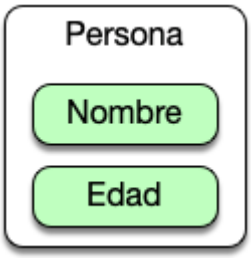
El concepto de Java Default Constructor o constructor por defecto siempre trae locos a todos los desarrolladores cuando están empezando . ¿Para que sirve el constructor por defecto y qué peculiaridades tiene?. Cuando nosotros construimos una clase como Persona esta por defecto incluye un constructor. Es decir si queremos construir un objeto siempre podremos hacerlo porque el constructor por defecto viene integrado con la clase.



Es decir si nosotros creamos la clase Persona con nombre y edad el código será de este estilo

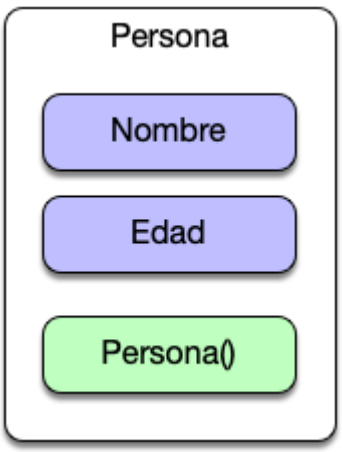
```
public class Persona {
    private String nombre;
    private int edad;

    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;
    }

}
```

Java Default Constructor

En principio no tenemos constructor en la clase pero esto no importa ya que Java nos incluirá uno por defecto que contenta el siguiente código:

```
// Constructor
    public Persona() {
    }
```



Esto nos permite de una manera sencilla instanciar un objeto de la clase:

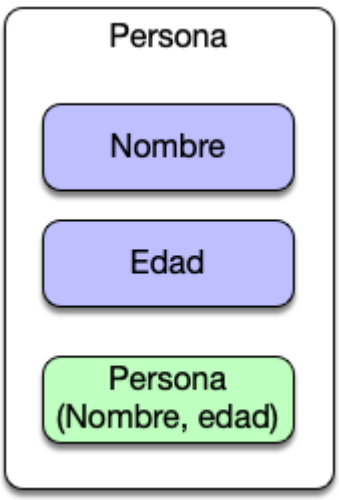
```
package com.arquitecturajava;

public class Principal {

        public static void main(String[] args) {
                Persona p= new Persona();
```

```
                p.setNombre("pepe");
                p.setEdad(20);

                System.out.println(p.getNombre());
                System.out.println(p.getEdad());
        }

}
```

Añadir Constructor a medida

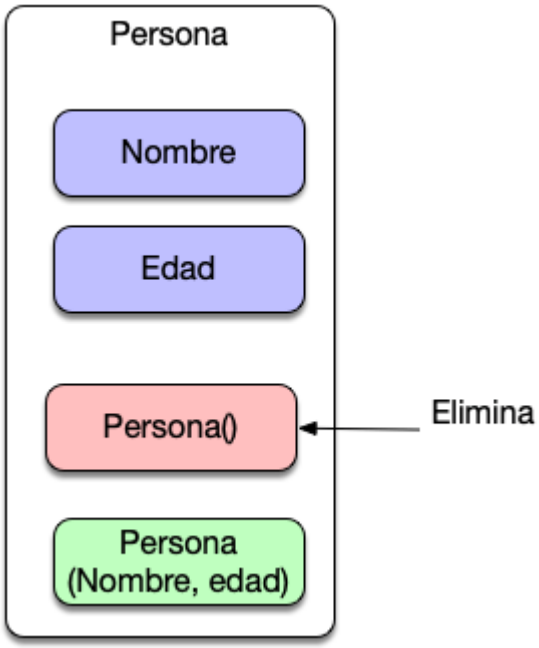
El problema viene cuando alguien decide añadir un constructor a medida . Es decir el programador decide que el constructor va a tener una serie de parámetros:

```
        public Persona(String nombre, int edad) {
                super();
                this.nombre = nombre;
                this.edad = edad;
        }
```



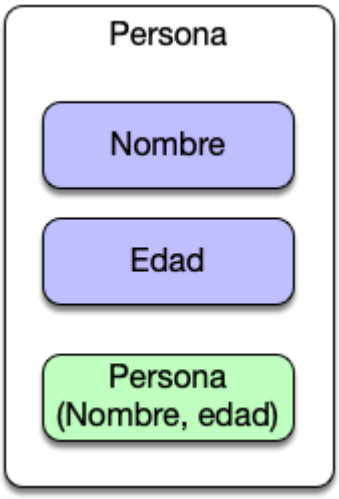
Sorprendentemente el programa anterior deja de compilar . Muchos nuevos programadores

no entienden muy bien a que se debe esto. Pero es muy sencillo si uno añade un constructor a medida, Java no añadirá un constructor por defecto sin parámetros. Ya que el programador ha dejado claro que es lo que quiere definir a nivel de forma de instanciar las clases.



Constructores y Herencia

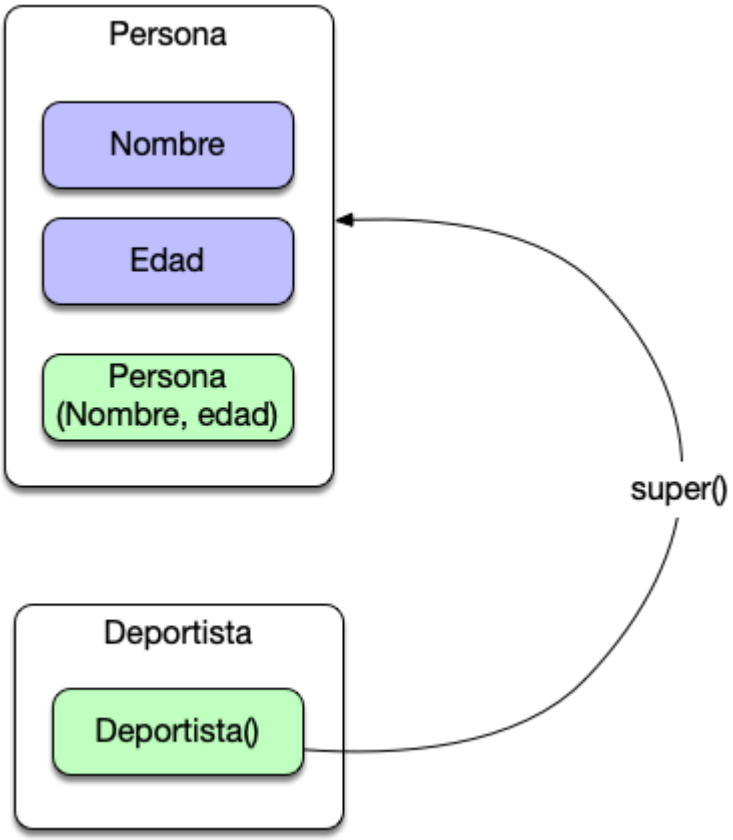
El problema se puede volver más complejo cuando tenemos una jerarquía de herencia . Por ejemplo imaginémonos que disponemos de la clase Persona con la siguiente estructura.



Si disponemos de una clase Deportista que Hereda de Persona esta aunque este vacía dará un problema de compilación.

```
public class Deportista extends Persona{

}
```

Esto se debe a que la clase Deportista al no tener ningún constructor se le añade un constructor por defecto e invocara al constructor por defecto de la clase padre que no existe.



Para solventar este tema necesitamos que el constructor de Deportista invoque al constructor de la clase Persona con varios parámetros.

```
public class Deportista extends Persona{

	public Deportista(String nombre, int edad) {
		super(nombre, edad);
		// TODO Auto-generated constructor stub
	}

}
```

De esta manera dejaremos de tener problemas de compilación . Tengamos siempre en cuenta que gestionar constructores no es tan sencillo como parece en Java.

- Java Constructores this() y super()
- Java Constructor y buenas prácticas
- El concepto de Java constructor reference
- Entity to DTO y Java 8
- Java 8 default methods y extensibilidad