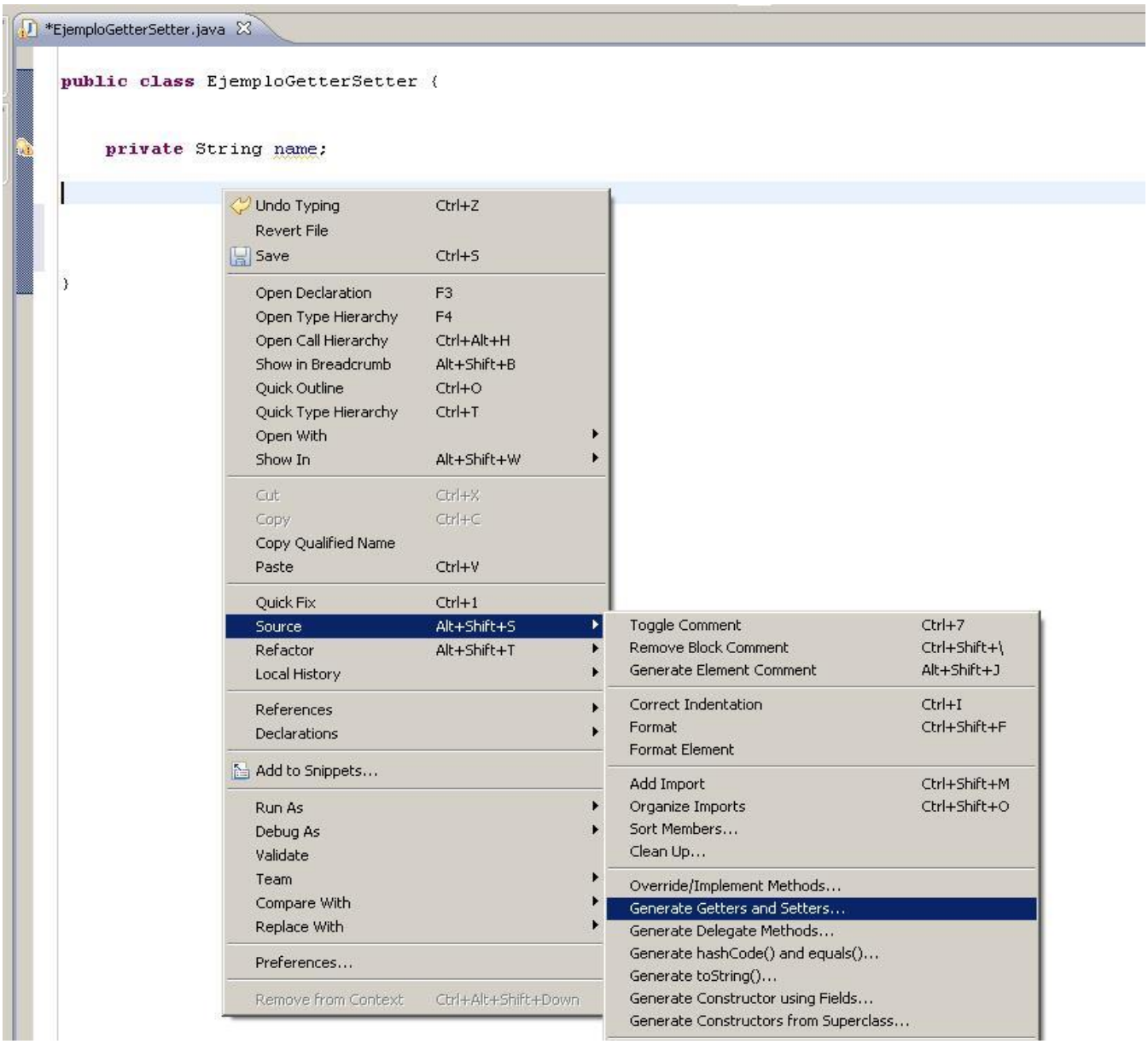
Trucos Eclipse

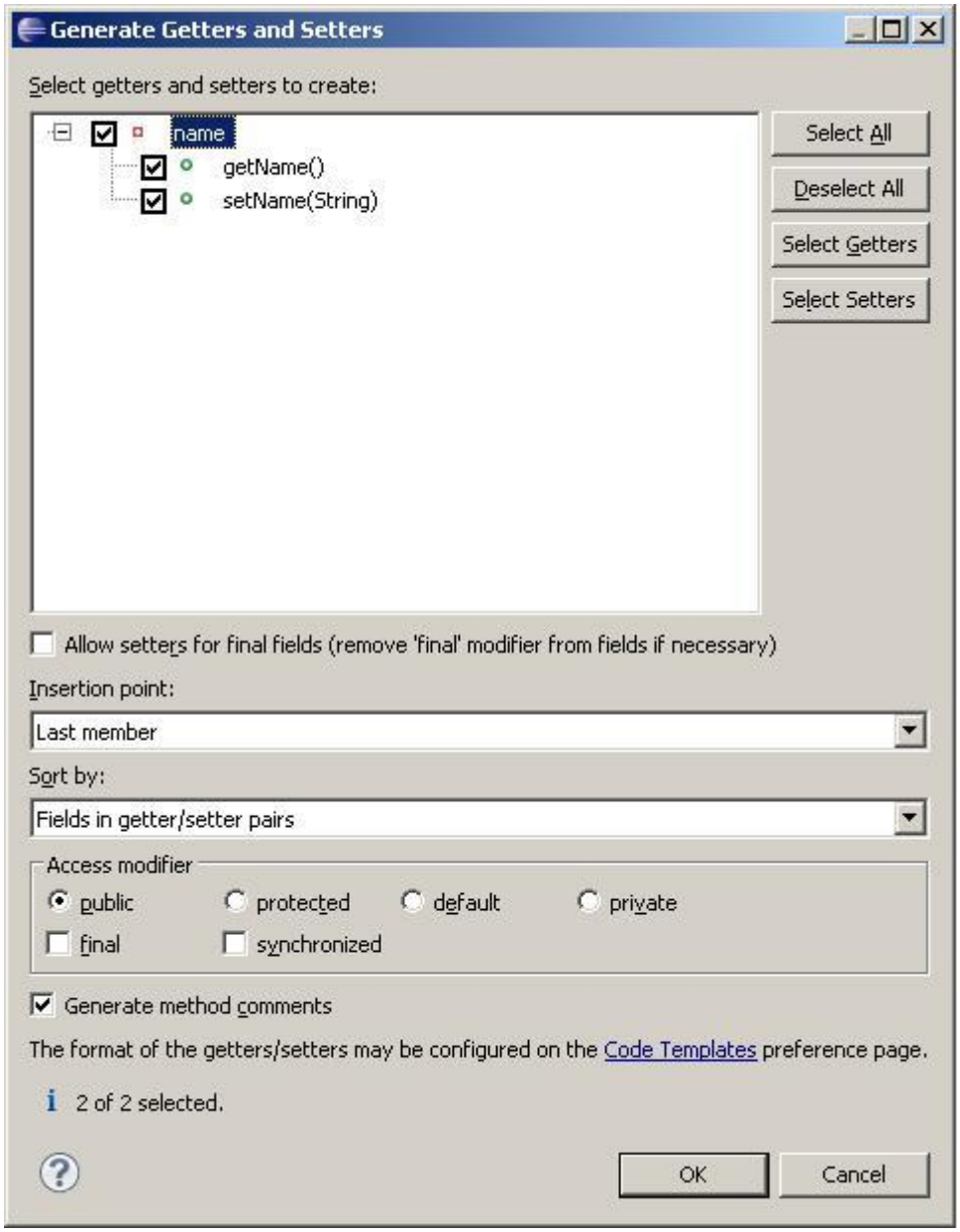
Miguel Sallent

