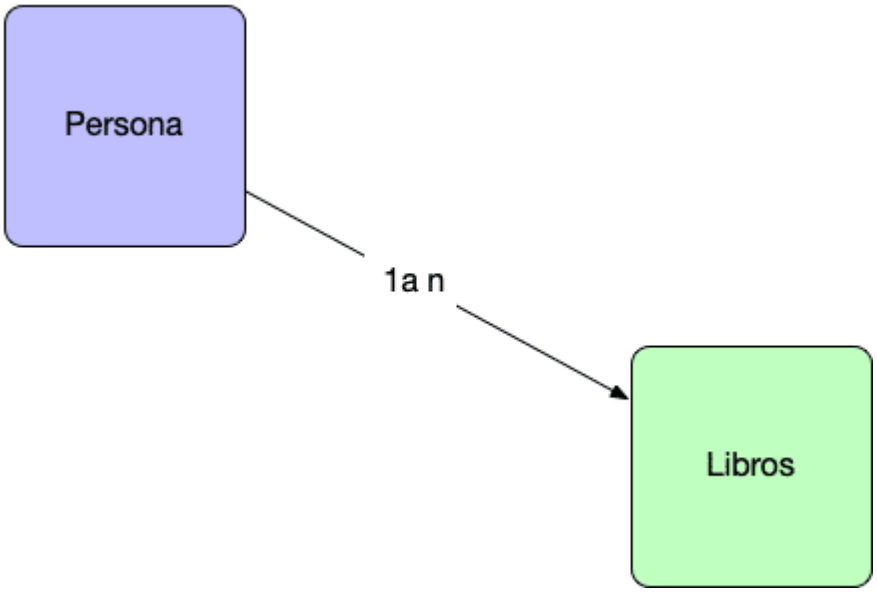
public class Persona {

	private String dni;
	private String nombre;
	private int edad;
	private List<Libro> libros= new ArrayList<Libro>();
	public List<Libro> getLibros() {
		return libros;
```

```
        }
        public void setLibros(List<Libro> libros) {
                this.libros = libros;
        }
        public String getDni() {
                return dni;
        }
        public void setDni(String dni) {
                this.dni = dni;
        }
        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;
        }
        public Persona(String dni, String nombre, int edad) {
                super();
                this.dni = dni;
                this.nombre = nombre;
                this.edad = edad;
        }
}
```

Esta clase tiene una relación con una colección de Libros .Cada persona tiene un conjunto

de libros que lee.



Veamos su código:

```
package com.arquitecturajava;

public class Libro {

	private String isbn;
	private String titulo;
	private String autor;
	public String getIsbn() {
		return isbn;
	}
	public void setIsbn(String isbn) {
		this.isbn = isbn;
	}
```

```
        public String getTitulo() {
                return titulo;
        }
        public void setTitulo(String titulo) {
                this.titulo = titulo;
        }
        public String getAutor() {
                return autor;
        }
        public void setAutor(String autor) {
                this.autor = autor;
        }
        public Libro(String isbn, String titulo, String autor) {
                super();
                this.isbn = isbn;
                this.titulo = titulo;
                this.autor = autor;
        }
}
```

Todo es correcto y mucha gente me dice que hemos diseñado un API sencilla para manejar las Personas y su relación con los Libros.

Sencillez vs Simplicidad

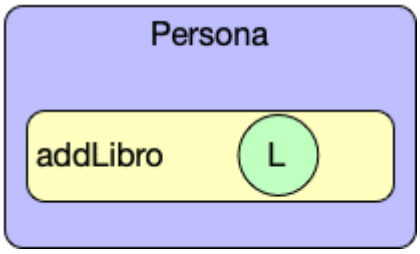
Realmente lo que hemos desarrollado es un API simple en el cual las cosas que podemos hacer son limitadas . Si queremos añadir un Libro a una Persona no nos quedará más remedio que obtener de la Persona la colección de Libros y añadir el Libro:

```
Persona p= new Persona("123","pepe",30);
                Libro l= new Libro("1","java","pedro");
                p.getLibros().add(l);
```

No es quizás el código más elegante del mundo . Esto se debe a que no hemos desarrollado un API sencilla para las operaciones fundamentales que un agregado necesita.

¿Cómo podemos mejorar las cosas?

Una solución rápida es añadir un método addLibro a la clase Persona.



Veamos el código :

```
public void addLibro(Libro l) {
                libros.add(l);
        }
```

Esto hará que nuestras operaciones sean más sencillas de manejar.

