



### Trabajo 3. Mantenibilidad y refactorización

EVOLUCIÓN Y MANTENIMIENTO DEL  
SOFTWARE ©2021

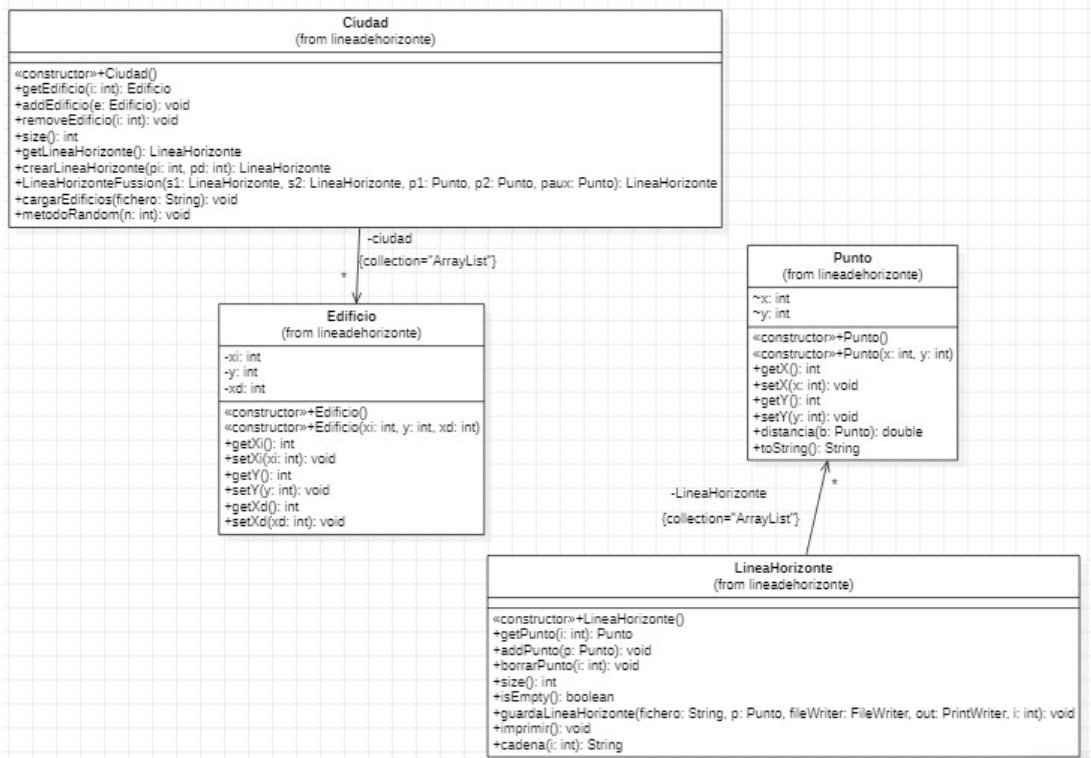
Héctor Cantero Álvarez bl0523

## Contenido

Análisis inicial .....	2
Diagrama de clases inicial .....	2
Diagrama de secuencia del método “crearLineaHorizonte” .....	3
Análisis de “bad smells” con Better Code Hub.....	4
Unidades de código extensas .....	5
Unidades de código complejas.....	5
Código repetido.....	6
Interfaces extensas.....	6
Conceptos fuera de módulos .....	7
Componentes arquitectónicos desbalanceados .....	7
Análisis de “bad smells” con PMD .....	8
Copy/Paste .....	8
Violaciones .....	8
Issues creados .....	9
Resultados de la refactorización .....	10
Nuevo diagrama de clases.....	10
Issues resueltos .....	10
Nivel final de conformidad alcanzado .....	11