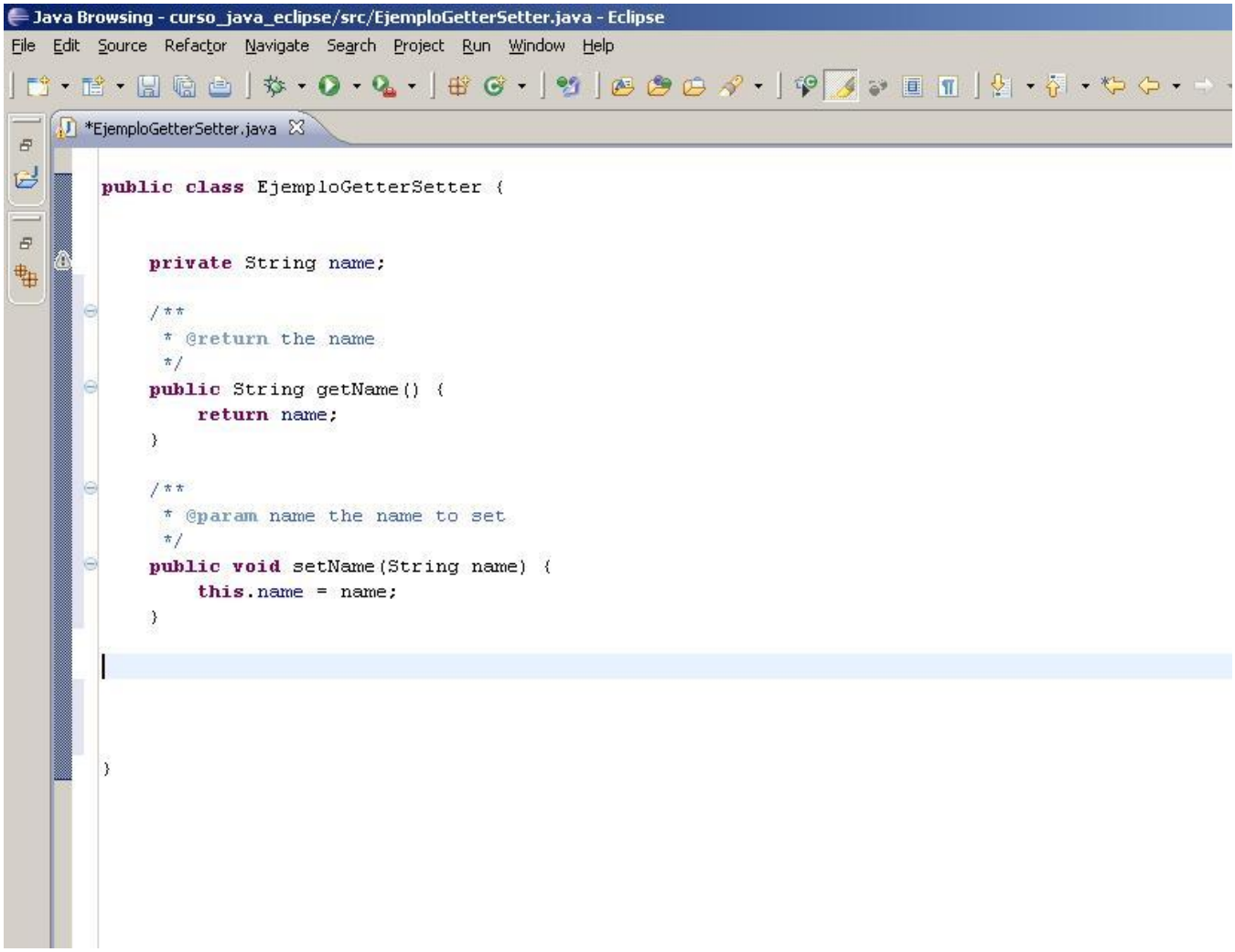
En este documento vamos a ver algunas utilidades que nos ofrece Eclipse y que se utilizan con mucha frecuencia.

