Entornos de Desarrollo

Tema 4

Optimización y documentación

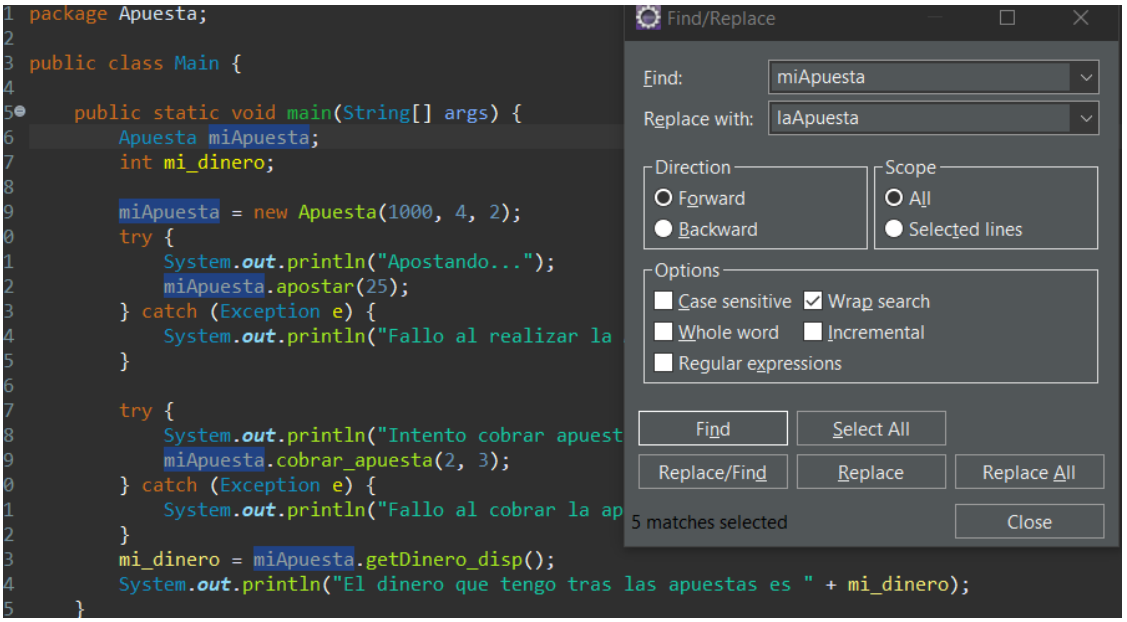
Índice

1. Refactorización 3
    1.1. Cambiar nombre de variable 3
    1.2. Crear método 4
    1.3. Encapsular atributos 5
    1.4. Añadir nuevo parámetro 5
2. Analizador de código 6
    2.1. Descarga e instala el plugin PMD 6
    2.2. Ejecuta el analizador de código PMD 7
    2.3. Añadir reglas 8
3. Javadoc 10
    3.1. Inserta comentarios Javadoc 10
    3.2. Genera documentación Javadoc 12

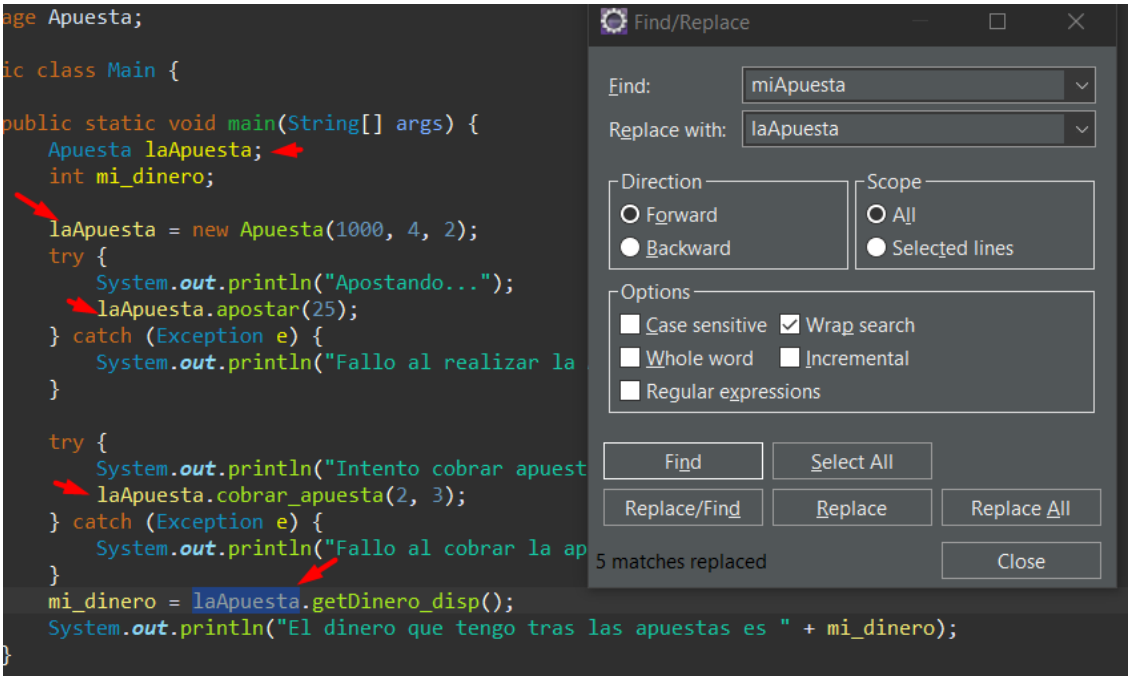
1. Refactorización

1.1. Cambia el nombre de la variable “miApuesta” por "laApuesta”.

Para poder cambiar el nombre de la variable en todo el Main sin tener que buscar uno a uno los lugares donde se usa dicha variable, haremos “ctrl+f”. En la ventana emergente, en el primer recuadro escribimos la variable que queremos buscar y en el de abajo, la variable que queremos poner:



De este modo, reemplazamos todos sus usos de una sola vez:



1.2. Introduce el método operativa Apuesta, que englobe las sentencias de la clase Main que operan con el objeto laApuesta.

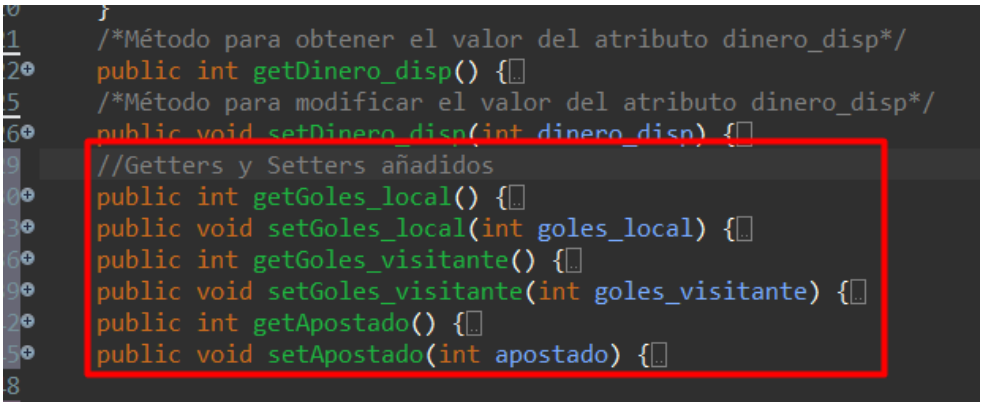
Para poder usar todo lo implementado en el Main en un método a parte, tendremos que cambiar algunos aspectos, como cuando llama a la variable “laApuesta” o el uso de la otra variable “mi_dinero” que puede sustituirse por una llamada al método “getter” del atributo “dinero_disp” de la clase apuesta, ya que esta variable no se usa para nada más:

```
public static void main(String[] args) {
    Apuesta laApuesta;
    //int mi_dinero;

    laApuesta = new Apuesta(1000, 4, 2);
    /*Dejamos comentado el código previo
     *
     * try {
        System.out.println("Apostando...");
        laApuesta.apostar(25);
    } catch (Exception e) {
        System.out.println("Fallo al realizar la Apuesta");
    }

    try {
        System.out.println("Intento cobrar apuesta segun el resultado del partido");
        laApuesta.cobrar_apuesta(2, 3);
    } catch (Exception e) {
        System.out.println("Fallo al cobrar la apuesta");
    }
    mi_dinero = laApuesta.getDinero_disp();
    System.out.println("El dinero que tengo tras las apuestas es " + mi_dinero);*/

    //usamos el método creado a raíz de lo usado en Main
    laApuesta.operativaApuesta();
```

```
public void operativaApuesta() {
    try {
        System.out.println("Apostando...");
        this.apostar(25);
    } catch (Exception e) {
        System.out.println("Fallo al realizar la Apuesta");
    }

    try {
        System.out.println("Intento cobrar apuesta segun el resultado del partido");
        this.cobrar_apuesta(2, 3);
    } catch (Exception e) {
        System.out.println("Fallo al cobrar la apuesta");
    }
    System.out.println("El dinero que tengo tras las apuestas es " + this.getDinero_disp());
}
```

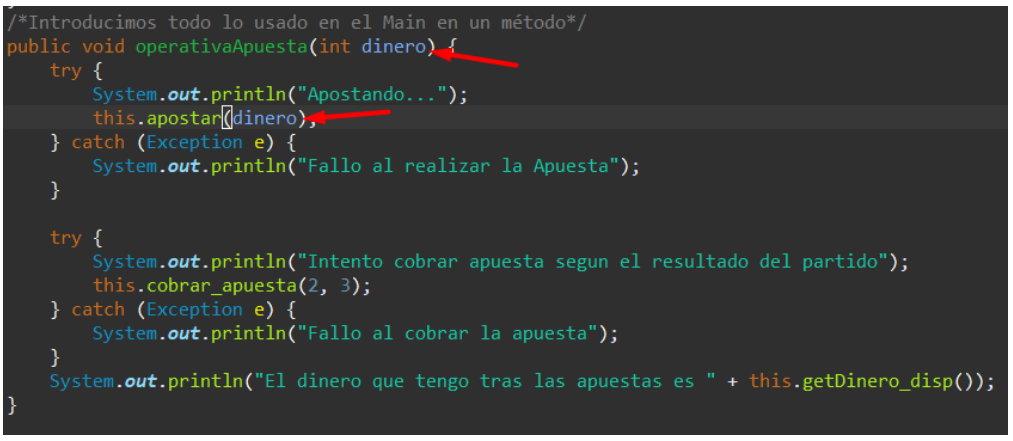
1.3. Encapsula todos los atributos de la clase Apuesta.

En la clase “Apuesta” encontramos cuatro atributos privados, por lo que no puede accederse a ellos desde fuera. Para que dichos atributos sean accesibles tenemos que definir tanto sus métodos getters como setters:



1.4. Añadir un nuevo parámetro al método operativa_Apuesta, de nombre dinero y de tipo int.

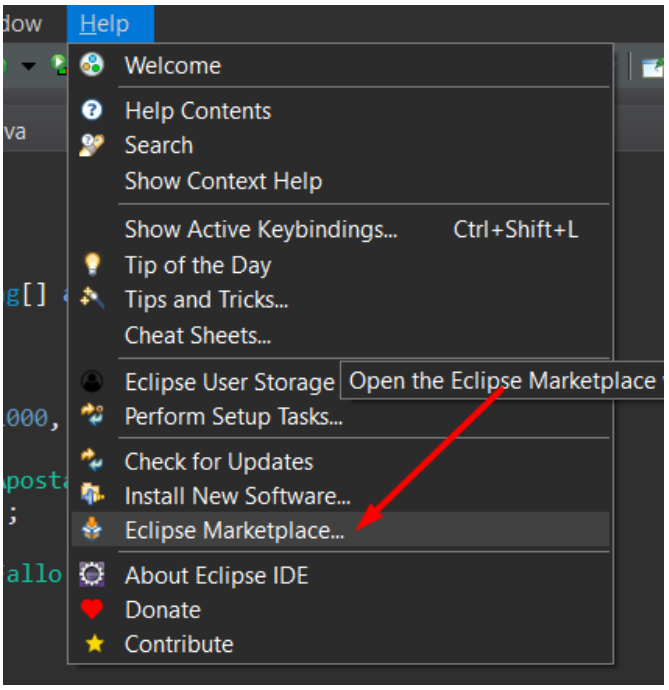
Modificamos el método creado previamente, de manera que, usando un parámetro que le pasamos al método, este pueda usarse dentro del mismo como parámetro de otro método:

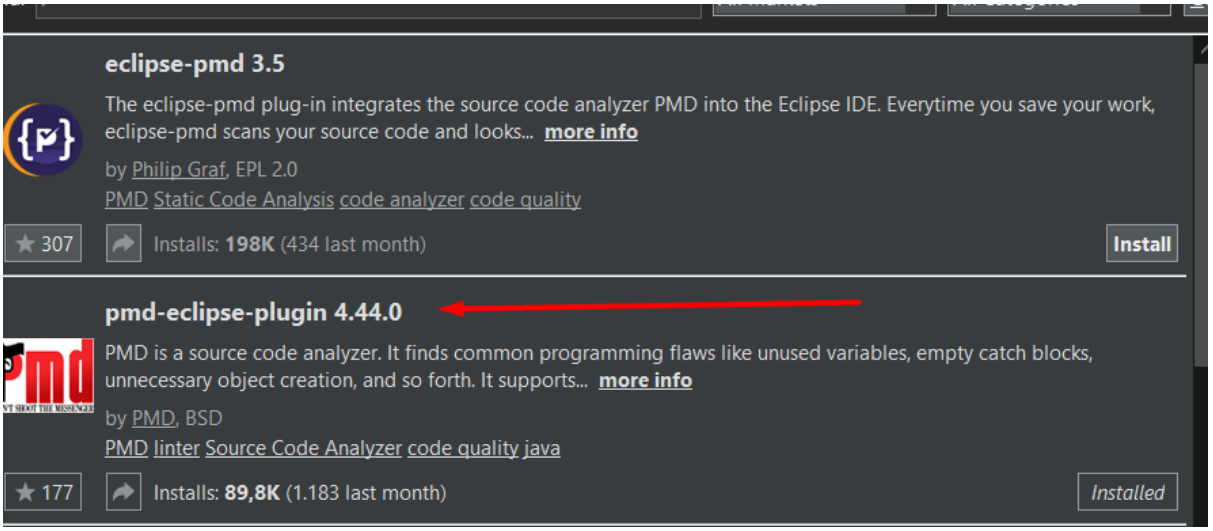


2. Analizador de código

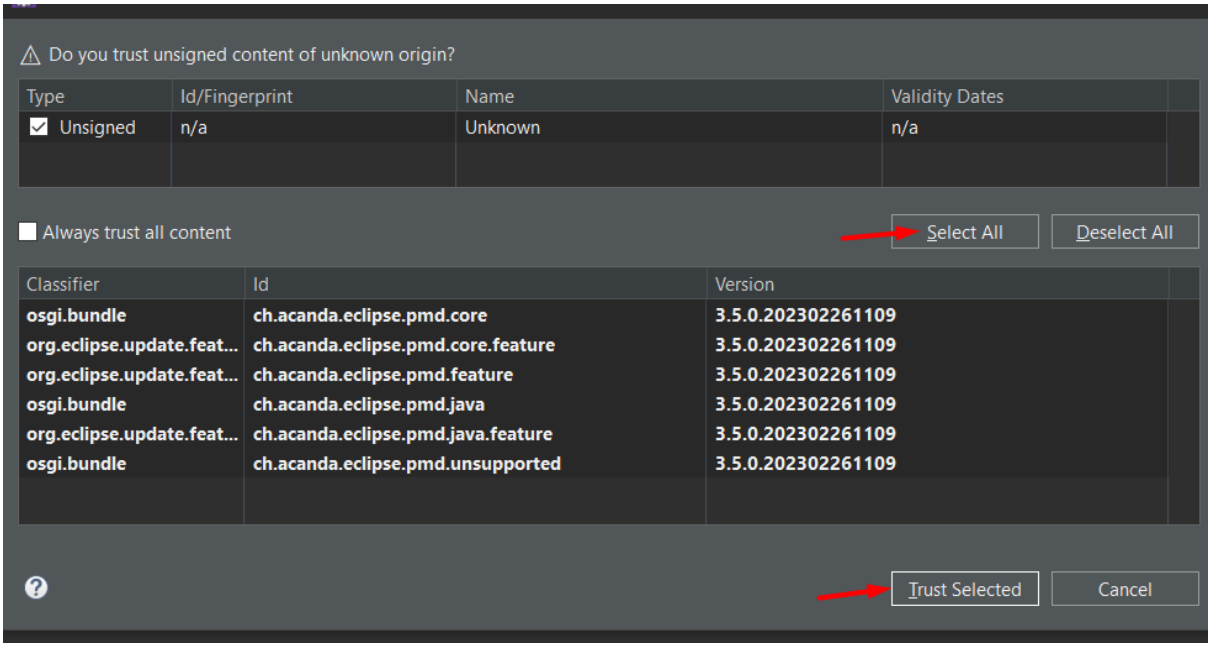
2.1. Descarga e instala el plugin PMD.

Para descargar e instalar dicho plugin, primero accedemos al MarketPlace de Eclipse y en la ventana emergente buscamos PMD. Elegimos la primera opción:

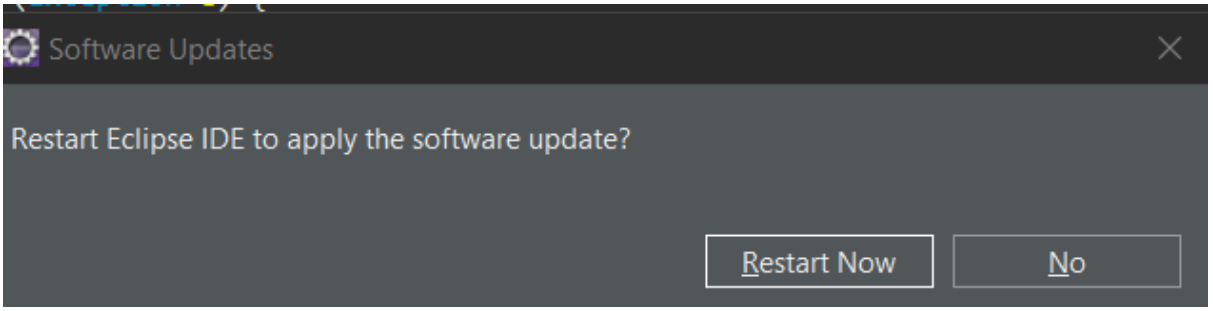




Tendremos que aceptar los términos y condiciones, en la otra ventana que surja marcaremos todo y damos a aceptar:

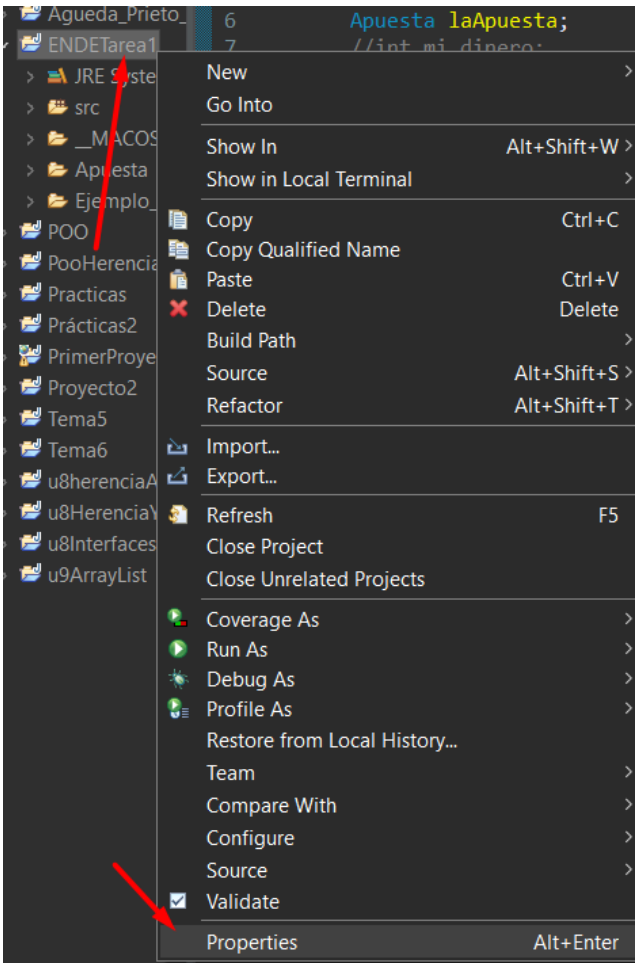


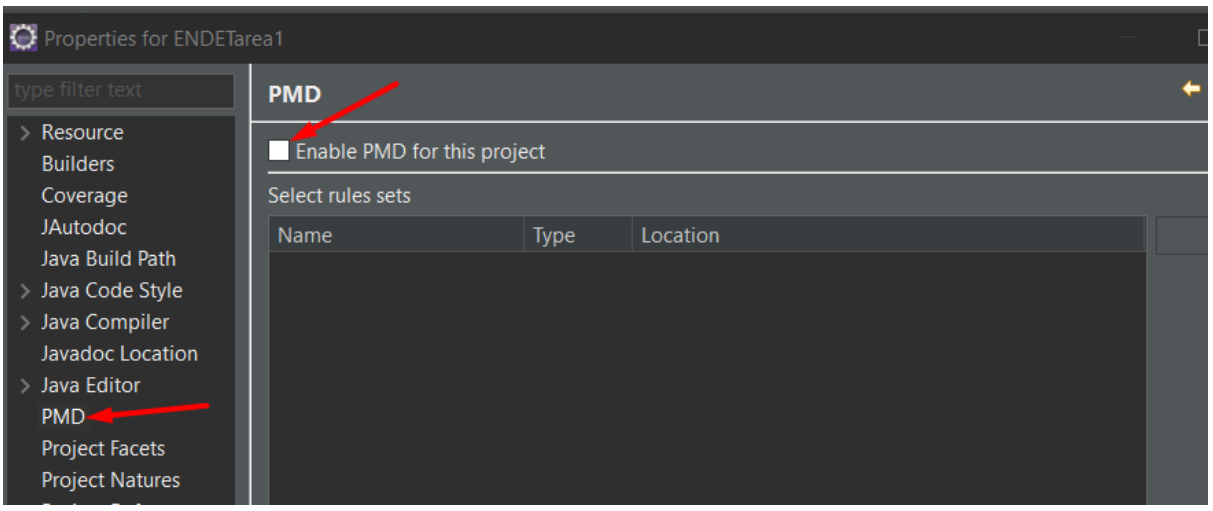
Para finalizar la instalación, tendremos que reiniciar Eclipse:



2.2. Ejecuta el analizador de código PMD.

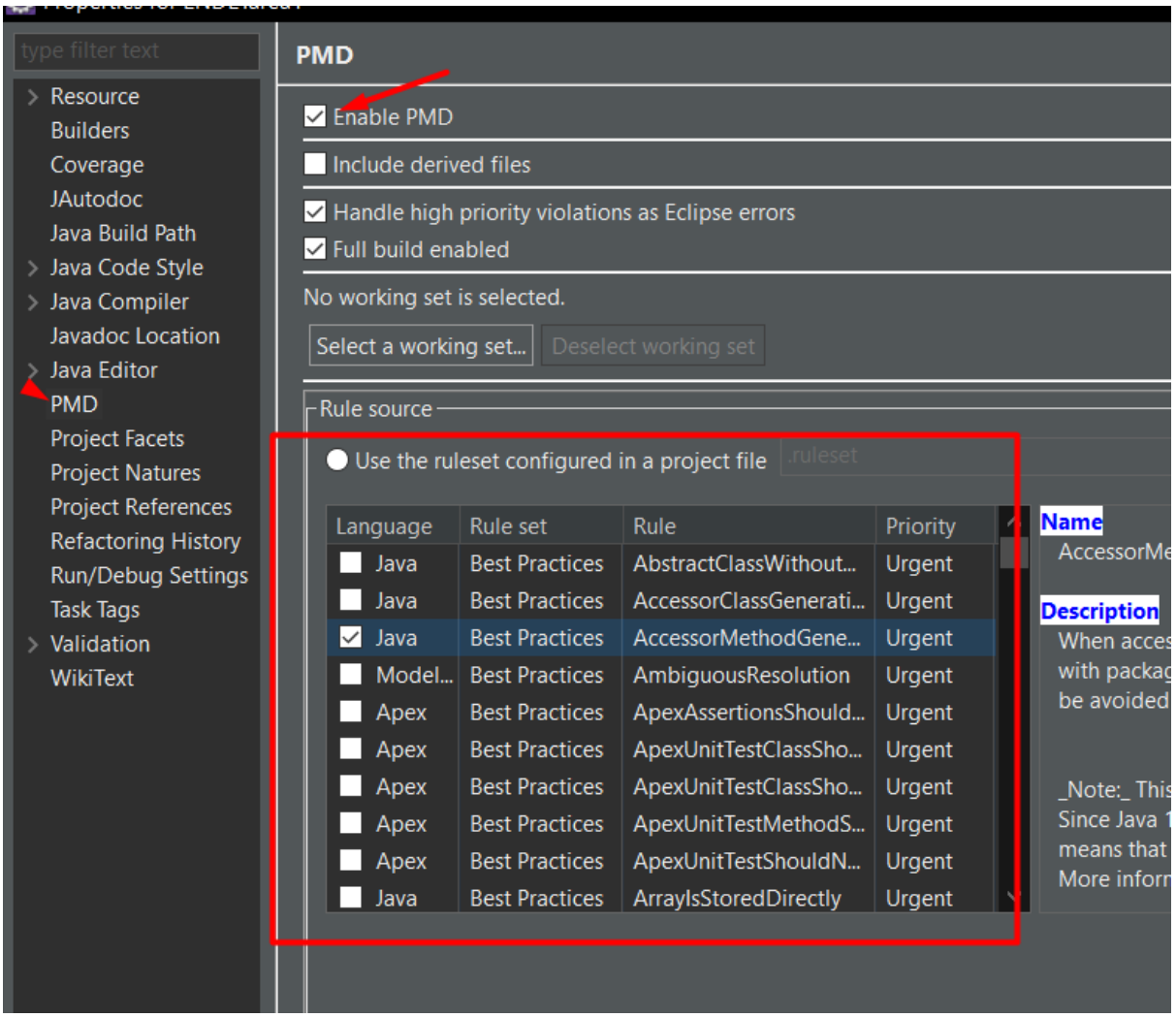
Tras haber instalado el plugin, para poder habilitarlo tenemos que seleccionar el proyecto sobre el que queremos ejecutarlo, dando clic derecho buscamos la opción "***Propiedades***" y, en la ventana emergente que aparece, buscar "***PMD***":



