

ESTÁNDAR DE CONTEO

Equipo 2

9 de febrero de 2025

Realizó Mantenimiento:

Canul Ordoñez, Josué Israel

Garcilazo Cuevas, Mónica

Leo Fernández, José Carlos

Pool Flores, Endrick Alfredo

Rodríguez Coral, Samuel David

1 CONTROL DE DOCUMENTACIÓN

1.1 Control de configuración

Título:	Estándar de conteo
Referencia:	N/A
Autor:	Cristian David Pan Zaldivar
Fecha:	9 de febrero de 2025

1.2 Histórico de versiones

Versión	Fecha	Estado	Responsable	Nombre del archivo
1.0.3	09 – 02 – 2025	A	Cristian Pan	Estandar_de _conteo_1.0.3
1.0.4	04 – 03 – 2025	B	Cristian Pan	Estándar_de _conteo_1.04
2.0.0	02 – 04 – 2025	B	Arturo Quezada Daniel Rosado	Estándar_de _conteo_2.0.0
3.0.0	05 – 05 – 2025	A	José Carlos Leo Fernández	Estándar_de _conteo_3.0.0

Estado: (B)orador, (R)evisión, (A)probado

1.3 Histórico de cambios

Versión	Fecha	Cambios
1.0.1	09 – 02 – 2025	Se creó la estructura, así como se enlistaron los elementos a considerar para el conteo de líneas lógicas.
1.0.2	16 – 02 – 2025	Se añadió la descripción de cada uno de los apartados del documento.
1.0.3	25 – 02 – 2025	Corrección de formato y redacción.
1.0.4	04 – 03 – 2025	Redefinición de líneas lógicas
2.0.0	02 – 04 – 2025	Se elimino la descripción de las líneas lógicas para ajustarse a los requisitos del nuevo software. Se agregó la definición

		del conteo para los métodos de clases. Se agregaron las reglas para el conteo de líneas físicas y métodos en una clase interna.
3.0.0	05 – 05 – 2025	Se incorporaron las reglas para el conteo de cambios entre versiones: líneas añadidas, borradas y modificadas. Se estableció el formato de línea con un límite de 80 caracteres, considerando líneas divididas como una sola unidad lógica. También se definió el etiquetado de líneas nuevas y eliminadas.

1	Control de documentación	2
1.1	Control de configuración	2
1.2	Histórico de versiones	2
1.3	Histórico de cambios	2
2	Propósito	5
3	Estándar de conteo	5
3.1	Reglas Generales	5
3.2	Conteo de Líneas Físicas	6
3.3	Conteo de Métodos	6
3.4	Caso Especial: Clases Internas	6
3.5	Comparación de Líneas	7

CONTENIDO

2 PROPÓSITO

El siguiente estándar de conteo tiene como finalidad esclarecer los aspectos que serán considerados para la aplicación automatizada de la métrica de Líneas de Código (LOC) y de Métodos por clase. Se espera que este sea de utilidad para que los usuarios comprendan las reglas que se han establecido para diferenciar lo que es y no es una línea física.

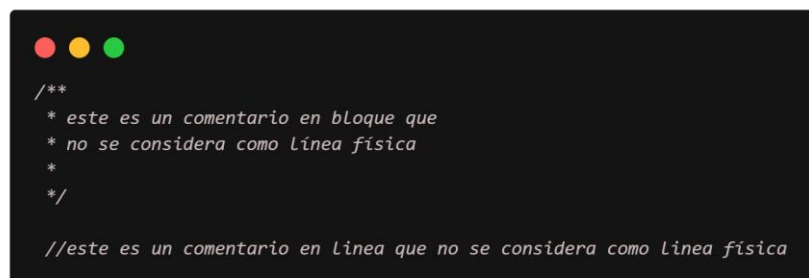
Cabe destacar, que es necesario seguir el documento “Estándar de codificación” realizado por el equipo para que haya consistencia en el conteo, principalmente se deberán de seguir aquellas reglas que influyen en el formato vertical del archivo .java.

3 ESTÁNDAR DE CONTEO

3.1 Reglas Generales

Para el presente estándar, no se contabilizará como parte de líneas físicas lo siguiente:

- Comentarios en línea (*//*).
- Comentarios en bloque (*/*...*/*).
- Líneas en blanco o solo con espacios y tabulaciones.



```
/**
 * este es un comentario en bloque que
 * no se considera como línea física
 */

//este es un comentario en línea que no se considera como línea física
```

Imagen 1. Ejemplo de comentarios en línea y en bloque

Por su parte, no se contabilizará como métodos lo siguiente:

Aquellos métodos que se encuentren fuera de una declaración de clase. Una clase en Java es una plantilla o estructura que define atributos y métodos para crear objetos. Un método se puede considerar parte de una clase si está dentro de sus corchetes.

```
public class Ejemplo {  
  
}
```

Imagen 2. Ejemplo de una clase

3.2 Conteo de Líneas Físicas

Como parte de las líneas físicas se contabilizarán todas las líneas que contengan código ejecutable o declaraciones y estén dentro de una declaración de clase.

```
// Esto es un comentario y no cuenta como línea física  
public class Ejemplo { // 1 línea física  
    private int valor; // 2 línea física  
  
    public void metodo() { // 3 línea física  
        valor = 5; // 4 línea física  
    } // 5 línea física  
} // 6 línea física
```

Imagen 3. Ejemplo de conteo de líneas físicas

3.3 Conteo de Métodos

Se contabilizarán como métodos todas aquellas rutinas que se encuentren dentro de una declaración de clase y que contengan código ejecutable o declaraciones.

```
public class MiClase {  
    public void metodoValido() {  
        System.out.println("Este método pertenece a una clase y será contabilizado.");  
    }  
}
```

Imagen 4. Ejemplo de método válido

3.4 Caso Especial: Clases Internas

En el caso de las clases internas, se aplican las siguientes reglas para el conteo de líneas y métodos:

Conteo de líneas físicas:

- La clase externa incluye tanto sus propias líneas físicas como las de la clase interna.
- La clase interna solamente incluye sus propias líneas físicas.

Conteo de métodos:

- La clase externa no incluye en su conteo de métodos aquellos que pertenezcan a la clase interna.
- En la clase interna se contabilizan únicamente los métodos definidos dentro de ella.

Por ejemplo, en el siguiente código (Img. 5), la clase “MiClase” tiene 1 método y 10 líneas físicas, mientras que la clase “MiClaseInterna” tiene 1 método y 5 líneas físicas.

```
1 public class MiClase {  
2  
3     public boolean metodo1(){  
4         return true;  
5     }  
6  
7     public class MiClaseInterna {  
8         public boolean metodo2(){  
9             return true;  
10        }  
11    }  
12  
13 }
```

Imagen 5. Ejemplo de código con una clase interna

3.5 Comparación de Líneas

Para analizar los cambios entre versiones de un programa, se establecerán las siguientes reglas:

Línea original: si una línea se encuentra idéntica en ambas versiones (anterior y nueva), se considera original y no ha sufrido cambios.


Línea añadida: si una línea está presente en la nueva versión pero no existe en la versión previa, se clasifica como añadida.

Línea borrada: si una línea está en la versión anterior pero no se encuentra en la nueva versión, se clasifica como borrada.

Además:

Líneas modificadas: cuando una línea añadida presenta una alta similitud con una línea borrada, se considera una modificación. Para determinar esto, se utilizará la distancia de Levenshtein como métrica de comparación. Si la similitud entre ambas líneas es igual o superior al 80%, se tratará como una modificación menor. En caso contrario, se considerará como una línea completamente nueva. Para más detalle sobre la métrica de distancia de Levenshtein, se puede consultar [GeeksforGeeks](#) o [ScienceDirect](#).

Regla de longitud de línea: cada línea no debe ocupar más allá de 80 caracteres. Por lo tanto, se debe formatear visualmente cualquier línea que rebase este límite en dos o más líneas. No obstante, para efectos del conteo, esta línea seguirá contando como una única línea de código.




```
=== COMPARACIÓN PARA Main.java ===

--- VERSIÓN ANTIGUA ---
[ORIGINAL] package mantenimiento.codecounter;
[DELETED] import mantenimiento.codecounter.exceptions.FolderNotFoundException;
[ORIGINAL] System.out.println("Bienvenido al sistema de conteo de líneas");

--- VERSIÓN NUEVA ---
[ORIGINAL] package mantenimiento.codecounter;
[NEW] import mantenimiento.codecounter.models.ProgramBuilder;
[MODIFIED] System.out.println("Bienvenido al sistema de conteo de código");
```

Imagen 6. Ejemplo de comparación de líneas.



```
[SPLITED] for (int i = 0, j = 0; i < content.size() && j < contentToCompare.size(); i++,
[NEW] j++) {

[SPLITED] System.out.println("\nIngrese la ruta del directorio para guardar los
[NEW] reportes:");

[SPLITED] System.out.printf("Líneas modificadas (MODIFIED): %d%n",
[NEW] globalChangeCounts.getOrDefault(MODIFIED, 0));
```

Imagen 7. Uso de la etiqueta [SPLITED] para líneas largas.