En Java creamos objetos todos los días y para ellos usamos constructores. Todos estamos muy habituados a programarlos pero siempre hay algunos bordes que se nos escapan, vamos a revisarlos. Supongamos que tenemos la siguiente clase:

```
package com.arquitecturajava;

public class Persona {

private String nombre;

public String getNombre() {
return nombre;
}

public void setNombre(String nombre) {
this.nombre = nombre;
}
}
```

Java Constructores por defecto

¿Tiene esta clase algún constructor?. La respuesta es sí toda clase Java si no se le incluye ningún constructor el compilador añade un constructor por defecto. Así pues el código para el compilador sería el siguiente:

```
package com.arquitecturajava;

public class Persona {

private String nombre;
public Persona() {

}

public String getNombre() {
return nombre;
}

public void setNombre(String nombre) {
this.nombre = nombre;
}
}
```

Como vemos para el compilador existe un constructor por defecto vacío. ¿Ahora bien y si el código de nuestra clase incluyera un constructor con un parámetro?

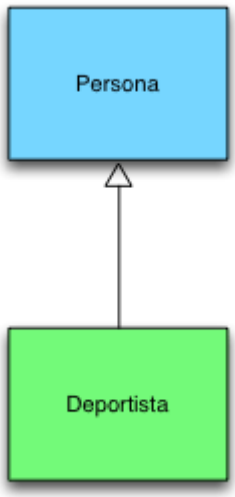
```
package com.arquitecturajava;

public class Persona {

private String nombre;

public Persona(String nombre) {
```

```
this.nombre = nombre;
}

public String getNombre() {
return nombre;
}

public void setNombre(String nombre) {
this.nombre = nombre;
}
}

 
```

En ese caso Java no añade el constructor por defecto. Así pues en ambos casos tenemos un único constructor.

Java constructores y super()

Las dudas con los constructores aparecen ligadas a las jerarquías de clases y a la palabra super(). Supongamos que tenemos la siguiente jerarquía:



En este caso podemos tener dos clases con el siguiente código por simplificar al máximo :

```
package com.arquitecturajava;

public class Persona {

private String nombre;

public String getNombre() {
return nombre;
}

public void setNombre(String nombre) {
this.nombre = nombre;
}
}
```

```
package com.arquitecturajava;

public class Deportista extends Persona{

}
```

Aunque en el código no aparezcan constructores existen dos constructores por defecto uno en cada clase (Persona,Deportista) con el siguiente código:

Persona.java

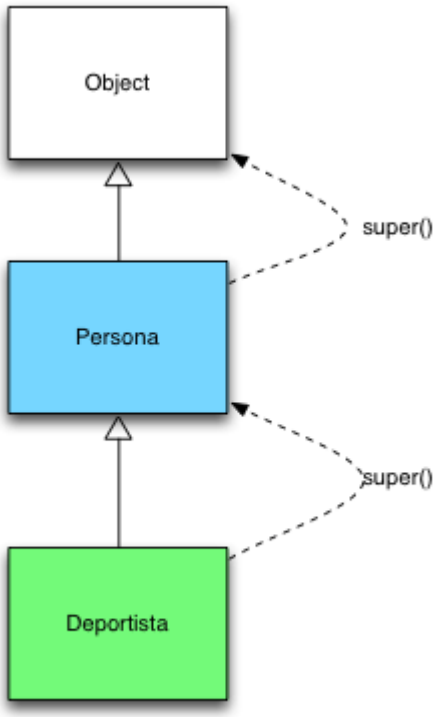
```
public Persona() {
super();
}
```

Deportista.java

```
public Deportista() {
super();

}
```

Como podemos ver todos los constructores llaman por defecto al constructor de la clase superior a través de una llamada a super() (en este caso al constructor por defecto). Esto es debido a que los constructores no se heredan entre jerarquías de clases. Por lo tanto la palabra super() siempre es la primera línea de un constructor e invoca al constructor de la

clase superior que comparta el mismo tipo de parametrización.



Aunque nosotros no pongamos la palabra super() esta siempre será añadida salvo que nosotros la añadamos. Por ejemplo si nuestros constructores tienen parámetros las claúsulas super que deberemos construir serán las siguientes entre Persona y Deportista para que el código compile:

```
package com.arquitecturajava;

public class Persona {

public Persona(String nombre) {
```

```
super();
this.nombre = nombre;
}

private String nombre;

public String getNombre() {
return nombre;
}

public void setNombre(String nombre) {
this.nombre = nombre;
}
}
```

```
package com.arquitecturajava;

public class Deportista extends Persona{

public Deportista(String nombre) {
super(nombre);

}

}
```

Ya que sino el compilador añadirá super() por defecto y el código no compilará al carecer la clase Persona de un constructor por defecto.

Usando this()

La otra posibilidad a super() es el uso de this() en la primera linea de un constructor. Esto lo que hace es invocar a otro constructor que este en la misma clase y que soporte el conjunto de parámetros que le pasamos.

```
package com.arquitecturajava;

public class Deportista extends Persona{

private String deporte;

public String getDeporte() {
return deporte;
}

public Deportista(String nombre, String deporte) {
this(nombre);
this.deporte = deporte;
}

public void setDeporte(String deporte) {
this.deporte = deporte;
}
```

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

}
```

El uso de this() y de super() es excluyente o usamos uno u otro.

Otros artículos relacionados: Java Generic, Java ForEach vs Iterator , Expresiones Lambda

Cecilio Álvarez Caules
Cecilio Álvarez Caules Oracle Java Certified Architech