El método Java toString() es de los métodos más utilizados de la clase object cuando estamos trabajando en el día a día. ¿Para qué sirve este método? . Realmente su uso esta muy orientado a presentarnos una información legible del objeto con el que en un momento estamos trabajando. Para ello siempre tendremos que sobreescribirlo y definir el tipo de implementación que más nos encaje, vamos a ver un ejemplo partiendo de la clase Factura:

```
package com.arquitecturajava;

public class Factura {

	private int numero;
	private String concepto;
	private double importe;
	public int getNumero() {
		return numero;
	}
	public void setNumero(int numero) {
		this.numero = numero;
	}
	public String getConcepto() {
		return concepto;
	}
	public void setConcepto(String concepto) {
		this.concepto = concepto;
	}
	public double getImporte() {
		return importe;
	}
	public void setImporte(double importe) {
		this.importe = importe;
	}
```

```
        public Factura(int numero, String concepto, double importe) {
                super();
                this.numero = numero;
                this.concepto = concepto;
                this.importe = importe;
        }
}
```

Esta clase que acabamos de construir nos permite definir varios objetos a través de un programa principal.

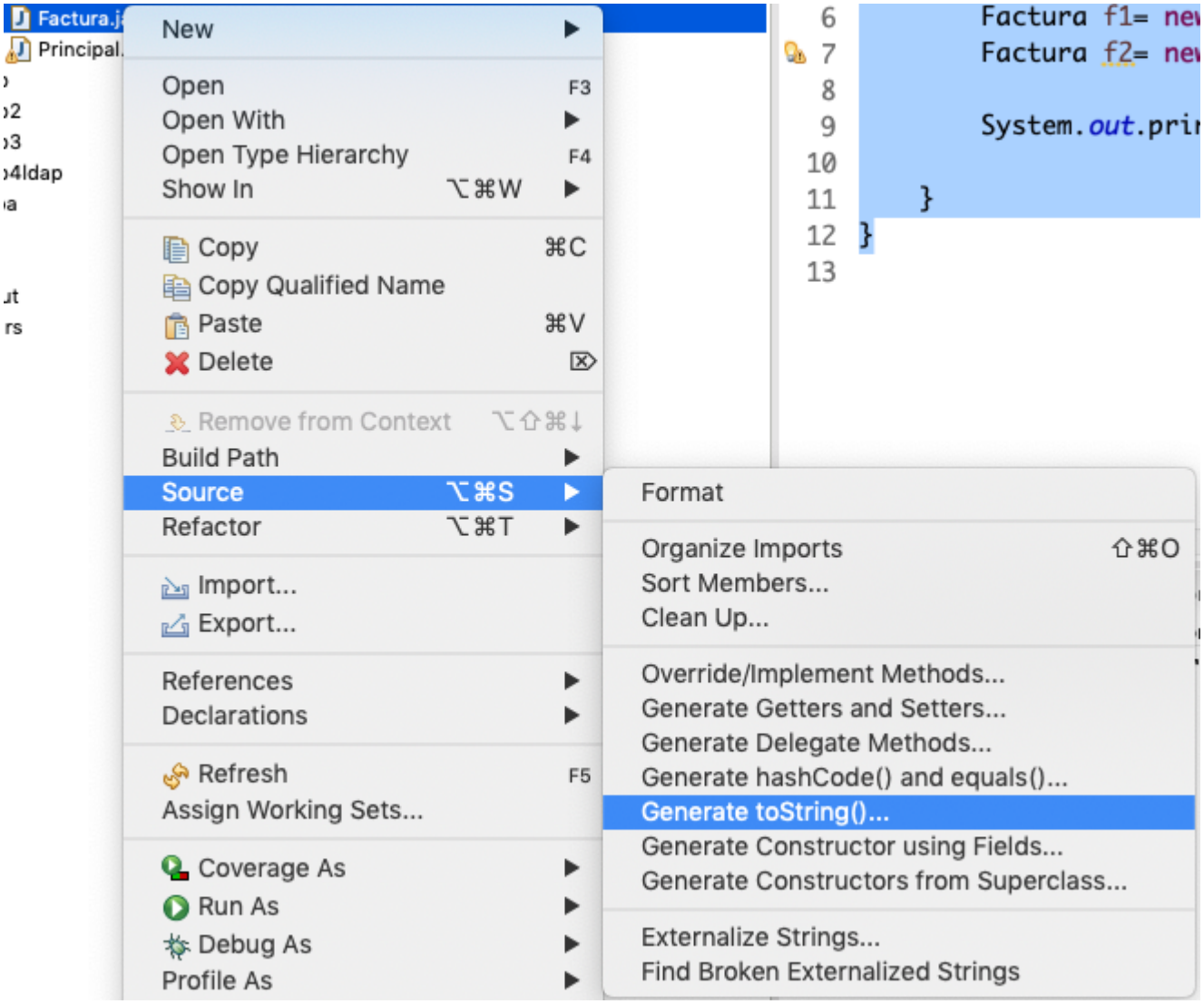
```
package com.arquitecturajava;

public class Principal {
        public static void main(String[] args) {

                Factura f1= new Factura(1,"ordenador",300);
                Factura f2= new Factura(1,"ordenador",300);
                System.out.println(f1);
        }
}
```

Si ejecutamos el código podremos ver en consola el resultado .

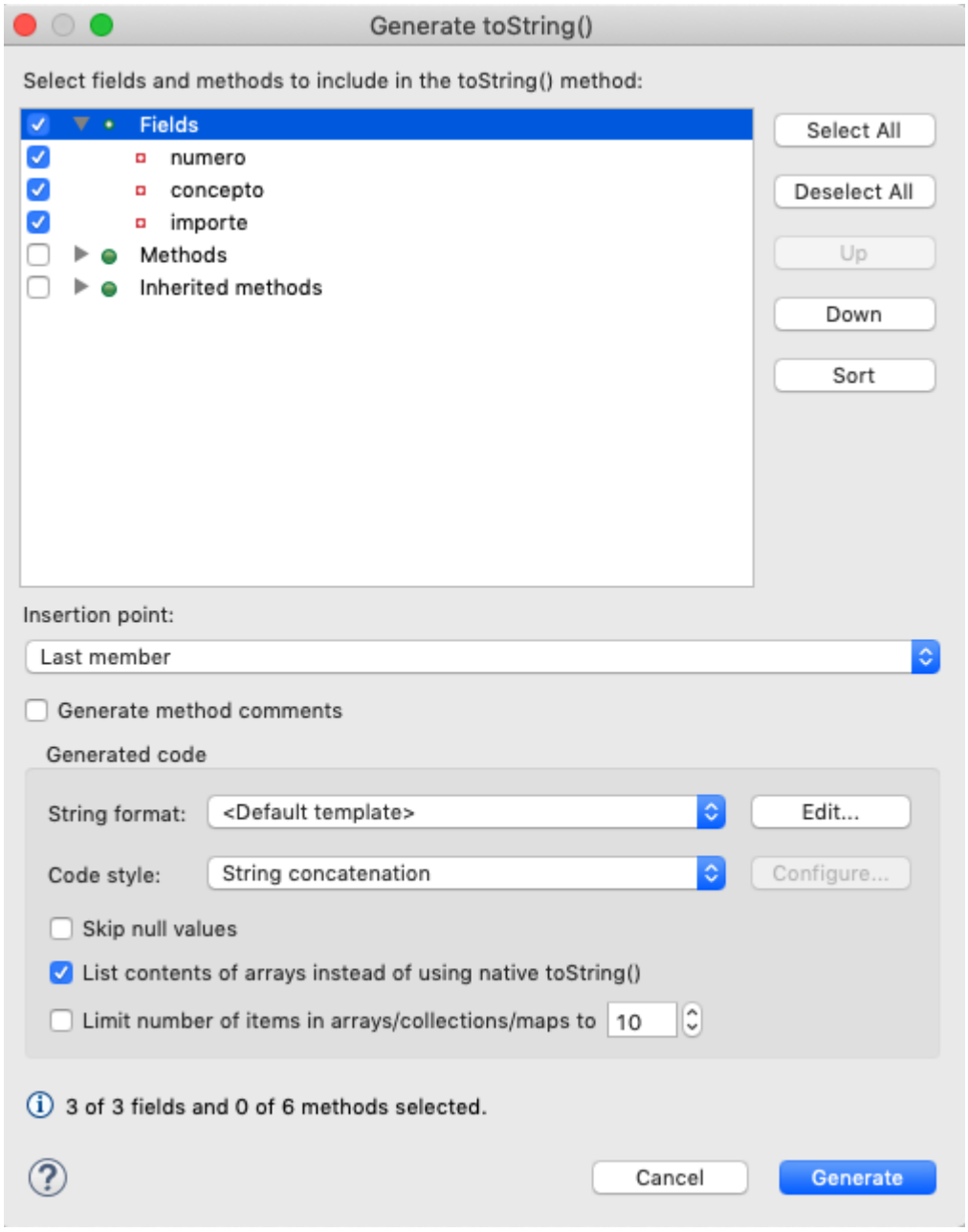
```
com.arquitecturajava.Factura@87aac27
```

Java toString

Nos muestra la referencia al objeto Factura pero no nos aporta una información significativa sobre él. Para solventar estas casuísticas esta el método toString() que permite que cuando nosotros solicitemos información por la consola del objeto esta sea mucho más clara.
Usando Eclipse es muy sencillo de implementar ya que el entorno dispone de un asistente.



Una vez seleccionamos el asistente nos mostrará que campos deseamos que se impriman.



Una vez seleccionados nos añadirá a la clase el método Java toString()

```
@Override
	public String toString() {
		return "Factura [numero=" + numero + ", concepto=" +
concepto + ", importe=" + importe + "]";
	}
```

De esta forma cada vez que solicitemos que se imprima un objeto por la consola podremos ver su contenido de una forma mucho mas natural.

```
Factura [numero=1, concepto=ordenador, importe=300.0]
```

Hemos conseguido que la información sea mucho más clara a la hora de trabajar.

Java ToString y Lambdas

Recordemos que este método se usa habitualmente cuando referenciamos métodos con expresiones lambda y en estos casos también nos puede ser muy útil.

```
package com.arquitecturajava;

import java.util.Arrays;
import java.util.List;

public class Principal {
	public static void main(String[] args) {

		Factura f1= new Factura(1,"ordenador",300);
		Factura f2= new Factura(1,"ordenador",300);
		List<Factura> lista=Arrays.asList(f1,f2);
		lista.forEach(System.out::println);
	}
}
```

En este caso el código define una lista y la recorremos con un Stream.

```
Factura [numero=1, concepto=ordenador, importe=300.0]
Factura [numero=1, concepto=ordenador, importe=300.0]
```

Acostumbremonos siempre a sobreescribir este método en nuestras clases de negocio ya que nos ayudará siempre a clarificar la información que estamos manejando.Java Override

- Java Comparator Interface
- Java Overload
- Java Object JDK
- Java interface
- Java Stream

Cecilio Álvarez Caules
Cecilio Álvarez Caules Oracle Java Certified Architech