1. Eclipse puede generar automáticamente los getter y los setter de una clase. Estos métodos se utilizan para leer y escribir los atributos privados de un objeto. De esta forma conseguimos uno de los principios básicos de la Programación Orientada a Objetos, que es el encapsulamiento: los atributos se declaran privados y desde fuera del objeto sólo se puede acceder a ellos a través de los métodos habilitados para ello, que sí son públicos.

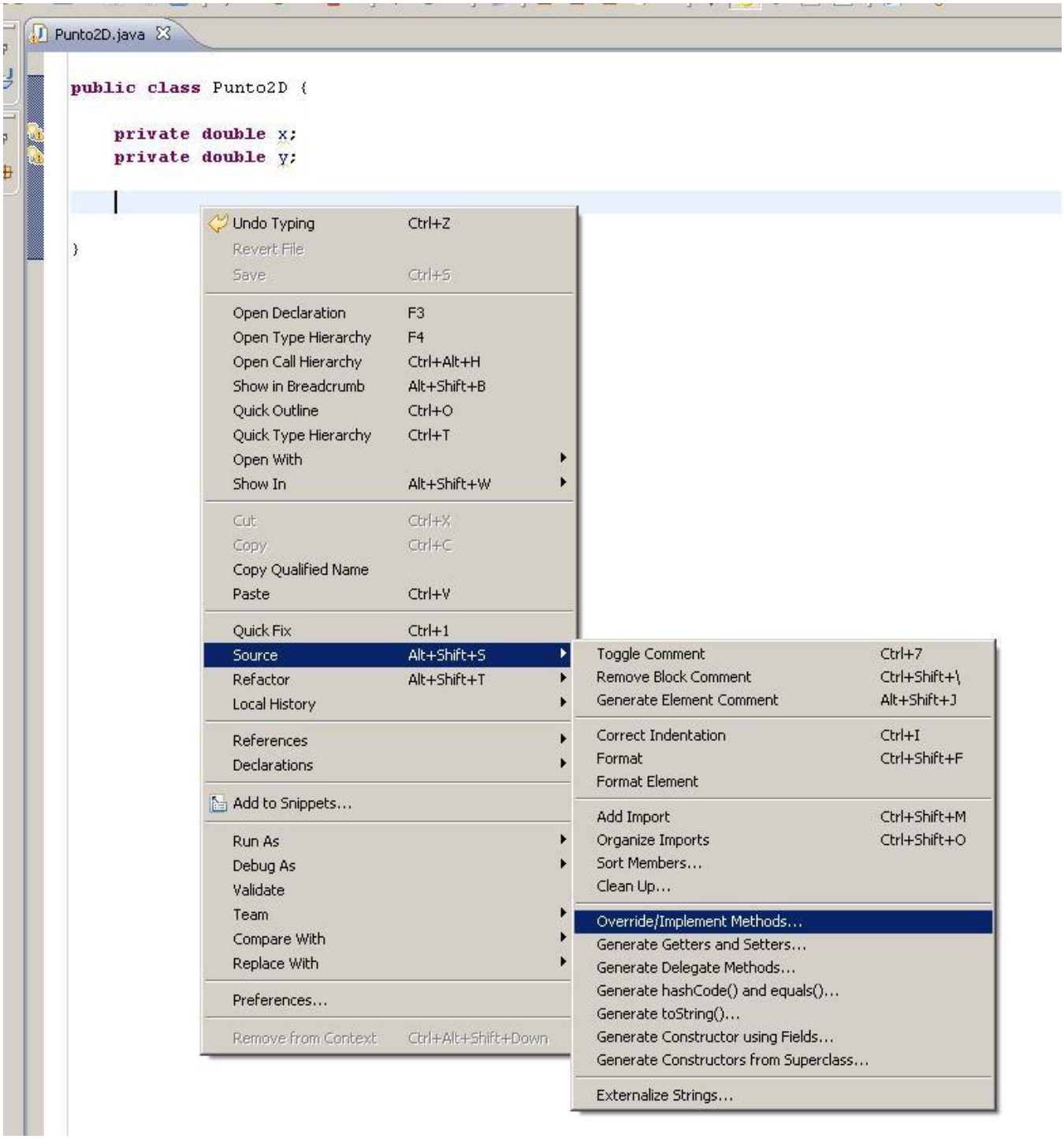
   Veamos cómo nos ayuda Eclipse con esto. Para ello, primero tenéis que crear la variable o variables para las que queréis crear los métodos, y después le dais al botón derecho del ratón -> Source -> Generate Getters and Setters