```
package com.arquitecturajava;
```

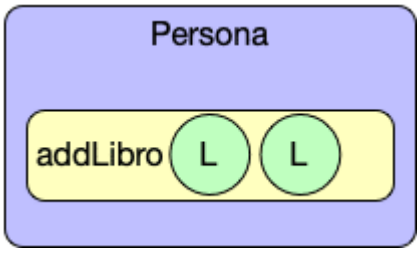
```
public class Principal3 {

	public static void main(String[] args) {
		Persona p= new Persona("123","pepe",30);
		Libro l= new Libro("1","java","pedro");
		p.addLibro(l);
	}

}
```

Sencillez y APIS

¿Es suficiente? . Probablemente no ya que en la mayoría de las ocasiones podemos querer añadir varios Libros a la Persona de golpe.



Usando Java VarArgs

Veamos como se realiza esta operativa.

```
public void addLibro(Libro... nuevos) {
```

```
                Collections.addAll(libros, nuevos);
        }
```

Usamos las capacidades de varargs de Java y pasamos un numero de argumentos variables a una sobrecarga del método addLibro() . De esta forma en el programa main podemos añadir varios libros de forma directa.

```
package com.arquitecturajava;

public class Principal3 {

        public static void main(String[] args) {
                Persona p= new Persona("123","pepe",30);
                Libro l= new Libro("1","java","pedro");
                Libro l2= new Libro("2","net","juan");
                p.addLibro(l,l2);
        }

}
```

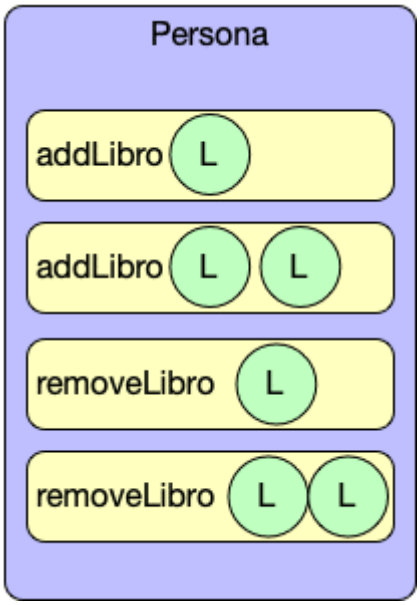
APIS y Homogeneidad

Estas operaciones facilitan el añadir Libros a las Personas diseñando un API sencilla que como podemos ver dista mucho del concepto de API Simple. Igual que hemos añadido Libros podemos implementar los métodos de eliminar. En este caso los denominare remove.

```
        public void removeLibro(Libro l) {
                libros.remove(l);
```

```
}
public void removeLibro(Libro ...otros) {
        libros.removeAll(Arrays.asList(otros));
}
```

Aquí podemos ver como los métodos son muy similares a los de añadir para mantener el API sencilla a través del concepto de homegeneidad ya que los métodos son prácticamente idénticos a los de añadir y complementarios a estos.



Simplicidad vs Sencillez

Tengamos siempre en cuenta que construir un API simple es muy muy sencillo ya que prácticamente no tenemos que hacer nada con lo básico vale . Sin embargo construir un API sencilla es otra cosa cuando hablamos de sencillez hablamos de que el desarrollador se encuentre cómodo y natural a la hora de usar los diferentes métodos. Crear un API sencilla es algo complejo.

Otros artículos relacionados

1. Diseño API REST , rendimiento y OverFetching
2. CRUST y el diseño de APIS
3. REST API y su inmutabilidad

[/ihc-hide-content]