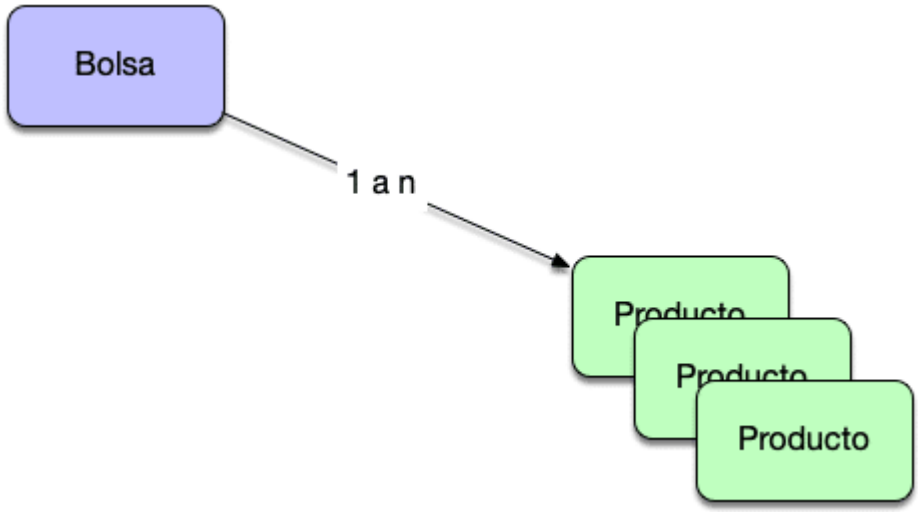
Tabla de Contenidos

El concepto de Java Sobrecarga de métodos es uno de los conceptos más clásicos de programación . La sobrecarga hace referencia a un método al cual se le pueden pasar diferentes tipos o números de argumentos. Vamos a verlo a través de ejemplos sencillos. Para ello partiremos de dos clases . La clase Bolsa y la clase Producto. Una Bolsa contiene varios productos.




```
package com.arquitecturajava;

public class Producto {

	private String nombre;
	private double precio;
	public String getNombre() {
		return nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public double getPrecio() {
		return precio;
	}
	public void setPrecio(double precio) {
		this.precio = precio;
	}
	public Producto(String nombre, double precio) {
		super();
		this.nombre = nombre;
```

```
                this.precio = precio;
        }
}
```

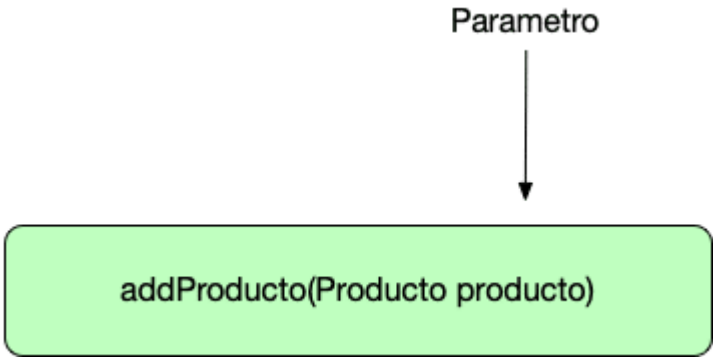
```
package com.arquitecturajava;

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

public class Bolsa {

        private List<Producto> lista = new ArrayList<Producto>();

        public void addProducto(Producto p) {

                lista.add(p);
        }
}
```

El método addProducto

Estamos ante una situación muy sencilla con la clase Bolsa que contiene una lista de Productos y para añadir elementos a la bolsa incorpora el método addProducto

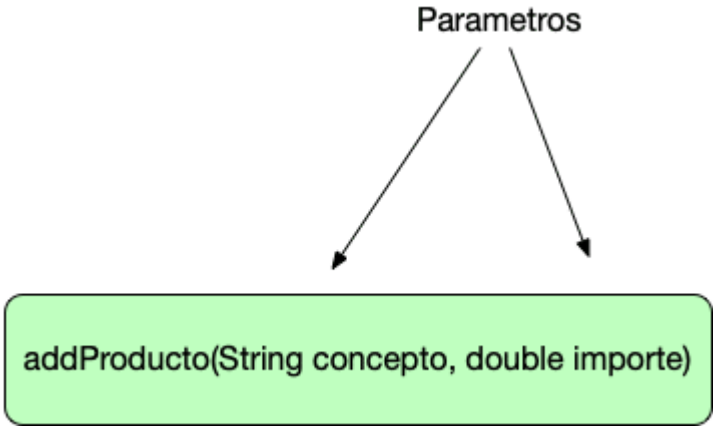
```
 public void addProducto(Producto p) {

                lista.add(p);
        }
```

Este método es elemental y simplemente añade nuevos Productos a la Bolsa. no hace nada más



Java Sobrecarga de métodos

Vamos a sobrecargarlo y generar un método con el mismo nombre pero diferente tipo de argumentos que nos permita por ejemplo añadir también un Producto pero pasándole el concepto y el importe.



Esta es la sobrecarga más sencilla ya que simplemente cambiamos el tipo de parámetros pero seguimos añadiendo el mismo concepto a la lista un producto.

```
public void addProducto(String nombre, double precio) {

                lista.add(new Producto(nombre, precio));
        }
```

El programa principal gana en flexibilidad y permite las dos opciones

```
package com.arquitecturajava;

public class Principal {

	public static void main(String[] args) {
		Producto p1= new Producto ("galletas",3);
		Bolsa b= new Bolsa();
		b.addProducto(p1);
		b.addProducto("chocolate",4);
	}

}
```

Otras Sobrecargas

No solo esta sobrecarga es posible sino que ademas podríamos añadir otras que nos permitan una mayor flexibilidad. Por ejemplo podemos añadir un método que nos permita pasar como parámetro una lista de Productos y añadirlos todos de golpe. Este método no se podrá denominar addProducto ya que este indica que solo añade uno . Se tendrá que denominar addProductos por lo tanto “no es una sobrecarga” pero si añade flexibilidad al código:

```
public void addProductos(List<Producto> productos) {

		lista.addAll(productos);
	}
```

Java varargs

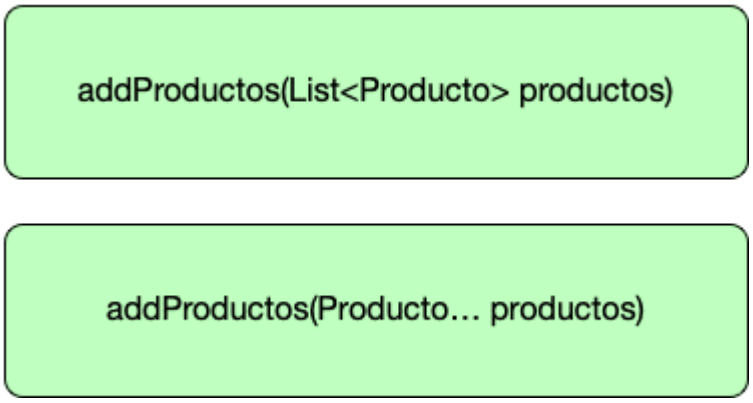
¿Cómo podemos sobrecargarle ? . Bueno podemos usar las capacidades de varargs de Java y añadir una sobrecarga que nos permita pasar varios productos de forma manual.

```
public void addProductos(Producto... productos) {

                for (Producto f : productos) {

                        lista.add(f);
                }
        }
```

Vamos a ver cómo usar ambos.

```
package com.arquitecturajava;

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

public class Principal {

        public static void main(String[] args) {
                Producto p1= new Producto ("galletas",3);
                Bolsa b= new Bolsa();
                b.addProducto(p1);
                b.addProducto("chocolate",4);
                List<Producto> productos= new ArrayList<Producto>();
                productos.add(new Producto("pan",2));
                productos.add(new Producto("leche",3));
                b.addProductos(productos);
                b.addProductos(new Producto("pizza",5),new
Producto("lechuga",3));
        }

}
```

Acabamos de sobrecargar otro método en nuestra clase Bolsa y disponemos de dos implementaciones



Java Sobrecarga y Constructores

El concepto de sobrecarga también se suele aplicar a los constructores ya que estos no son ni más ni menos que métodos de la clase . En nuestro caso podríamos decidir que la Bolsa cuando se crea se le pueda pasar una lista inicial de productos.

```
package com.arquitecturajava;

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

public class Bolsa {

	private List<Producto> lista = new ArrayList<Producto>();

	public Bolsa() {
		super();
	}
```

```
	public Bolsa(List<Producto> lista) {
		super();
		this.lista = lista;
	}

	public void addProducto(Producto p) {

		lista.add(p);
	}

	public void addProducto(String nombre, double precio) {

		lista.add(new Producto(nombre, precio));
	}

	public void addProductos(List<Producto> productos) {

		lista.addAll(productos);
	}

	public void addProductos(Producto... productos) {

		for (Producto f : productos) {

			lista.add(f);
		}
	}

}
```

El código es muy similar al ejemplo anterior pero aplicado a la gestión de constructores.

Otros artículos relacionados

- Java Constructores this() y super()
- Java this vs this()
- Java Overload y preguntas de certificacion
- Curso Java