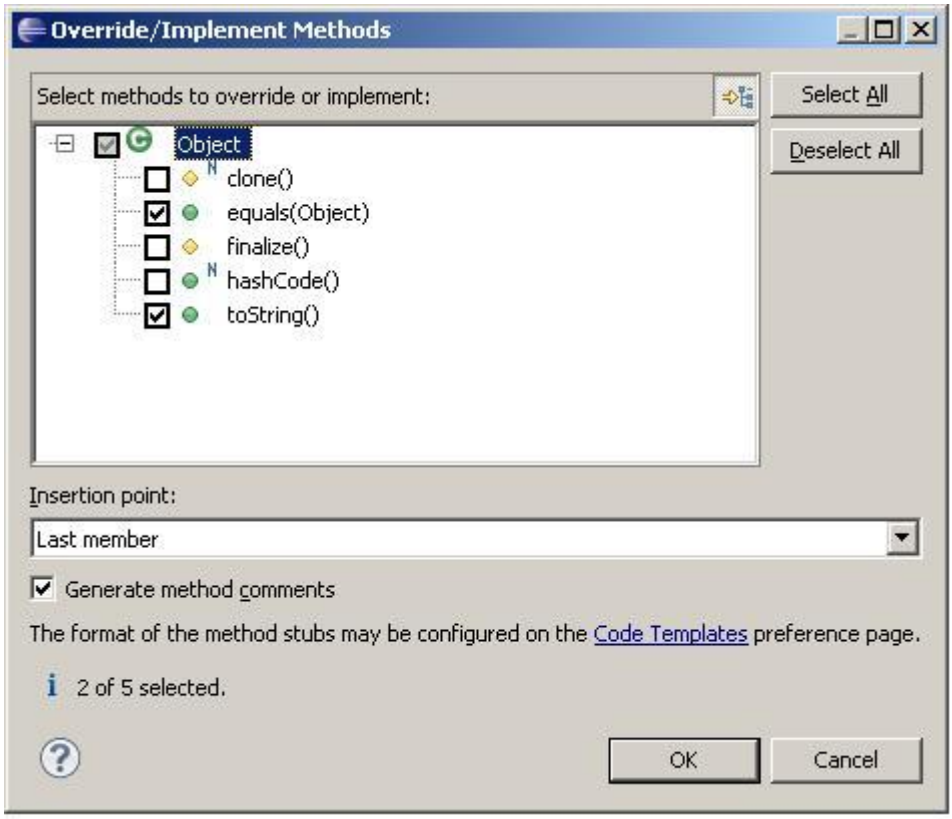


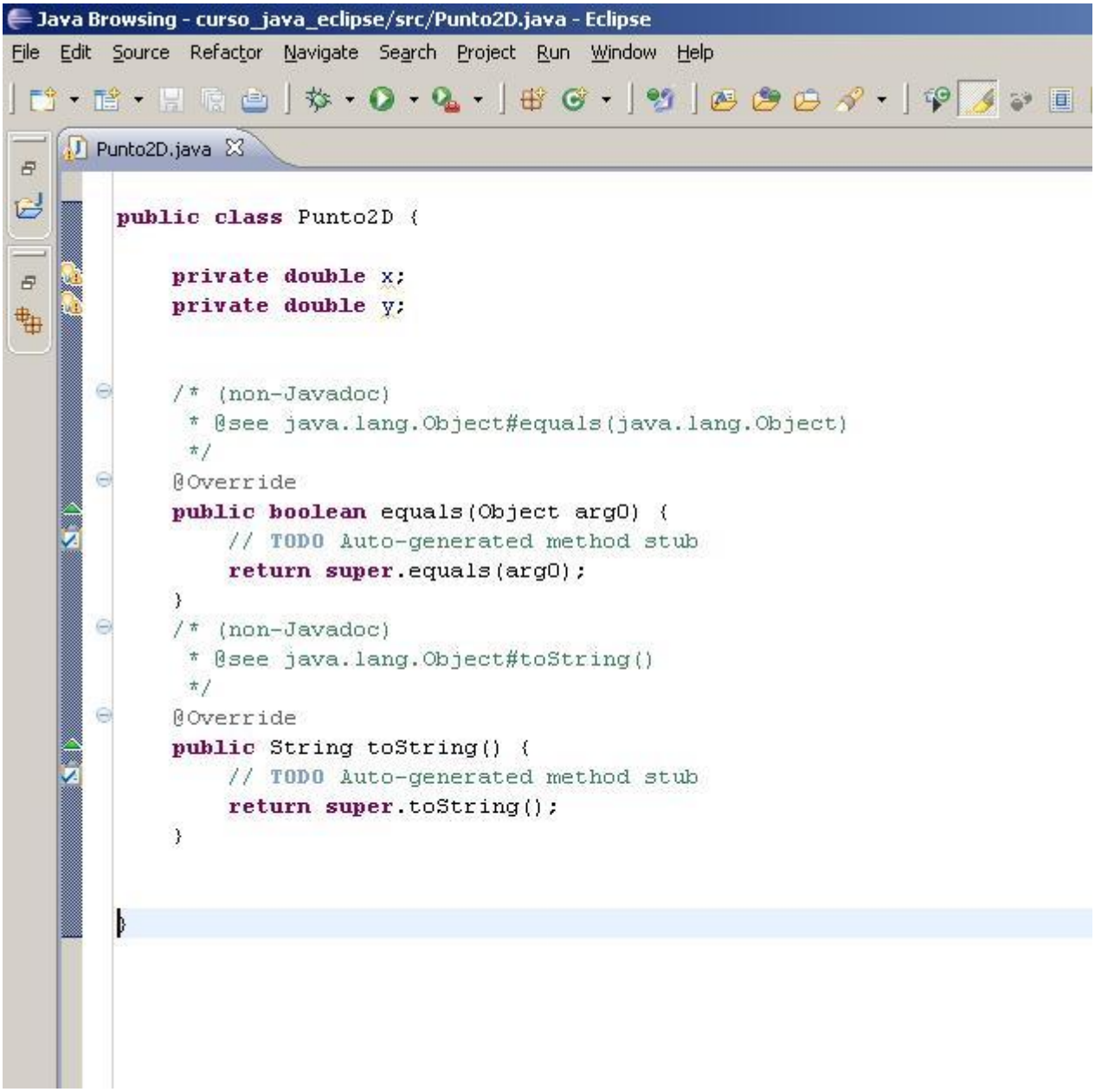

Generate Getters and Setters
Select getters and setters to create:
name
getName()
setName(String)
Select All
Deselect All
Select Getters
Select Setters
Allow setters for final fields (remove 'final' modifier from fields if necessary)
Insertion point:
Last member
Sort by:
Fields in getter/setter pairs
Access modifier
public
protected
default
private
final
synchronized
Generate method comments
The format of the getters/setters may be configured on the Code Templates preference page.
2 of 2 selected.
OK
Cancel


Java Browsing - curso_java_eclipse/src/EjemploGetterSetter.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
*EjemploGetterSetter.java
public class EjemploGetterSetter {
private String name;
/**
* @return the name
*/
public String getName() {
return name;
}
/**
* @param name the name to set
*/
public void setName(String name) {
this.name = name;
}
}

2. También podemos generar automáticamente el cuerpo de métodos heredados que queremos sobrescribir, como por ejemplo el caso de los métodos equals() y toString(). Para ello, pulsamos botón derecho -> Source -> Override/Implement methods.

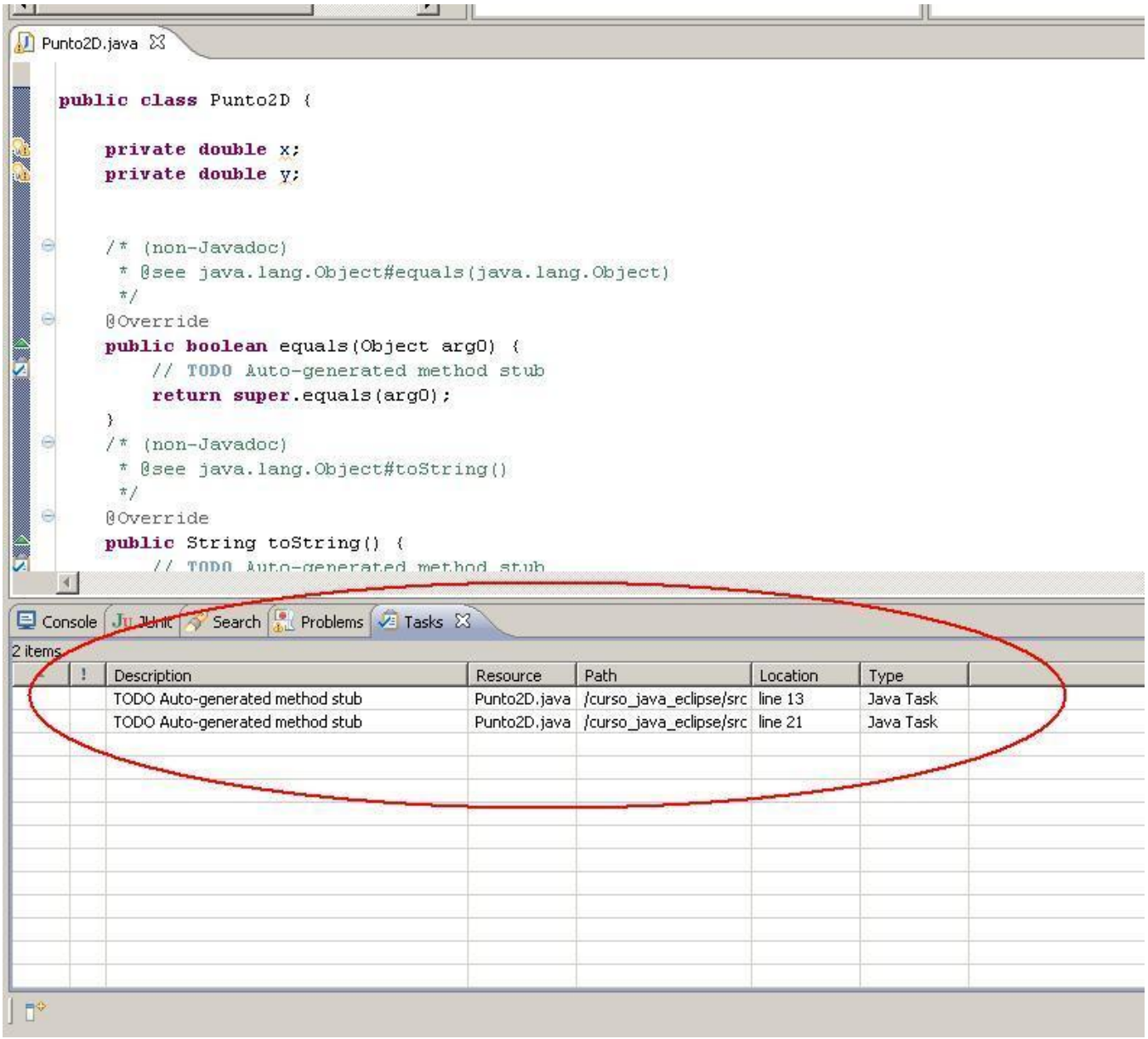



Override/Implement Methods
Select methods to override or implement:
Select All
Deselect All
Object
clone()
equals(Object)
finalize()
hashCode()
toString()
Insertion point:
Last member
Generate method comments
The format of the method stubs may be configured on the Code Templates preference page.
2 of 5 selected.
OK
Cancel



Lógicamente Eclipse sólo escribe el esqueleto de los métodos, y después nosotros debemos rellenar el contenido, ya que sólo nosotros podemos determinar cuándo consideramos que dos objetos de una clase son iguales o cómo convertirlo a String. Eclipse nos rellena estos métodos con una implementación por defecto pero es sólo para que compile, deberemos borrar el interior del método y escribir nuestro propio código.

3. Cuando generamos código automáticamente, como en el ejemplo anterior, Eclipse va poniendo unos comentarios "TODO" (del inglés "To do" = "Por hacer"). Son marcas de tareas pendientes de hacer. Para ver la lista de tareas pendientes de hacer: menú Window -> Show view -> Tasks.

