El concepto de Java Overload es muy conocido por todos los programadores . Java nos permite declarar el mismo método pero con diferentes tipo o número de argumentos en una clase. Esto nos aporta una gran flexibilidad ya que simplifica el número de métodos de los cuales tenemos que acordarnos . Vamos a abordar algunos ejemplos y peculiaridades que soporta.

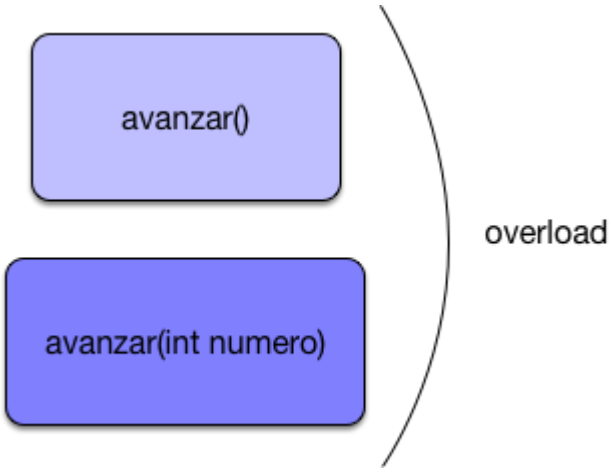
Java Overload Básico

Vamos a usar la tabla Libro para hablar del concepto de overload

```
package com.arquitecturajava.ejemplo1;

public class Libro {

	private int paginas;
	private String titulo;
	private int paginaActual;

	public int getPaginaActual() {
		return paginaActual;
	}

	public void setPaginaActual(int paginaActual) {
		this.paginaActual = paginaActual;
	}

	public Libro(int paginas, String titulo) {
		super();
```

```
            this.paginas = paginas;
            this.titulo = titulo;
        }

        public int getPaginas() {
            return paginas;
        }

        public void setPaginas(int paginas) {
            this.paginas = paginas;
        }

        public String getTitulo() {
            return titulo;
        }

        public void setTitulo(String titulo) {
            this.titulo = titulo;
        }

        public void avanzar() {
            paginaActual++;
        }

        public void avanzar(int numero) {
            paginaActual += numero;

        }
}
```

Acabamos de construir el ejemplo más sencillo de overload. Tenemos un método que soporta que le pasemos 0 o 1 parámetro para avanzar página.



Dependiendo de si pasamos un valor o no se ejecuta un método u otro y avanzamos una o más páginas:

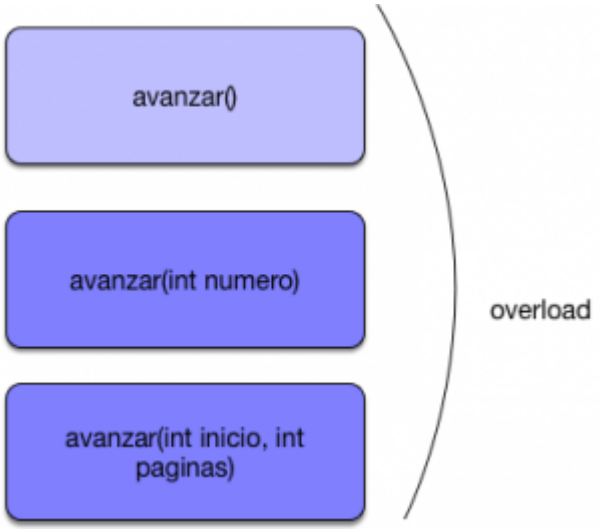
```
package com.arquitecturajava;

public class Ejemplo001 {

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		Libro l= new Libro(250,"java");
		l.avanzar();
		l.avanzar(5);
		System.out.println(l.getPaginaActual());

	}
```

```
}
```

Hemos avanzado seis páginas:

De igual forma se admitiría el paso de varios parámetros por ejemplo para definir el punto de inicio y el bloque de páginas avanzadas.

```
public void avanzar(int inicio, int paginas) {

        paginas = inicio + paginas;
}
```



Hasta aquí todo es muy sencillo . Sin embargo cuando a uno le preguntan en un examen de certificación las cosas se pueden complicar. ¿Es válido tener por ejemplo los siguientes dos métodos?

```
public void avanzar(int numero) {
                paginaActual += numero;

        }

        public void avanzar(long numero) {

                paginaActual+=numero;
        }
```

Si no hay ningún problema ya que son tipos diferentes. ¿Y si fuera de esta manera?

```
public void avanzar(int numero) {

System.out.println("basico")
paginaActual += numero;

}

public void avanzar(Integer numero) {

System.out.println("objeto")

paginaActual+=numero;
}
```

Java Overload curiosidades

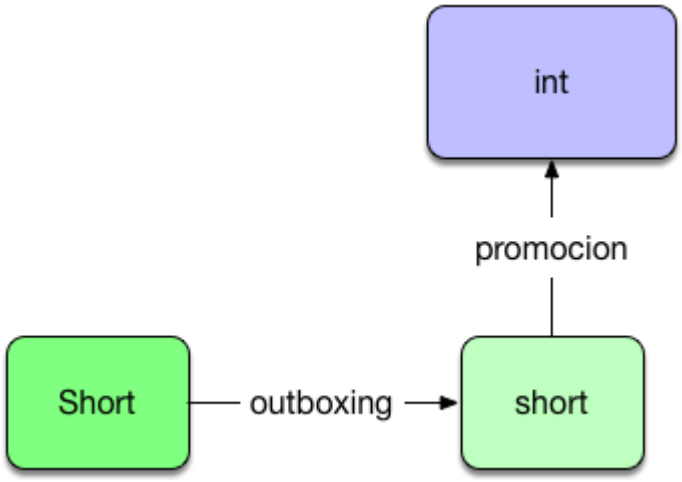
Tampoco sería un problema los métodos están soportados aunque no tenga mucho sentido usarlos de esta forma. ¿Qué pasaría si el programa principal fuera así?:

```
Libro l= new Libro(250,"java");

 short numero1= 5;
 Short numero= new Short(numero1);
 l.avanzar(numero);
```

Este ejemplo es ya más complejo y suele ser una pregunta habitual de un examen de

certificación: Las respuestas posibles son.

1. Se ejecuta el método avanzar(Integer numero)
2. Se ejecuta el método avanzar (int numero)
3. No compila porque no hay ningún método que soporte short
4. No compila porque no puede existir un método Integer y uno int
   Como vemos las respuestas nos generan bastantes dudas. ¿Cual es la buena? .En este caso la máquina virtual comprueba que no existe ningún método que corresponda con un objeto Short. Al no existir el método se realiza un outboxing y se convierte el objeto Short en un tipo básico short. Lamentablemente para este no existe tampoco un método concreto . Sin embargo Java realizará una promoción del tipo básico short al tipo básico int. Ahora si que existe un método que puede encajar. Recordemos que short es más pequeño que int y promocionarlo no da ningún problema.



Así pues si ejecutamos el programa por pantalla imprimirá:

La respuesta correcta era la 2 .Como vemos incluso cosas muy sencillas se pueden complicar cuando entramos a detalle , en este caso el concepto de Java Overload

Otros artículos relacionados:

1. ¿Cuales son las certificaciones Java?
2. Java Herencia y sus limitaciones
3. Java String Pool , un concepto importante
4. Java Boxing