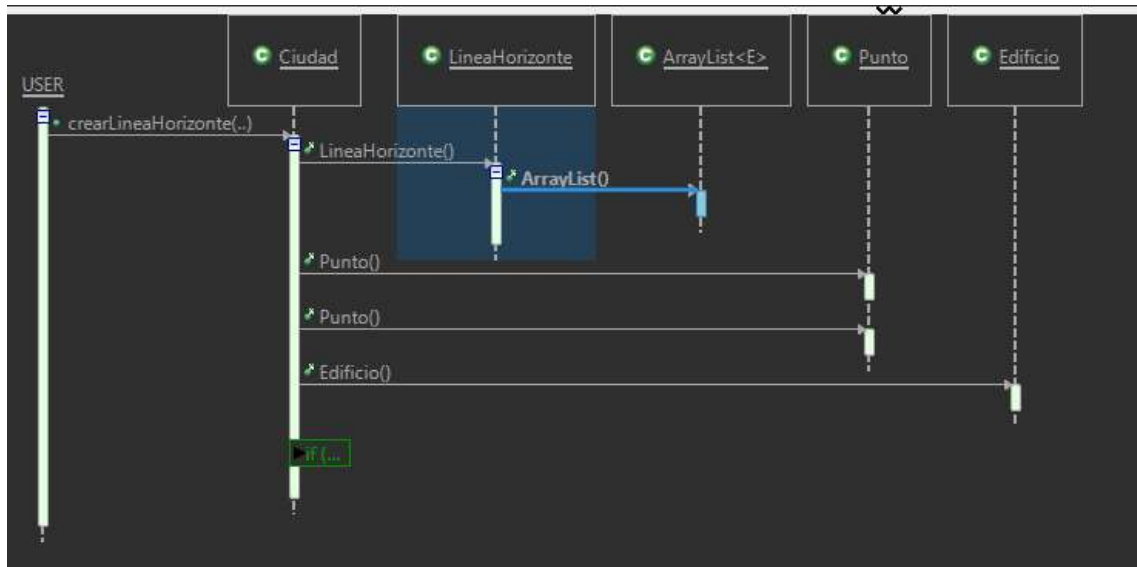
## Análisis inicial

### Diagrama de clases inicial



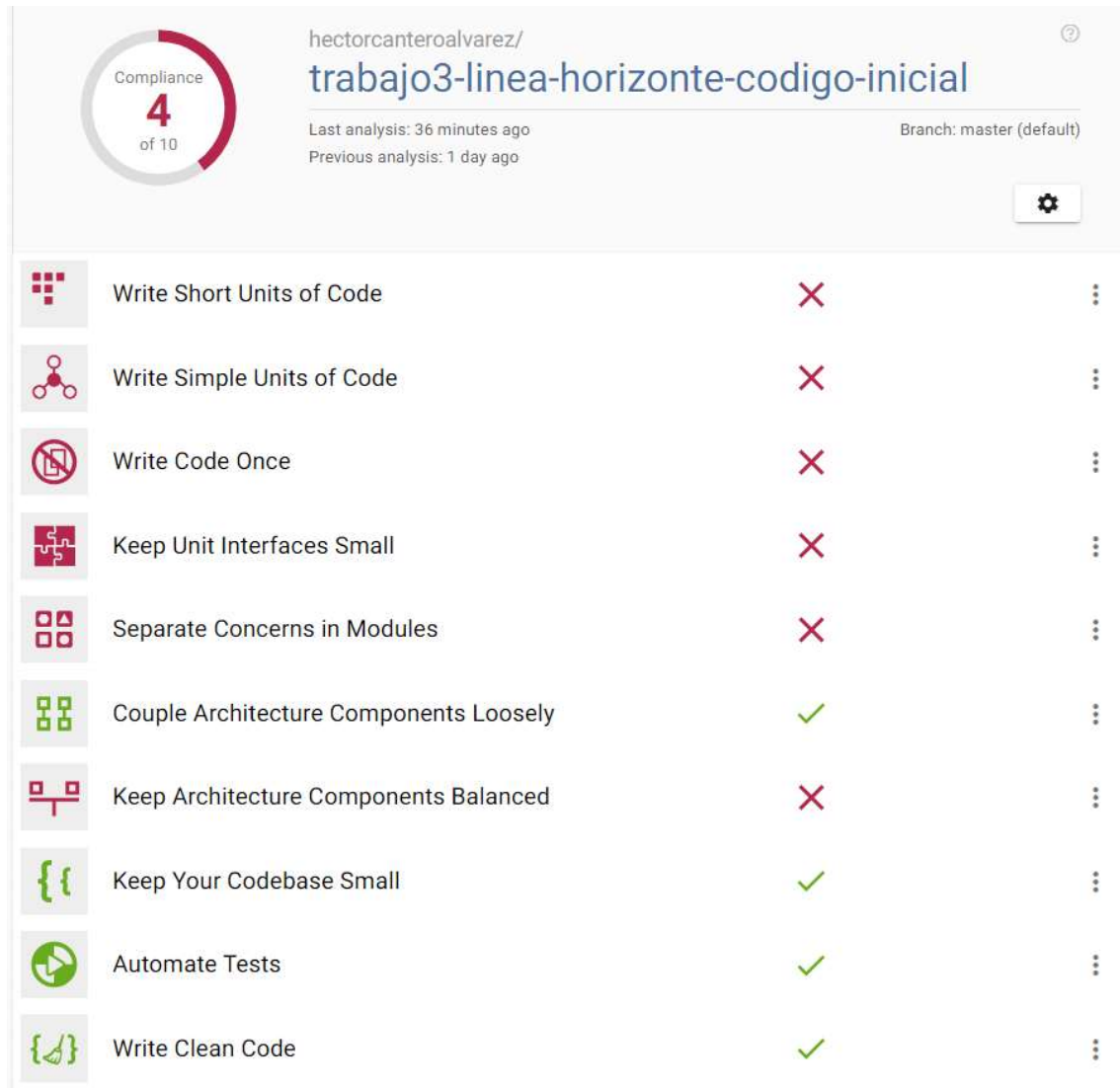
El diagrama de clases inicial muestra la relación entre las clases Ciudad-Edificio y Punto-LineaHorizonte. Una Ciudad está compuesta de edificios y una LineaHorizonte por puntos, lo cual concuerda con los atributos de las clases vistos en el código, sin embargo, como veremos a continuación las clases están más relacionadas.

### Diagrama de secuencia del método “crearLineaHorizonte”



Podemos ver que el método es bastante complejo, ya que hace uso de todas las clases, a pesar de que una *Ciudad* solo tiene *Edificios*, para generar la *LineaHorizonte* de una Ciudad es necesario usar esta clase, que a su vez trata con la clase *Punto*.

## Análisis de “bad smells” con Better Code Hub



El nivel de conformidad inicial es 4. A continuación se muestra en detalle cada “bad smell”.

## Unidades de código extensas

The screenshot shows the 'Write Short Units of Code' window. It features a 'Refactoring candidates' table with columns for 'Unit' and 'Lines of Code'. A legend at the bottom indicates color-coded ranges for lines of code: green for 'at most 15 lines of code', yellow for 'more than 15 lines of code', orange for 'more than 30 lines of code', and red for 'more than 60 lines of code'. The 'Ciudad.LineaHorizonteFussion' method is highlighted in red, indicating it exceeds 60 lines.

Unit	Lines of Code
<input type="checkbox"/> Ciudad.LineaHorizonteFussion(LineaHorizonte, LineaHorizonte, Punto, Punto, Punto)	79
<input type="checkbox"/> Ciudad.crearLineaHorizonte(int, int)	25
<input type="checkbox"/> LineaHorizonte.guardaLineaHorizonte(String, Punto, FileWriter, PrintWriter, int)	18
<input type="checkbox"/> Ciudad.cargarEdificios(String)	16

Cómo podemos ver, la longitud de los métodos es demasiado grande, en especial en la clase *Ciudad* donde el método *LineaHorizonteFussion* tiene más de 60 líneas. Es necesario refactorizar estos métodos descomponiéndolos en unidades de código más cortas.

## Unidades de código complejas

The screenshot shows the 'Write Simple Units of Code' window. It features a 'Refactoring candidates' table with columns for 'Unit', 'Lines of Code', and 'Branch points'. A legend at the bottom indicates color-coded ranges for McCabe complexity: green for 'McCabe of at most 5', yellow for 'McCabe above 5', orange for 'McCabe above 10', and red for 'McCabe above 25'. The 'Ciudad.LineaHorizonteFussion' method is highlighted in red, indicating its complexity is above 25.

Unit	Lines of Code	Branch points
<input type="checkbox"/> Ciudad.LineaHorizonteFussion(LineaHorizonte, LineaHorizonte, Punto, Punt...	79	14

La complejidad ciclomática del método *LineaHorizonteFussion* es demasiado elevada y esto daría lugar a un número de pruebas mínimo muy elevado para cubrir todos los caminos de ejecución.

## Código repetido

The 'Write Code Once' window displays a list of refactoring candidates under the heading 'Refactoring candidates'. A sidebar on the left is labeled 'GUIDELINE EXPLANATION'. At the top right, there is a 'Show snoozed' button and a help icon. The candidates are listed in a table with columns for a checkbox, a description, and 'Lines of Code'.

		Lines of Code
<input checked="" type="checkbox"/>	Duplicate	
<input type="checkbox"/>	7 lines occurring 2 times in Ciudad.java	7
<input type="checkbox"/>	6 lines occurring 2 times in 2 files: LineaHorizonte.java, Punto.java	6

Below the table is a progress bar with a green segment and a small red segment, followed by a red 'X' icon. A legend at the bottom indicates that green represents 'non-duplicated code' and red represents 'duplicated code'.

Tenemos código idéntico en varias funciones, por lo que es posible refactorizar el código e implementar un método que satisfaga a ambas funciones evitando repeticiones.

## Interfaces extensas

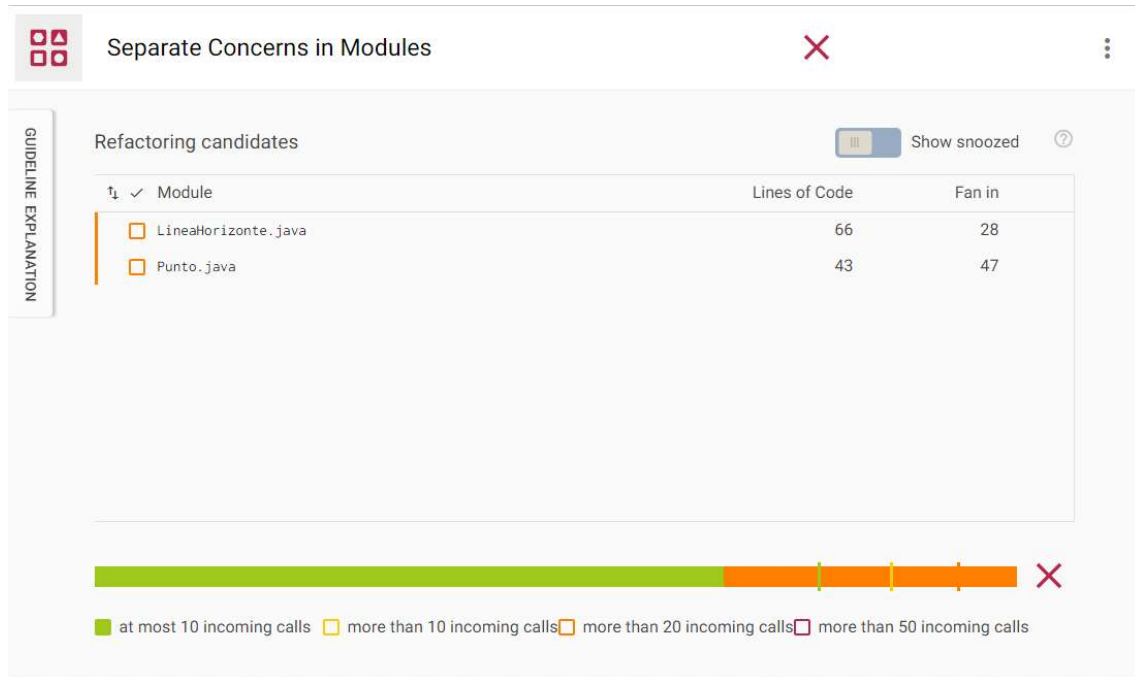
The 'Keep Unit Interfaces Small' window displays a list of refactoring candidates under the heading 'Refactoring candidates'. A sidebar on the left is labeled 'GUIDELINE EXPLANATION'. At the top right, there is a 'Show snoozed' button and a help icon. The candidates are listed in a table with columns for a checkbox, 'Unit', 'Lines of Code', and 'Parameters'.

	Unit	Lines of Code	Parameters
<input type="checkbox"/>	Ciudad.LineaHorizonteFussion(LineaHorizonte, LineaHorizonte, Punto, Punt...	79	5
<input type="checkbox"/>	LineaHorizonte.guardaLineaHorizonte(String, Punto, FileWriter, PrintWrit...	18	5
<input type="checkbox"/>	Edificio.Edificio(int, int, int)	5	3

Below the table is a progress bar with segments of green, yellow, orange, and red, followed by a red 'X' icon. A legend at the bottom indicates the parameter counts: green for 'at most 2 parameters', yellow for 'more than 2 parameters', orange for 'more than 4 parameters', and red for 'more than 6 parameters'.

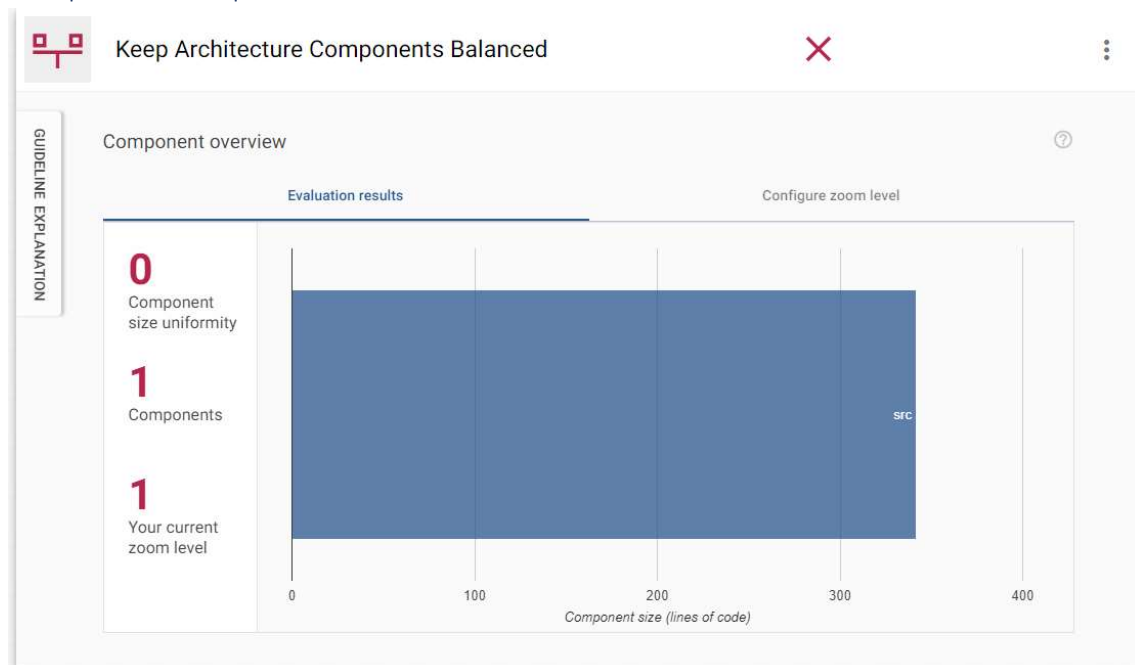
El número de parámetros de entrada es bastante elevado, hay que comprobar si realmente es necesario pasar todos esos parámetros y en caso de ser así, implementar una manera más eficaz de hacerlo.

## Conceptos fuera de módulos



Las llamadas a métodos de otras clases son muy elevadas, por lo que es posible que sea necesario pasar las responsabilidades de las funcionalidades que se esté tratando directamente a dichas clases.

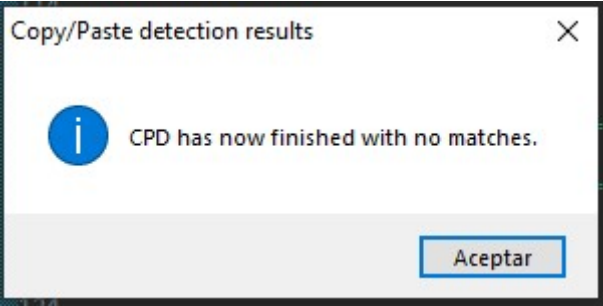
## Componentes arquitectónicos desbalanceados



El tamaño relativo de los componentes está desequilibrado, algunas clases tienen un tamaño mucho más extenso que otras.

Análisis de “bad smells” con PMD

Copy/Paste



El análisis de Copia/Pega no encuentra ningún bloque de código sospechoso

Violaciones

Element	# Violations	# Violations/KLOC	# Violations/Method
> etsisi.ems2020.trabajo3.lineadehorizonte	74	N/A	N/A
▼ etsisi.ems2020.trabajo3.lineadehorizonte	241	923.4	4.30
> Ciudad.java	134	827.2	4.79
> Edificio.java	22	956.5	2.75
> LineaHorizonte.java	48	1230.8	4.00
> Main.java	14	1076.9	14.00
> Punto.java	23	958.3	3.29

El análisis de violaciones en cambio, si que muestra un número considerable de ellas, por lo que habrá que revisarlas y solucionar tantas cómo sea posible.



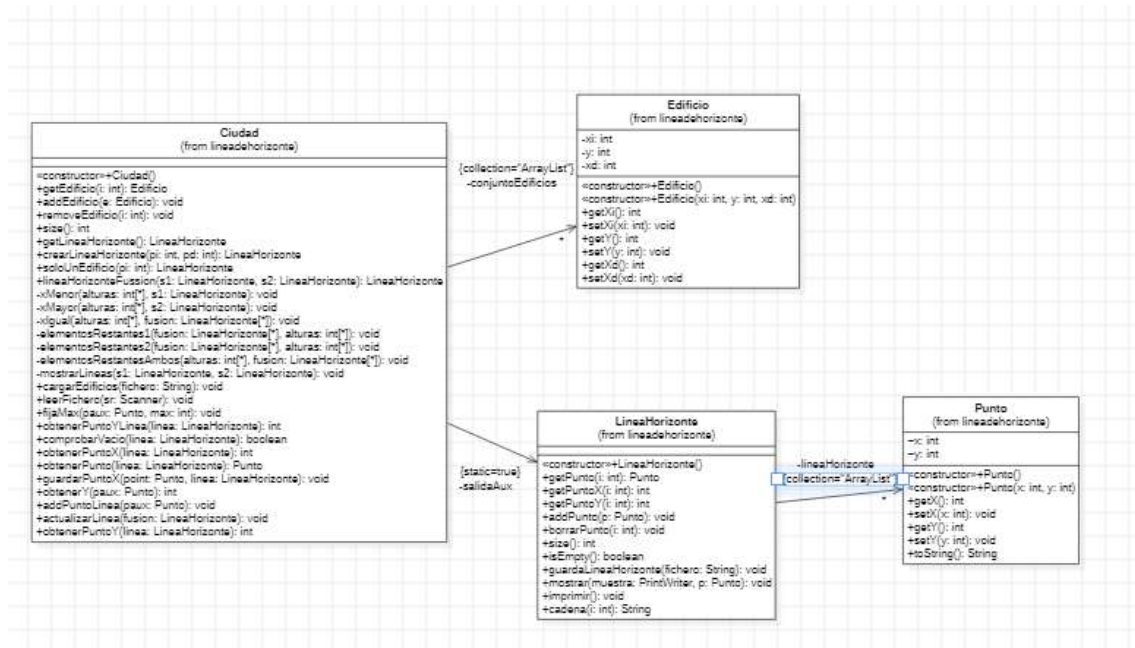
## Issues creados

<input type="checkbox"/>	12 Open ✓ 0 Closed	Author ▾	Label ▾	Projects ▾	Milestones ▾	Assignee ▾	Sort ▾
<input type="checkbox"/>	<b>Correct PMD Violations</b> enhancement #13 opened 23 minutes ago by hectorcanteroalvarez						
<input type="checkbox"/>	<b>Refactor Punto.java</b> enhancement #12 opened 1 hour ago by hectorcanteroalvarez						
<input type="checkbox"/>	<b>Refactor LineaHorizonte.java</b> enhancement #11 opened 1 hour ago by hectorcanteroalvarez						
<input type="checkbox"/>	<b>Refactor Edificio.Edificio(int,int,int)</b> enhancement #10 opened 1 hour ago by hectorcanteroalvarez						
<input type="checkbox"/>	<b>Refactor LineaHorizonte.guardaLineaHorizonte(String,Punto,FileWriter,PrintWriter,int)</b> enhancement #9 opened 1 hour ago by hectorcanteroalvarez						
<input type="checkbox"/>	<b>Refactor Ciudad.LineaHorizonteFussion(LineaHorizonte,LineaHorizonte,Punto,Punto,Punto)</b> enhancement #8 opened 1 hour ago by hectorcanteroalvarez						
<input type="checkbox"/>	<b>Refactor 6 lines occurring 2 times in 2 files: LineaHorizonte.java, Punto.java</b> enhancement #7 opened 1 hour ago by hectorcanteroalvarez						
<input type="checkbox"/>	<b>Refactor 7 lines occurring 2 times in Ciudad.java</b> enhancement #6 opened 1 hour ago by hectorcanteroalvarez						
<input type="checkbox"/>	<b>Refactor Ciudad.LineaHorizonteFussion(LineaHorizonte,LineaHorizonte,Punto,Punto,Punto)</b> enhancement #5 opened 1 hour ago by hectorcanteroalvarez						
<input type="checkbox"/>	<b>Refactor Ciudad.crearLineaHorizonte(int,int)</b> enhancement #4 opened yesterday by hectorcanteroalvarez						
<input type="checkbox"/>	<b>Refactor LineaHorizonte.guardaLineaHorizonte(String,Punto,FileWriter,PrintWriter,int)</b> enhancement #3 opened yesterday by hectorcanteroalvarez						
<input type="checkbox"/>	<b>Refactor Ciudad.cargarEdificios(String)</b> enhancement #2 opened yesterday by hectorcanteroalvarez						

Tras el análisis inicial, estos han sido los “issues” creados que se tratarán de resolver en este trabajo.

## Resultados de la refactorización

### Nuevo diagrama de clases



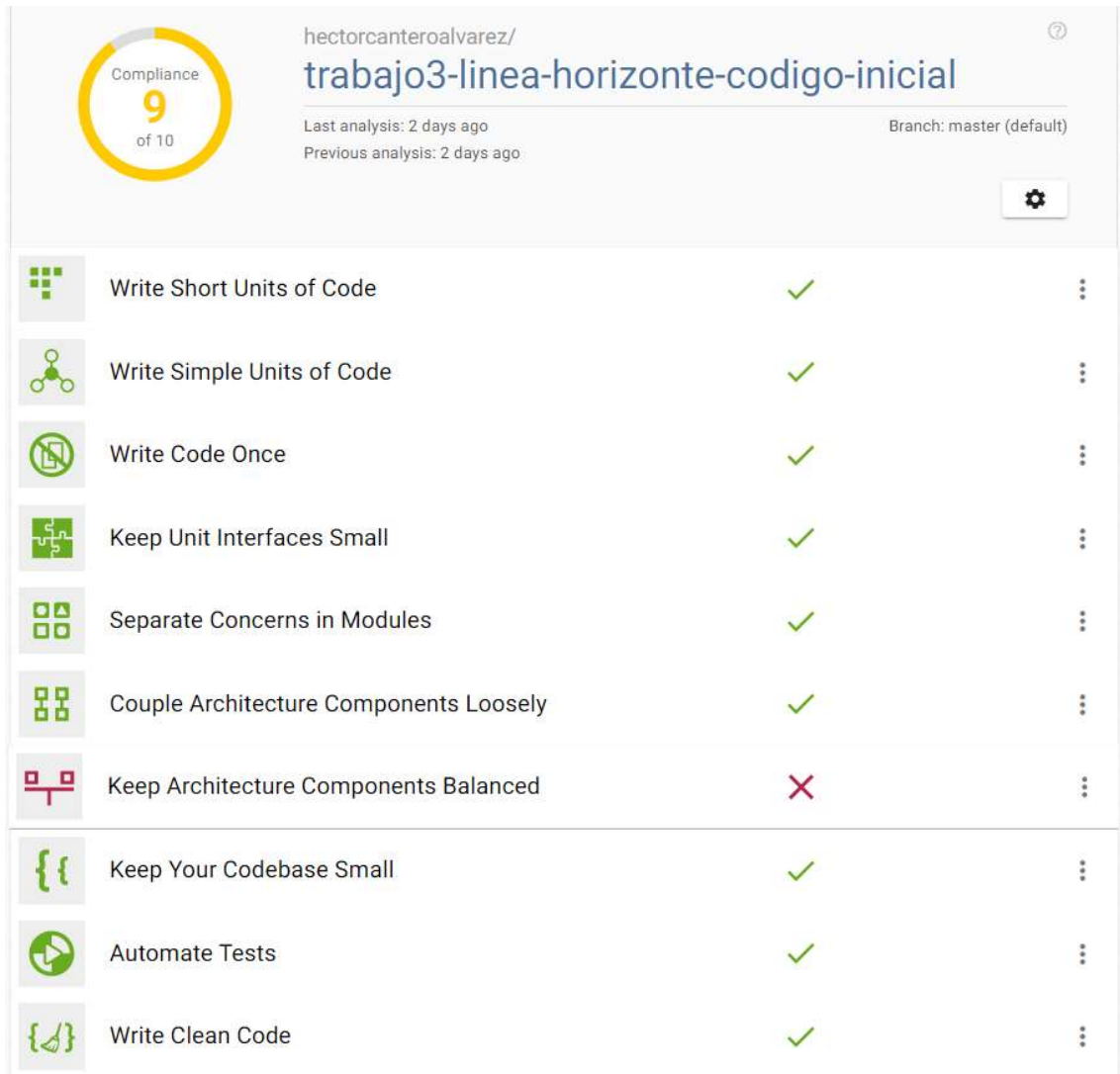
Tras la refactorización, este es el nuevo diagrama de clases. Como se puede apreciar, el número de métodos de *Ciudad* ha crecido considerablemente y ahora existe una relación directa entre esta y *LineaHorizonte*, ya que cada ciudad, tiene una *LineaHorizonte*.

### Issues resueltos

0 Open	11 Closed	Author	Label	Projects	Milestones	Assignee	Sort
			Correct PMD Violations	enhancement			1
		#14 by hectorcanteralvarez	was closed 19 seconds ago				
			Refactor Punto.java	enhancement			1
		#12 by hectorcanteralvarez	was closed 2 days ago				
			Refactor LineaHorizonte.java	enhancement			1
		#11 by hectorcanteralvarez	was closed 2 days ago				
			Refactor LineaHorizonte.guardaLineaHorizonte(String,Punto,FileWriter,PrintWriter,int)	enhancement			1
		#9 by hectorcanteralvarez	was closed 5 days ago				
			Refactor Ciudad.LineaHorizonteFusion(LineaHorizonte,LineaHorizonte,Punto,Punto,Punto)	enhancement			2
		#8 by hectorcanteralvarez	was closed 5 days ago				
			Refactor 6 lines occurring 2 times in 2 files: LineaHorizonte.java, Punto.java	enhancement			1
		#7 by hectorcanteralvarez	was closed 5 days ago				
			Refactor 7 lines occurring 2 times in Ciudad.java	enhancement			1
		#6 by hectorcanteralvarez	was closed 5 days ago				
			Refactor Ciudad.LineaHorizonteFusion(LineaHorizonte,LineaHorizonte,Punto,Punto,Punto)	enhancement			1
		#5 by hectorcanteralvarez	was closed 3 days ago				
			Refactor Ciudad.crearLineaHorizonte(int,int)	enhancement			1
		#4 by hectorcanteralvarez	was closed 7 days ago				
			Refactor LineaHorizonte.guardaLineaHorizonte(String,Punto,FileWriter,PrintWriter,int)	enhancement			1
		#3 by hectorcanteralvarez	was closed 8 days ago				
			Refactor Ciudad.cargarEdificios(String)	enhancement			1
		#2 by hectorcanteralvarez	was closed 8 days ago				

Estos son los “issues” que se han resuelto al refactorizar el código. No todos han podido ser resueltos, como se puede apreciar, sin embargo, como veremos a continuación, el nivel de conformidad ha aumentado considerablemente.

## Nivel final de conformidad alcanzado



#2 Se ha eliminado el código muerto de *cargarEdificios* y añadido un nuevo methodo ( aplicando extract method) *leerFichero* para reducir el número de líneas.

#3 Se ha creado un nuevo método *mostrar* que es llamado desde *guardarLineaHorizonte*, lo cual reduce el número de líneas de este último método al separar las funcionalidades.

#4 Se ha añadido el método *soloUnEdificio*, de forma que se ha reducido el número de líneas en *crearLineaHorizonte* separando la funcionalidad de esta. De paso se han movido algunas variables locales del método a otros métodos, para reducir el número de parámetros de entrada en #8.

#5 Se ha reducido el número de líneas en el método *lineaHorizonteFusion* dividiendo este en funcionalidades diferentes implementadas en varios métodos que son llamados desde el método principal. También se han empleado estructuras de datos de tipo array para reducir el número de parámetros de entrada.

#6 Se ha eliminado código muerto que no se usaba y daba lugar a código duplicado.

#7 Se ha cambiado el código duplicado por la llamada al método *toString* de la clase *Punto*, que ya que tenía la misma funcionalidad.

**#8** Se han eliminado los parámetros que no eran necesarios como entrada ya que era posible inicializarlos en el cuerpo del método, reduciendo el número de parámetros de entrada a dos.

**#11 y #12** se ha reducido el número de invocaciones directas a los métodos de otras clases, encapsulando en diferentes métodos dentro de la clase *Ciudad*, lo que facilita modificaciones de código futuras si fuera necesario.

**#13** Las violaciones de PMD revisadas son las siguientes:

Blocker violations: Todas han sido corregidas.

Critical violations: No han sido corregidas porque corresponde a llamadas System.Print.

Urgent violations: La mayoría no han sido corregidas ya que se referían a añadir comentarios, tamaño de los comentarios, tamaño del nombre de las variables o poner variables como final. Se han corregido *cast* innecesarios.

Important violations: Algunas no han sido corregidas porque se referían a quitar paréntesis que sí eran necesarios o usar "Varargs" en los parámetros de entrada de algunos métodos. Otras referidas a cambiar el nombre de variables como "out" o quitar paréntesis que, si eran innecesarios, sí han sido corregidas .