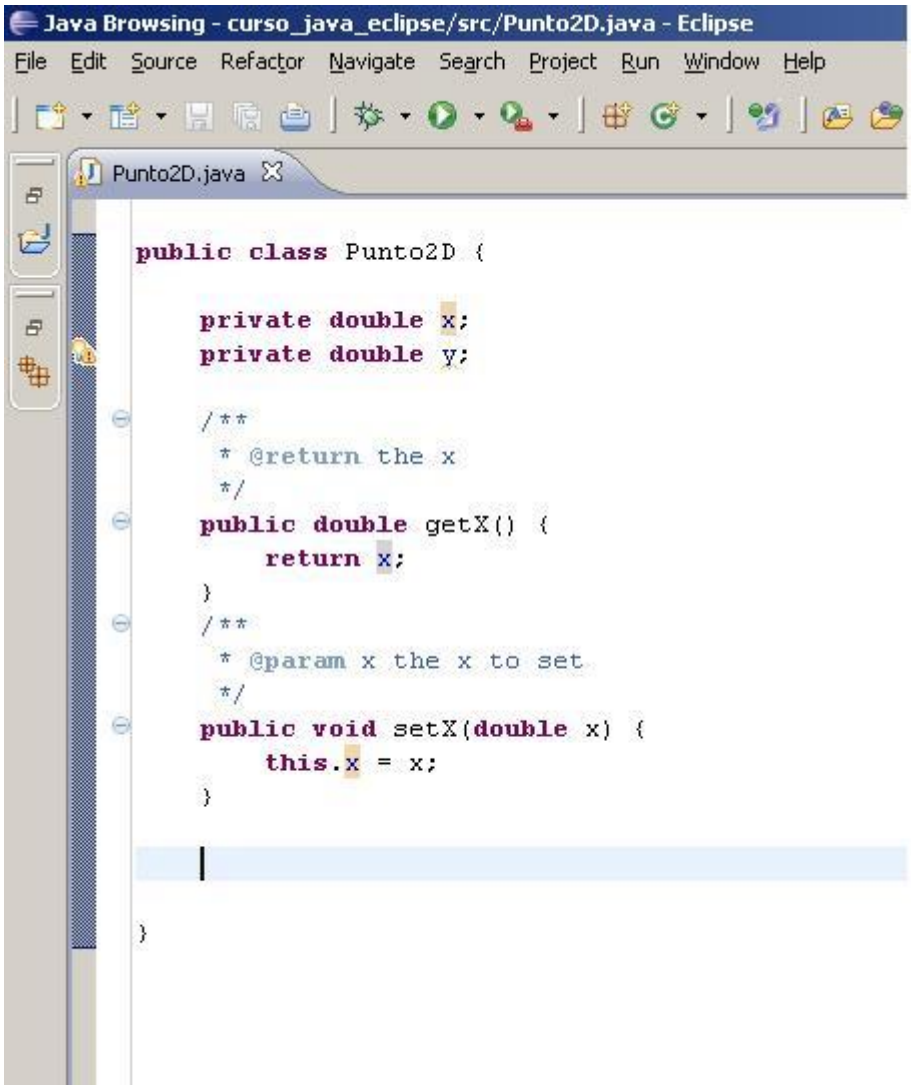


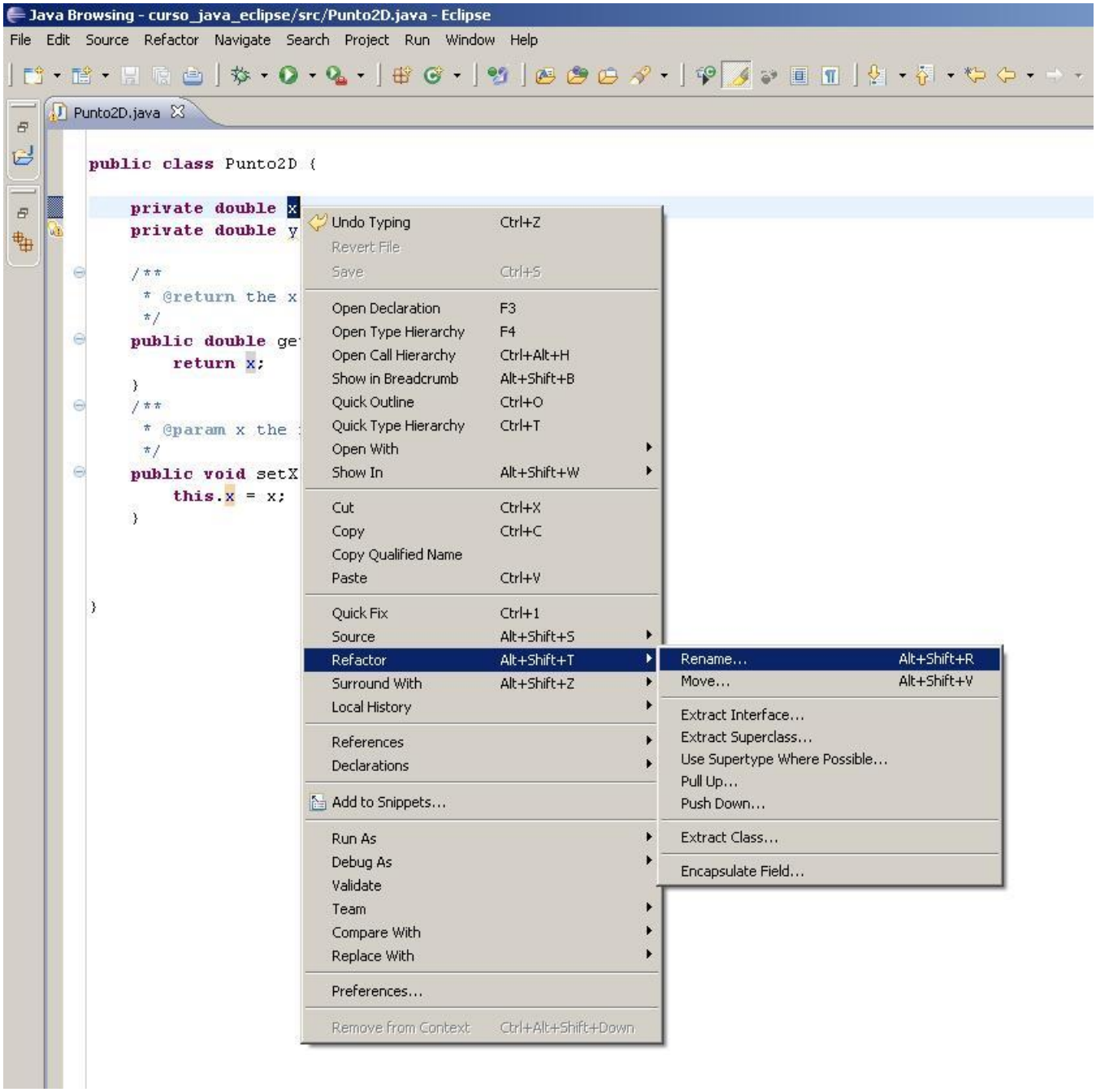
Haciendo doble click en un TODO nos abrirá el fichero correspondiente en la línea indicada.

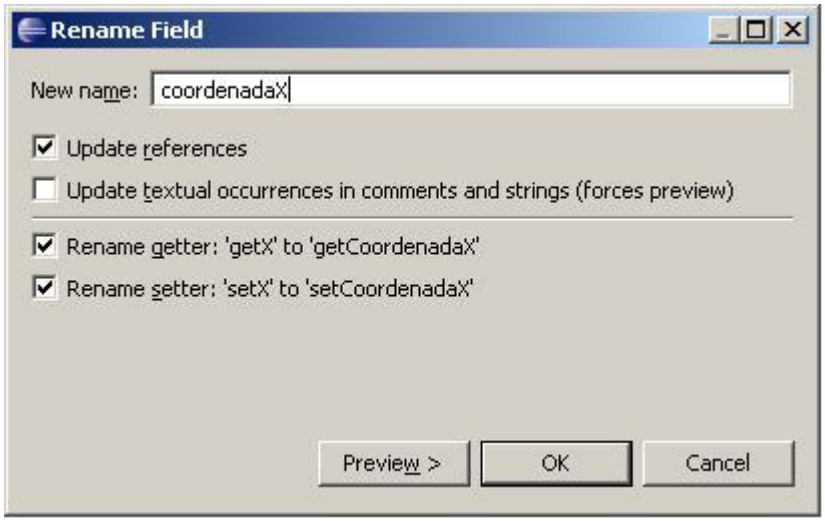
Una vez hecha la tarea hay que borrar el comentario TODO para que no aparezca en la lista. También podemos ir poniendo nuestros propios TODO manuales para que no se nos olvide nada.

Esto realmente es muy útil porque a medida que los proyectos aumentan en complejidad la generación automática de código se utiliza mucho, y de esta forma nunca olvidamos nada.

4. Cuando se desea renombrar una clase, un método o una variable, lo que se hace es un "Refactor". Situándonos con el cursor encima de la variable/método/clase, pulsamos botón derecho -> Refactor -> Rename. De esta manera se actualiza el nombre **en todos los lugares** en los que se hace referencia al elemento renombrado. Incluso Eclipse se encarga de renombrar los getter y los setter si se lo pedimos.




Java Browsing - curso_java_eclipse/src/Punto2D.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Punto2D.java
public class Punto2D {
private double x
private double y
/**
* @return the x
*/
public double ge
return x;
}
/**
* @param x the
*/
public void setX
this.x = x;
}
}
Undo Typing Ctrl+Z
Revert File
Save Ctrl+S
Open Declaration F3
Open Type Hierarchy F4
Open Call Hierarchy Ctrl+Alt+H
Show in Breadcrumb Alt+Shift+B
Quick Outline Ctrl+O
Quick Type Hierarchy Ctrl+T
Open With
Show In Alt+Shift+W
Cut Ctrl+X
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Quick Fix Ctrl+1
Source Alt+Shift+S
Refactor Alt+Shift+T
Surround With Alt+Shift+Z
Local History
References
Declarations
Add to Snippets...
Run As
Debug As
Validate
Team
Compare With
Replace With
Preferences...
Remove from Context Ctrl+Alt+Shift+Down
Rename... Alt+Shift+R
Move... Alt+Shift+V
Extract Interface...
Extract Superclass...
Use Supertype Where Possible...
Pull Up...
Push Down...
Extract Class...
Encapsulate Field...



Este es el resultado: