

Facultad de matemáticas

Reporte de casos de pruebas para el Módulo de Conteo de líneas lógicas

Versión 1.0

Autores:

Aaron Isaac Graniel Arzat

Fernando Joachín Prieto

David Peña Muñoz

Jose Luis Pooc Moo

Andrea Isabel Torres Perez

**CONTROL DE DOCUMENTACIÓN**

**Control de Configuración**

| **Título:** | Reporte de casos de pruebas para el Módulo de Conteo de líneas lógicas |
| --- | --- |
| **Referencia:** | REP\_CasosPruebasConteoLineasLogicas\_v1.0\_2025-03-05.docx |
| **Autor(es):** | David Peña Muñoz, Andrea Isabel Torres Perez |
| **Fecha:** | 05/03/2025 |

**Histórico de versiones**

| **Versión** | **Fecha** | **Estado** | **Responsable** | **Nombre de archivo** |
| --- | --- | --- | --- | --- |
| 1.0 | 05/03/2025 | A | Andrea Isabel Torres Perez | REP\_CasosPruebasConteoLineasLogicas\_v1.0\_2025-03-05.docx |

**Estado:** (B)orrador, (R)evisión, (A)probado

**Histórico de cambios**

| **Versión** | **Fecha** | **Cambios** |
| --- | --- | --- |
| 1.0 | 05/03/2025 | Primera versión. |
| 1.0 | 05/03/2025 | Actualización de casos de prueba |

**Índice**

[**1. Introducción 4**](#_qal6c07zv3iu)

[**2. Casos de pruebas unitarias del módulo de Conteo de líneas lógicas 4**](#_39awp88azdjy)

[2.1. No Líneas lógicas 4](#_3oxk5msgrf6a)

[2.2. Declaración de clase 7](#_pqlybgtr8xnv)

[2.3. Declaración de interfaz 10](#_30pszj5zi00c)

[2.4. Declaración de enum 13](#_83meftxjlgo3)

[2.5. Declaración de método 16](#_6jcjdsq048no)

[2.6. Estructura if 19](#_uuz7e62f5q59)

[2.7. Estructura for 21](#_slb8otn0crds)

[2.8. Estructura while 24](#_uxm45tng8ayq)

[2.9. Estructura switch 27](#_igt7v4nj0mu1)

[2.10. Estructura Try 30](#_n03yvvovtf99)

[2.11. Entrada mixta de líneas lógicas 33](#_89yv6de0qxb9)

# **Introducción**

Este documento detalla los casos de prueba unitarios para la verificación y validación del módulo de Conteo de líneas lógicas referente al sistema de conteo de líneas lógicas y físicas.

# **Casos de pruebas unitarias del módulo de Conteo de líneas lógicas**

### 2.1. No Líneas lógicas

*Tabla 1 - Caso de prueba 2.1*

| **Caso de prueba** | No existen líneas lógicas | | |
| --- | --- | --- | --- |
| **Identificador caso de prueba** | CP01\_NoLineasLogicas | | |
| **Función probar** | Contador de líneas lógicas. | | |
| **Objetivo** | Verificar que el método count retorne 0 cuando el archivo contiene únicamente líneas que no son consideradas lógicas. | | |
| **Descripción** | Se crea un archivo temporal con un conjunto de líneas que, al procesarlas, no se consideran lógicas. Se invoca el método *count* y se espera que el resultado sea 0. | | |
| **Criterios de éxito** | El método devuelve 0. | | |
| **Criterios de falla** | Retorna un valor distinto a 0 o lanza una excepción inesperada. | | |
| **Precondiciones** | El archivo temporal se crea correctamente en la ruta especificada.  Las líneas de código cumplen con la sintaxis estándar. | | |
| **Necesidades para el caso de prueba** | Datos de pruebas:  ● "package com.example;"  ● "import java.util.List;"  ● "" (línea vacía)  ● "// This is a comment"  ● "@Deprecated"  ● "int x = 0" | | |
| **Autor** | Andrea Torres Pérez | | |
| **Fecha de creación** | 05/03/2025 | | |
| **Flujo del caso de prueba** | **No paso** | **Usuario del sistema** | **Sistema** |
| **1** | Crea el archivo temporal con las líneas de entrada indicadas. |  |
| **2** | Se invoca el método *count* de LogicalLineCounter con dicho archivo. |  |
| **3** |  | El sistema procesa las líneas mediante isLogicalLine. |
| **4** |  | El sistema retorna el total de líneas lógicas (se espera 0). |
| **5** | Verifica que el resultado coincide con lo esperado. |  |
| **Post condiciones** | N/A. | | |

*Tabla 2 - Resultado del caso de prueba 2.1*

| **Numero de ejecución** | 1 | |
| --- | --- | --- |
| **Estatus (Pasó / No pasó)** | Pasó | |
| **Responsable de la ejecución** | Andrea Isabel Torres Pérez | |
| **Fecha de ejecución** | 05/03/2025 | |
| **Salida:** | No se generó ninguna salida visible en la terminal durante la ejecución de la prueba. | |
| **Faltas detectadas** | | |
| **Descripción falta** | | **Severidad (Alta, Media, Baja)** |
| N/A | | |
|  |  |  |

### 

### 

### 2.2. Declaración de clase

*Tabla 3 - Caso de prueba 2.2*

| **Caso de prueba** | Declaración de clase | | |
| --- | --- | --- | --- |
| **Identificador caso de prueba** | CP02\_DeclaracionDeClase | | |
| **Función probar** | Contador de líneas lógicas. | | |
| **Objetivo** | Verificar que la declaración de una clase se reconozca correctamente como una línea lógica. | | |
| **Descripción** | Se crea un archivo temporal que incluye la declaración de una clase. El método *count* debe identificar esta línea como lógica y contar 1 línea. | | |
| **Criterios de éxito** | El método retorna 1. | | |
| **Criterios de falla** | Retorna un valor distinto a 1, o lanza una excepción inesperada. | | |
| **Precondiciones** | El archivo se crea con sintaxis estándar para la declaración de clases. | | |
| **Necesidades para el caso de prueba** | Datos de pruebas:  ● "package com.example;"  ● ""  ● "public class TestClass {"  ● " // class body"  ● "}" | | |
| **Autor** | Andrea Torres Pérez | | |
| **Fecha de creación** | 05/03/2025 | | |
| **Flujo del caso de prueba** | **No paso** | **Usuario del sistema** | **Sistema** |
| **1** | Se genera el archivo temporal con las líneas de entrada. |  |
| **2** | Se invoca el método *count*. |  |
| **3** |  | El sistema procesa las líneas, detecta la declaración de la clase y suma 1 línea lógica. |
| **4** |  | Se retorna el valor 1. |
| **5** | El usuario verifica que el resultado es 1. |  |
| **Post condiciones** | N/A. | | |

*Tabla 4 - Resultado del caso de prueba 2.2*

| **Numero de ejecución** | 1 | |
| --- | --- | --- |
| **Estatus (Pasó / No pasó)** | Pasó | |
| **Responsable de la ejecución** | Andrea Isabel Torres Pérez | |
| **Fecha de ejecución** | 05/03/2025 | |
| **Salida:** | No se generó ninguna salida visible en la terminal durante la ejecución de la prueba. | |
| **Faltas detectadas** | | |
| **Descripción falta** | | **Severidad (Alta, Media, Baja)** |
| N/A | | |
|  |  |  |

### 

### 

### 2.3. Declaración de interfaz

*Tabla 5 - Caso de prueba 2.3*

| **Caso de prueba** | Declaración de interfaz | | |
| --- | --- | --- | --- |
| **Identificador caso de prueba** | CP03\_ DeclaraciónDeInterfaz | | |
| **Función probar** | Contador de líneas lógicas. | | |
| **Objetivo** | Verificar que la declaración de una interfaz se reconozca como una línea lógica. | | |
| **Descripción** | Se genera un archivo temporal que contiene la declaración de una interfaz. El método *count* debe reconocer dicha línea y contar 1 línea lógica. | | |
| **Criterios de éxito** | El método retorna 1. | | |
| **Criterios de falla** | Retorna un valor distinto a 1 o se produce algún error inesperado. | | |
| **Precondiciones** | El archivo se crea siguiendo la sintaxis estándar para interfaces. | | |
| **Necesidades para el caso de prueba** | Datos de pruebas:  ● "package com.example;"  ● ""  ● "public interface TestInterface {"  ● " // interface body"  ● "}" | | |
| **Autor** | Andrea Torres Pérez | | |
| **Fecha de creación** | 05/03/2025 | | |
| **Flujo del caso de prueba** | **No paso** | **Usuario del sistema** | **Sistema** |
| **1** | Se crea el archivo temporal con las líneas de entrada. |  |
| **2** | Se invoca el método *count*. |  |
| **3** |  | El sistema procesa la línea de declaración de la interfaz y la cuenta como 1 línea lógica. |
| **4** |  | Retorna el valor de 1. |
| **5** | El usuario verifica el resultado. |  |
| **Post condiciones** | N/A. | | |

*Tabla 6 - Resultado del caso de prueba 2.3*

| **Numero de ejecución** | 1 | |
| --- | --- | --- |
| **Estatus (Pasó / No pasó)** | Pasó | |
| **Responsable de la ejecución** | Andrea Isabel Torres Pérez | |
| **Fecha de ejecución** | 05/03/2025 | |
| **Salida:** | No se generó ninguna salida visible en la terminal durante la ejecución de la prueba. | |
| **Faltas detectadas** | | |
| **Descripción falta** | | **Severidad (Alta, Media, Baja)** |
| N/A | | |
|  |  |  |

### 

### 

### 2.4. Declaración de enum

*Tabla 7 - Caso de prueba 2.4*

| **Caso de prueba** | Declaración de enum | | |
| --- | --- | --- | --- |
| **Identificador caso de prueba** | CP04\_ DeclaraciónDeEnum | | |
| **Función probar** | Contador de líneas lógicas. | | |
| **Objetivo** | Verificar que la declaración de un enum se cuente como una línea lógica. | | |
| **Descripción** | Se crea un archivo temporal con una declaración de enum. El método *count* debe detectar la línea de declaración y contar 1 línea lógica. | | |
| **Criterios de éxito** | El método retorna 1. | | |
| **Criterios de falla** | El método devuelve un valor distinto a 1, y/o se genera un error inesperado. | | |
| **Precondiciones** | El archivo se genera con sintaxis estándar para enums. | | |
| **Necesidades para el caso de prueba** | Datos de pruebas:  ● "package com.example;"  ● ""  ● "public enum TestEnum {"  ● " VALUE1, VALUE2;"  ● " // enum body"  ● "}" | | |
| **Autor** | Andrea Torres Pérez | | |
| **Fecha de creación** | 05/03/2025 | | |
| **Flujo del caso de prueba** | **No paso** | **Usuario del sistema** | **Sistema** |
| **1** | Se crea el archivo temporal con las líneas de entrada. |  |
| **2** | Se invoca el método *count*. |  |
| **3** |  | El sistema detecta la declaración del enum y cuenta 1 línea lógica. |
| **4** |  | Retorna el valor de 1. |
| **5** | Verifica el resultado obtenido. |  |
| **Post condiciones** | N/A. | | |

*Tabla 8 - Resultado del caso de prueba 2.4*

| **Numero de ejecución** | 1 | |
| --- | --- | --- |
| **Estatus (Pasó / No pasó)** | Pasó | |
| **Responsable de la ejecución** | Andrea Isabel Torres Pérez | |
| **Fecha de ejecución** | 05/03/2025 | |
| **Salida:** | No se generó ninguna salida visible en la terminal durante la ejecución de la prueba. | |
| **Faltas detectadas** | | |
| **Descripción falta** | | **Severidad (Alta, Media, Baja)** |
| N/A | | |
|  |  |  |

### 

### 

### 2.5. Declaración de método

*Tabla 9 - Caso de prueba 2.5*

| **Caso de prueba** | Declaración de método | | |
| --- | --- | --- | --- |
| **Identificador caso de prueba** | CP05\_DeclaracionDeMetodo | | |
| **Función probar** | Contador de líneas lógicas. | | |
| **Objetivo** | Verificar que la declaración de un método se cuente como una línea lógica. | | |
| **Descripción** | Se crea un archivo temporal que contiene la declaración de un método. El método *count* debe identificar dicha declaración y contar 1 línea lógica. | | |
| **Criterios de éxito** | El método retorna 1. | | |
| **Criterios de falla** | Retorna un valor distinto a 1 o se produce algún error inesperado. | | |
| **Precondiciones** | El archivo contiene el formato adecuado para declaraciones de métodos. | | |
| **Necesidades para el caso de prueba** | Datos de pruebas:  ● "public static int dividir(int a, int b) {"  ● " return a / b;"  ● "}" | | |
| **Autor** | Andrea Torres Pérez | | |
| **Fecha de creación** | 05/03/2025 | | |
| **Flujo del caso de prueba** | **No paso** | **Usuario del sistema** | **Sistema** |
| **1** | Se genera el archivo temporal con las líneas indicadas. |  |
| **2** | Se genera el archivo temporal con las líneas indicadas. |  |
| **3** |  | El sistema procesa la línea y determina que es lógica. |
| **4** |  | El sistema retorna 1. |
| **5** | El usuario verifica que el resultado es el esperado. |  |
| **Post condiciones** | N/A. | | |

*Tabla 10 - Resultado del caso de prueba 2.5*

| **Numero de ejecución** | 1 | |
| --- | --- | --- |
| **Estatus (Pasó / No pasó)** | Pasó | |
| **Responsable de la ejecución** | Andrea Isabel Torres Pérez | |
| **Fecha de ejecución** | 05/03/2025 | |
| **Salida:** | No se generó ninguna salida visible en la terminal durante la ejecución de la prueba. | |
| **Faltas detectadas** | | |
| **Descripción falta** | | **Severidad (Alta, Media, Baja)** |
| N/A | | |
|  |  |  |

### 

### 

### 2.6. Estructura if

*Tabla 11 - Caso de prueba 2.6*

| **Caso de prueba** | Estructura if | | |
| --- | --- | --- | --- |
| **Identificador caso de prueba** | CP06\_ EstructuraIf | | |
| **Función probar** | Contador de líneas lógicas. | | |
| **Objetivo** | Verificar que la sentencia if se cuente como una línea lógica. | | |
| **Descripción** | Se crea un archivo temporal que contiene una sentencia if con su bloque. El método *count* debe identificar la sentencia if y contar 1 línea lógica. | | |
| **Criterios de éxito** | El método devuelve true cuando es la declaración de un método y false en cualquier otro caso. | | |
| **Criterios de falla** | El método retorna 1. | | |
| **Precondiciones** | La sintaxis del if es la estándar. | | |
| **Necesidades para el caso de prueba** | Datos de pruebas:  ● "if (x > 0) {"  ● " // do something"  ● "}" | | |
| **Autor** | Andrea Torres Pérez | | |
| **Fecha de creación** | 05/03/2025 | | |
| **Flujo del caso de prueba** | **No paso** | **Usuario del sistema** | **Sistema** |
| **1** | Se genera el archivo temporal con las líneas indicadas. |  |
| **2** | Se genera el archivo temporal con las líneas indicadas. |  |
| **3** |  | El sistema procesa la línea y determina que es lógica. |
| **4** |  | El sistema retorna 1. |
| **5** | El usuario verifica que el resultado es el esperado. |  |
| **Post condiciones** | N/A. | | |

*Tabla 12 - Resultado del caso de prueba 2.6*

| **Numero de ejecución** | 1 | |
| --- | --- | --- |
| **Estatus (Pasó / No pasó)** | Pasó | |
| **Responsable de la ejecución** | Andrea Isabel Torres Pérez | |
| **Fecha de ejecución** | 05/03/2025 | |
| **Salida:** | No se generó ninguna salida visible en la terminal durante la ejecución de la prueba. | |
| **Faltas detectadas** | | |
| **Descripción falta** | | **Severidad (Alta, Media, Baja)** |
| N/A | | |

### 2.7. Estructura for

*Tabla 13 - Caso de prueba 2.7*

| **Caso de prueba** | Estructura for | | |
| --- | --- | --- | --- |
| **Identificador caso de prueba** | CP07\_EstructuraFor | | |
| **Función probar** | Contador de líneas lógicas. | | |
| **Objetivo** | Verificar que la sentencia for se cuente como una línea lógica. | | |
| **Descripción** | Se genera un archivo temporal con una estructura for. El método *count* debe reconocer la línea for como lógica y contar 1 línea. | | |
| **Criterios de éxito** | El método retorna 1. | | |
| **Criterios de falla** | Retorna un valor distinto a 1 o se produce algún error inesperado. | | |
| **Precondiciones** | La sintaxis del for es la estándar. | | |
| **Necesidades para el caso de prueba** | Datos de pruebas:  ● "for (int i = 0; i < 10; i++) {"  ● " // loop body"  ● "}" | | |
| **Autor** | Andrea Torres Pérez | | |
| **Fecha de creación** | 05/03/2025 | | |
| **Flujo del caso de prueba** | **No paso** | **Usuario del sistema** | **Sistema** |
| **1** | Se genera el archivo temporal con las líneas indicadas. |  |
| **2** | Se genera el archivo temporal con las líneas indicadas. |  |
| **3** |  | El sistema procesa la línea y determina que es lógica. |
| **4** |  | El sistema retorna 1. |
| **5** | El usuario verifica que el resultado es el esperado. |  |
| **Post condiciones** | N/A. | | |

*Tabla 14 - Resultado del caso de prueba 2.7*

| **Numero de ejecución** | 1 | |
| --- | --- | --- |
| **Estatus (Pasó / No pasó)** | Pasó | |
| **Responsable de la ejecución** | Andrea Isabel Torres Pérez | |
| **Fecha de ejecución** | 05/03/2025 | |
| **Salida:** | No se generó ninguna salida visible en la terminal durante la ejecución de la prueba. | |
| **Faltas detectadas** | | |
| **Descripción falta** | | **Severidad (Alta, Media, Baja)** |
| N/A | | |
|  |  |  |

### 

### 

### 2.8. Estructura while

*Tabla 15 - Caso de prueba 2.8*

| **Caso de prueba** | Estructura while | | |
| --- | --- | --- | --- |
| **Identificador caso de prueba** | CP08\_ EstructuraWhile | | |
| **Función probar** | Contador de líneas lógicas. | | |
| **Objetivo** | Verificar que la sentencia while se cuente como una línea lógica. | | |
| **Descripción** | Se crea un archivo temporal con una estructura while. El método *count* debe detectar la sentencia while y contar 1 línea lógica. | | |
| **Criterios de éxito** | El método retorna 1. | | |
| **Criterios de falla** | Retorna un valor distinto a 1 o se produce algún error inesperado. | | |
| **Precondiciones** | ● El archivo se genera con sintaxis estándar para while. | | |
| **Necesidades para el caso de prueba** | Datos de entrada:  ● "while (x < 10) {"  ● " // loop body"  ● "}" | | |
| **Autor** | David Muñoz Peña | | |
| **Fecha de creación** | 05/03/2025 | | |
| **Flujo del caso de prueba** | **No paso** | **Usuario del sistema** | **Sistema** |
| 1 | Se genera el archivo temporal con las líneas indicadas. |  |
| 2 | Se genera el archivo temporal con las líneas indicadas. |  |
| 3 |  | El sistema procesa la línea y determina que es lógica. |
| 4 |  | El sistema retorna 1. |
| 5 | El usuario verifica que el resultado es el esperado. |  |
| **Post condiciones** | N/A | | |

*Tabla 16 - Resultado del caso de prueba 2.8*

| **Numero de ejecución** | 1 | |
| --- | --- | --- |
| **Estatus (Pasó / No pasó)** | Pasó | |
| **Responsable de la ejecución** | David Muñoz Peña | |
| **Fecha de ejecución** | 05/03/2025 | |
| **Salida:** | No se generó ninguna salida visible en la terminal durante la ejecución de la prueba. | |
| **Faltas detectadas** | | |
| **Descripción falta** | | **Severidad (Alta, Media, Baja)** |
| N/A | | |
|  |  |  |

### 

### 

### 2.9. Estructura switch

*Tabla 17 - Caso de prueba 2.9*

| **Caso de prueba** | Estructura switch | | |
| --- | --- | --- | --- |
| **Identificador caso de prueba** | CP09\_ EstructuraSwitch | | |
| **Función probar** | Contador de líneas lógicas. | | |
| **Objetivo** | Verificar que la estructura switch se procese correctamente, distinguiendo entre la declaración del switch y los cases, según la lógica implementada. | | |
| **Descripción** | Se crea un archivo temporal con un bloque switch-case. Dependiendo de la implementación de LogicalLineCounter, se espera que se cuente la parte inicial. | | |
| **Criterios de éxito** | El método retorna 1. | | |
| **Criterios de falla** | Retorna un valor distinto a 1 o se produce algún error inesperado. | | |
| **Precondiciones** | El bloque switch-case respeta la sintaxis estándar de Java. | | |
| **Necesidades para el caso de prueba** | Datos de prueba:  · "switch (x) {"  · " case 1:"  · " break;"  · " default:"  · " break;"  · "}" | | |
| **Autor** | David Muñoz Peña | | |
| **Fecha de creación** | 05/03/2025 | | |
| **Flujo del caso de prueba** | **No paso** | **Usuario del sistema** | **Sistema** |
| 1 | Se genera el archivo temporal con las líneas indicadas. |  |
| 2 | Se genera el archivo temporal con las líneas indicadas. |  |
| 3 |  | El sistema procesa la línea y determina que es lógica. |
| 4 |  | El sistema retorna 1. |
| 5 | El usuario verifica que el resultado es el esperado. |  |
| **Post condiciones** | N/A | | |

*Tabla 18 - Resultado del caso de prueba 2.9*

| **Numero de ejecución** | 1 | |
| --- | --- | --- |
| **Estatus (Pasó / No pasó)** | Pasó | |
| **Responsable de la ejecución** | David Muñoz Peña | |
| **Fecha de ejecución** | 05/03/2025 | |
| **Salida:** | No se generó ninguna salida visible en la terminal durante la ejecución de la prueba. | |
| **Faltas detectadas** | | |
| **Descripción falta** | | **Severidad (Alta, Media, Baja)** |
| N/A | | |
|  |  |  |

### 

### 

### 2.10. Estructura Try

*Tabla 19 - Caso de prueba 2.10*

| **Caso de prueba** | Estructura Try | | |
| --- | --- | --- | --- |
| **Identificador caso de prueba** | CP10\_ EstructuraTry | | |
| **Función probar** | Contador de líneas lógicas. | | |
| **Objetivo** | Verificar que una estructura try se cuente como una única línea lógica. | | |
| **Descripción** | Se genera un archivo temporal con un bloque try-catch. El método *count* debe identificar que la estructura completa se cuenta como 1 línea lógica. | | |
| **Criterios de éxito** | El método retorna 1. | | |
| **Criterios de falla** | Retorna un valor distinto a 1 o se produce algún error inesperado. | | |
| **Precondiciones** | El archivo se genera con sintaxis estándar para try-catch. | | |
| **Necesidades para el caso de prueba** | Datos de prueba:  · "try {"  · " // try block"  · "} catch(Exception e) {"  · " // handle exception"  · "}" | | |
| **Autor** | David Muñoz Peña | | |
| **Fecha de creación** | 05/03/2025 | | |
| **Flujo del caso de prueba** | **No paso** | **Usuario del sistema** | **Sistema** |
| 1 | Se genera el archivo temporal con las líneas indicadas. |  |
| 2 | Se genera el archivo temporal con las líneas indicadas. |  |
| 3 |  | El sistema procesa la línea y determina que es lógica. |
| 4 |  | El sistema retorna 1. |
| 5 | El usuario verifica que el resultado es el esperado. |  |
| **Post condiciones** | N/A | | |

*Tabla 20 - Resultado del caso de prueba 2.10*

| **Numero de ejecución** | 1 | |
| --- | --- | --- |
| **Estatus (Pasó / No pasó)** | Pasó | |
| **Responsable de la ejecución** | David Muñoz Peña | |
| **Fecha de ejecución** | 05/03/2025 | |
| **Salida:** | No se generó ninguna salida visible en la terminal durante la ejecución de la prueba. | |
| **Faltas detectadas** | | |
| **Descripción falta** | | **Severidad (Alta, Media, Baja)** |
| N/A | | |
|  |  |  |

### 

### 

### 2.11. Entrada mixta de líneas lógicas

*Tabla 21 - Resultado del caso de prueba 2.11*

| **Caso de prueba** | Entrada mixta de líneas lógicas | | |
| --- | --- | --- | --- |
| **Identificador caso de prueba** | CP11\_ EntradaMixtaDeLíneasLogicas | | |
| **Función probar** | Contador de líneas lógicas. | | |
| **Objetivo** | Validar que el método *count* procese correctamente un bloque de código mixto, retornando el número exacto de líneas lógicas. | | |
| **Descripción** | Se crea un archivo temporal que contiene una combinación de declaraciones, estructuras de control y otros elementos. El método *count* debe analizar cada línea y sumar únicamente aquellas que sean lógicas, devolviendo el total esperado. | | |
| **Criterios de éxito** | El método retorna 4. | | |
| **Criterios de falla** | Retorna un valor distinto a 4 o se produce algún error inesperado. | | |
| **Precondiciones** | El archivo se genera con las líneas en el orden correcto y respetando la sintaxis estándar. | | |
| **Necesidades para el caso de prueba** | Datos de prueba:  · "package com.example;",  · "import java.util.List;",  · "",  · "public class MixedExample {",  · " Example example = new Example()",  · " public MixedExample() {",  · " // constructor body",  · " }",  · "",  · " public void exampleMethod() {",  · " if (x > 0) {",  · " System.out.println(\"Positive\");",  · " }",  · " }",  · "}" | | |
| **Autor** | David Muñoz Peña | | |
| **Fecha de creación** | 05/03/25 | | |
| **Flujo del caso de prueba** | **No paso** | **Usuario del sistema** | **Sistema** |
| 1 | Se genera el archivo temporal con las líneas indicadas. |  |
| 2 | Se genera el archivo temporal con las líneas indicadas. |  |
| 3 |  | El sistema procesa las línea y determina cuales son líneas lógica. |
| 4 |  | El sistema retorna 4. |
| 5 | El usuario verifica que el resultado es el esperado. |  |
| **Post condiciones** | N/A | | |

*Tabla 22 - Resultado del caso de prueba 2.11*

| **Numero de ejecución** | 1 | |
| --- | --- | --- |
| **Estatus (Pasó / No pasó)** | Pasó | |
| **Responsable de la ejecución** | David Muñoz Peña | |
| **Fecha de ejecución** | 05/03/2025 | |
| **Salida:** | No se generó ninguna salida visible en la terminal durante la ejecución de la prueba. | |
| **Faltas detectadas** | | |
| **Descripción falta** | | **Severidad (Alta, Media, Baja)** |
| N/A | | |