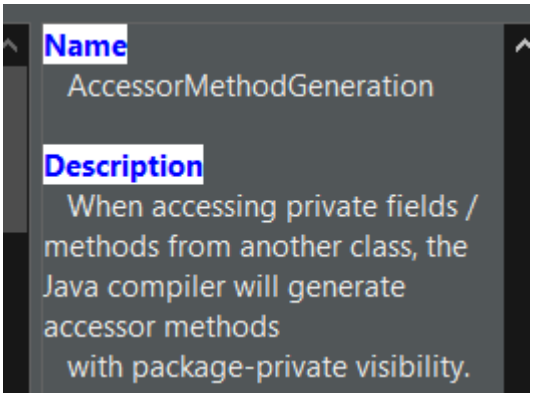


2.3. Añade tres reglas nuevas al analizador de PMD y comenta su utilidad.

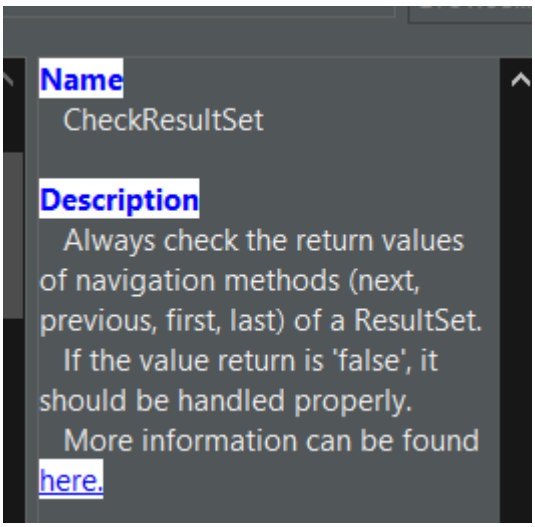
Desde “Project” accedemos a “properties”, seleccionamos PMD y en la ventana que abre, podemos encontrar un conjunto de reglas que podemos aplicar a nuestro proyecto:



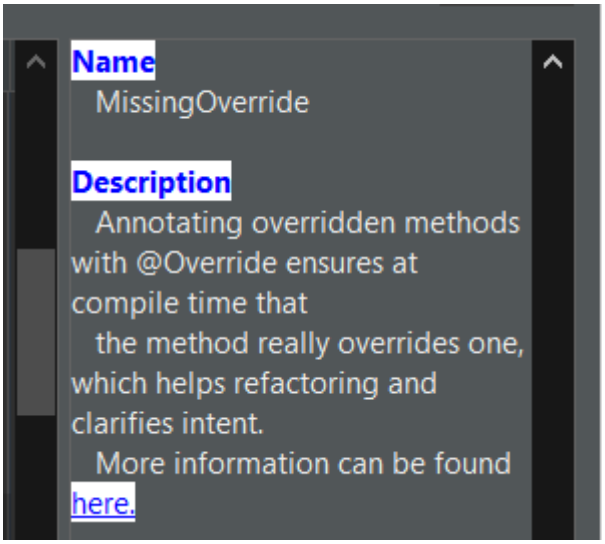
La primera que he seleccionado para que se aplique es, que cuando se intenten acceder a campos o métodos de otra clase, se generarán métodos para acceder a dichos elementos con un paquete de visibilidad privada:



La siguiente regla añadida se encarga de revisar “los returns” de los métodos:



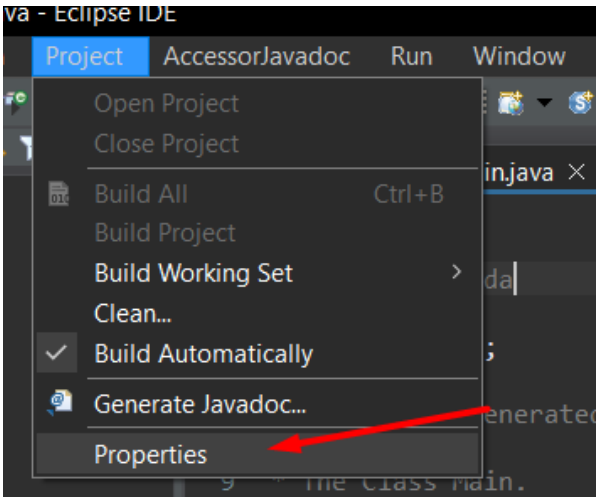
La última regla que he agregado se encarga de señalar los métodos “override”, comprobando que efectivamente hace referencia a otro método al compilar, de forma que ayuda a la refactorización:



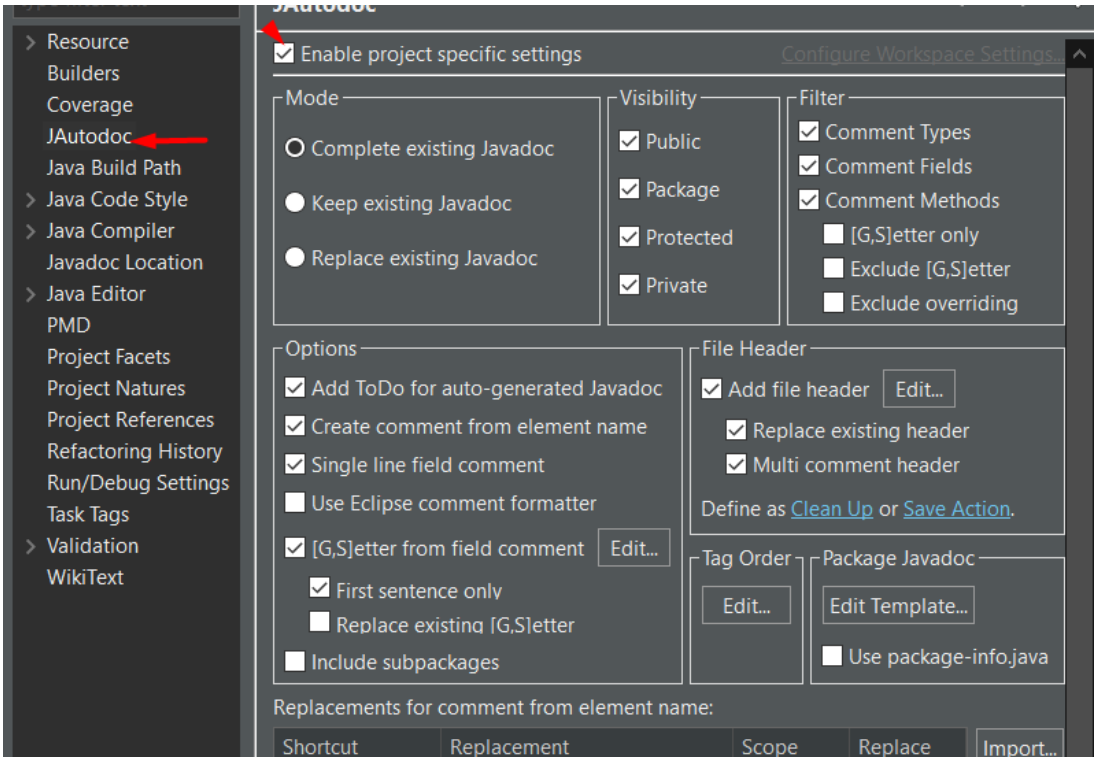
3. Javadoc

3.1. Inserta comentarios Javadoc en la clase Apuesta

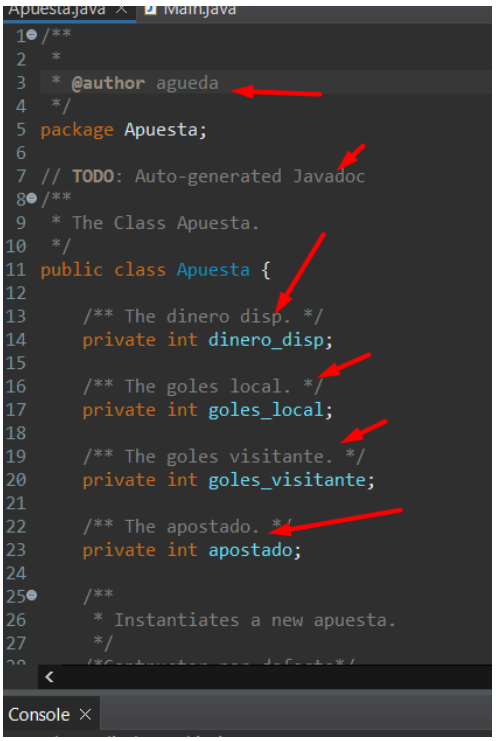
Para poder insertar comentarios, primero configuraremos el Javadoc en Eclipse, accediendo desde “Project” en “Properties”, buscando el apartado de “Javadoc”:



Nos aseguramos de que tenemos habilitada la opción que aparece en la parte superior y, después, simplemente tendremos que usar “control+alt+J”:

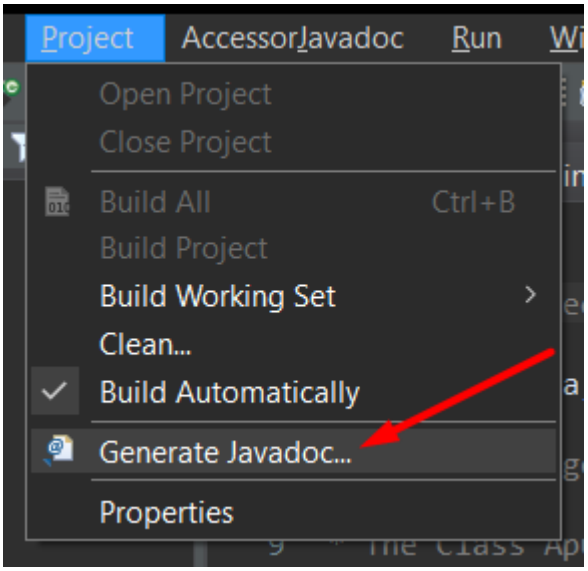


De esta forma, aparecerán comentarios en cada detalle del código:



3.2. Genera documentación Javadoc para todo el proyecto.

Para generar el documento Javadoc, accedemos a "Project", seleccionamos "Generate javadoc":



Y una vez seleccionado todo esto, se generará el conjunto de documentos relacionados con el proyecto, de modo que a tarea de documentación se simplifica y facilita: