

Facultad de matemáticas

Reporte de casos de pruebas para el Módulo de validación del formato del archivo entrante

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 validación del formato del archivo entrante |
| --- | --- |
| **Referencia:** |  |
| **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\_CasosPruebasValidacionFormatoArchivoEntrante\_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. |
|  |  |  |

**Índice**

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

[**2. Casos de pruebas unitarias del módulo de validación del formato del archivo entrante 4**](#_39awp88azdjy)

[2.1. Validación de la longitud de la línea menor a 120 caracteres 4](#_8gpinnsq41ua)

[2.2. Validación de la longitud de la línea mayor a 120 6](#_4jb6l9436mi8)

[2.3. Validación de una sola declaración ejecutable 9](#_3yxp8fmrnxk7)

[2.4. Validación de múltiples declaraciones ejecutables 11](#_jych9gytjair)

[2.5. Validación de llaves del estilo K&R 14](#_vgqwms7e6b0)

[2.6. Validación de llaves del estilo Allman 16](#_gji6pc6bb8m3)

[2.7. Validación de importaciones explícitas 18](#_2mczbi5mbsyb)

[2.8. Validación de importaciones comodines 21](#_crn4esm02ed1)

[2.9. Validación de anotaciones correctas 23](#_5ujiaiz14z8h)

[2.10. Validación de anotaciones incorrectas 25](#_br8s1zou7qur)

# **Introducción**

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

# **Casos de pruebas unitarias del módulo de validación del formato del archivo entrante**

## 2.1. Validación de la longitud de la línea menor a 120 caracteres

*Tabla 1 - Caso de prueba 2.1*

| **Caso de prueba** | Validación de la longitud de la línea menor a 120 caracteres | | |
| --- | --- | --- | --- |
| **Identificador caso de prueba** | CP11\_ValidacionLongitudLineaMenor120 | | |
| **Función probar** | FileFormatValidator.isValidLineLength | | |
| **Objetivo** | Verificar que el método isValidLineLength retorne true para líneas con menos de 120 caracteres | | |
| **Descripción** | Se prueba el método isValidLineLength solo acepte líneas con menos de 120 caracteres. | | |
| **Criterios de éxito** | El método retorna true para líneas con menos de 120 caracteres | | |
| **Criterios de falla** | El método retorna false para líneas con menos de 120 caracteres | | |
| **Precondiciones** | N/A | | |
| **Necesidades para el caso de prueba** | Un objeto javaFile con lo siguiente:  Una lista con el contenido del archivo a contar pero con sentencias menores a los 120 caracteres.  List<String> shortLineFile = Arrays.asList(  "public class Test {",  " public static void main(String[] args) {",  " System.out.println(\"Hello, World!\");",  " }",  "}"  );  El nombre del archivo con la extensión .java. ValidFile.java | | |
| **Autor** | Fernando Joachin Prieto | | |
| **Fecha de creación** | 23/02/2025 | | |
| **Flujo del caso de prueba** | **No paso** | **Usuario del sistema** | **Sistema** |
| **1** | **El usuario llamada al método isValidLineLength y le pasa el nombre del archivo y la lista del contenido con las líneas a contar** | **El sistema ejecuta isValidFileFormat con los parametros recibidos y llama a isValidLineLength para evaluar la longitud de la linea** |
| **2** |  | **El sistema retorna un true.** |
| **Post condiciones** | El sistema permite el conteo de lineas fisicas y logicas del archivo. | | |

*Tabla 2 - Resultado del caso de prueba 2.1*

| **Número de ejecución** | 1 | |
| --- | --- | --- |
| **Estatus (Pasó / No Pasó)** | Pasó | |
| **Responsable de la ejecucuión** | Fernando Joachín Prieto | |
| **Fecha de ejecución** | 23/02/2025 | |
| **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. Validación de la longitud de la línea mayor a 120

*Tabla 3 - Caso de prueba 2.2*

| **Caso de prueba** | Validación de la longitud de la línea mayor a 120 | | |
| --- | --- | --- | --- |
| **Identificador caso de prueba** | CP12\_ValidacionLongitudLineaMayor120 | | |
| **Función probar** | FileFormatValidator.isValidLineLength | | |
| **Objetivo** | Verificar que el método isValidLineLength retorne false para líneas con mayores de 120 caracteres | | |
| **Descripción** | Se prueba el método isValidLineLength no acepte líneas con más de 120 caracteres. | | |
| **Criterios de éxito** | El método retorna false para líneas con más de 120 caracteres | | |
| **Criterios de falla** | El método retorna true para líneas con más de 120 caracteres | | |
| **Precondiciones** | N/A | | |
| **Necesidades para el caso de prueba** | Un objeto javaFile con lo siguiente:  Una lista con el contenido del archivo a contar pero con una sentencia mayor a los 120 caracteres.  List<String> longLineFile = Arrays.asList(  "public class Test {",  " public static void main(String[] args) {",  " String mensaje = \"El sistema ha procesado la solicitud exitosamente y ha generado un identificador único para su referencia: 1234567890. Por favor, guarde este identificador para futuras consultas.\"; ",  " System.out.println(mensaje);",  " }",  "}"  );  El nombre del archivo con la extensión .java InvalidFile.java | | |
| **Autor** | Fernando Joachin Prieto | | |
| **Fecha de creación** | 23/02/2025 | | |
| **Flujo del caso de prueba** | **No paso** | **Usuario del sistema** | **Sistema** |
| **1** | **El usuario llamada al método isValidLineLength y le pasa el nombre del archivo y la lista del contenido con las líneas a contar.** | **El sistema ejecuta isValidFileFormat con los parametros recibidos y llama a isValidLineLength para evaluar la longitud de la línea.** |
|  | **2** |  | **El sistema retorna un false.** |
|  | **3** |  | **El sistema imprime**  **“Warning: Line 3 in file InvalidFile.java exceeds 120 characters.”** |
| **Post condiciones** | El sistema notifica sobre el error al usuario y no realiza el conteo de líneas físicas y lógicas del archivo. | | |

*Tabla 4 - Resultado del caso de prueba 2.2*

| **Número de ejecución** | 1 | |
| --- | --- | --- |
| **Estatus (Pasó / No Pasó)** | Pasó | |
| **Responsable de la ejecucuión** | Fernando Joachín Prieto | |
| **Fecha de ejecución** | 23/02/2025 | |
| **Salida** | Warning: Line 3 in file InvalidFile.java exceeds 120 characters. | |
| **Faltas detectadas** | | |
| **Descripción falta** | | **Severidad (Alta, Media, Baja)** |
| N/A | | |

### 

## 2.3. Validación de una sola declaración ejecutable

*Tabla 5 - Caso de prueba 2.3*

| **Caso de prueba** | Validación de una sola declaración ejecutable | | |
| --- | --- | --- | --- |
| **Identificador caso de prueba** | CP13\_ValidacionUnaDeclaracionEjectuable | | |
| **Función probar** | FileFormatValidator.isValidMultipleStatements | | |
| **Objetivo** | Verificar que el método isValidMultipleStatements retorne true para líneas con una sola declaración ejecutable. | | |
| **Descripción** | Se prueba el método isValidMultipleStatements con líneas de código para asegurar que acepte una sola declaración ejecutable. | | |
| **Criterios de éxito** | El método retorna true para líneas con una sola declaración ejecutable | | |
| **Criterios de falla** | El método retorna false para líneas con una sola declaración ejecutable | | |
| **Precondiciones** | N/A | | |
| **Necesidades para el caso de prueba** | Un objeto javaFile con lo siguiente:  Una lista con el contenido del archivo a contar pero con líneas que tiene sola una declaración ejecutable.  List<String> singleStatementFile = Arrays.asList(  "public class Test {",  " public static void main(String[] args) {",  " int a = 0;",  " for (int i = 0; i < 10; i++) {",  " System.out.println(a+i);",  " }",  " }",  "}"  );  El nombre del archivo con la extensión .java ValidFile.java | | |
| **Autor** | Fernando Joachin Prieto | | |
| **Fecha de creación** | 23/02/2025 | | |
| **Flujo del caso de prueba** | **No paso** | **Usuario del sistema** | **Sistema** |
| **1** | **El usuario llamada al método isValidLineLength y le pasa el nombre del archivo y la lista del contenido con las líneas a contar** | **El sistema ejecuta isValidFileFormat con los parametros recibidos y llama a isValidMultipleStatements para evaluar que solamente contenga líneas con una sola declaración ejecutable.** |
| **2** |  | **El sistema retorna un true.** |
| **Post condiciones** | El sistema permite el conteo de lineas fisicas y logicas del archivo. | | |

*Tabla 6 - Resultado del caso de prueba 2.3*

| **Número de ejecución** | 1 | |
| --- | --- | --- |
| **Estatus (Pasó / No Pasó)** | Pasó | |
| **Responsable de la ejecucuión** | Fernando Joachín Prieto | |
| **Fecha de ejecución** | 23/02/2025 | |
| **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. Validación de múltiples declaraciones ejecutables

*Tabla 7 - Caso de prueba 2.4*

| **Caso de prueba** | Validación de múltiples declaraciones ejecutables | | |
| --- | --- | --- | --- |
| **Identificador caso de prueba** | CP14\_ValidacionMúltiplesDeclaracionesEjecutables | | |
| **Función probar** | FileFormatValidator.isValidMultipleStatements | | |
| **Objetivo** | Verificar que el método isValidBracesStyle retorne false para múltiples declaraciones ejecutables. | | |
| **Descripción** | Se prueba el método isValidBracesStyle con líneas de código que tenga múltiples declaraciones ejecutables para asegurarse que no sean aceptadas. | | |
| **Criterios de éxito** | El método retorna false para líneas con múltiples declaraciones ejecutables | | |
| **Criterios de falla** | El método retorna true para líneas con múltiples declaraciones ejecutables | | |
| **Precondiciones** | N/A | | |
| **Necesidades para el caso de prueba** | Un objeto javaFile con lo siguiente:  Una lista con el contenido del archivo a contar pero con líneas que tienen múltiples declaraciones ejecutables.  List<String> multipleStatementsFile = Arrays.asList(  "public class Test {",  " public static void main(String[] args) {",  " int a = 5; int b = 10;",  " for (int i = 0; i < 10; i++) {",  " System.out.println(a+b+i);",  " }",  " }",  "}"  );  El nombre del archivo con la extensión .java InvalidFile.java | | |
| **Autor** | Fernando Joachin Prieto | | |
| **Fecha de creación** | 23/02/2025 | | |
| **Flujo del caso de prueba** | **No paso** | **Usuario del sistema** | **Sistema** |
| **1** | **El usuario llamada al método isValidLineLength y le pasa el nombre del archivo y la lista del contenido con las líneas a contar** | **El sistema ejecuta isValidFileFormat con los parametros recibidos y llama a isValidMultipleStatements para evaluar que solamente contenga líneas con una sola declaración ejecutable.** |
| **2** |  | **El sistema retorna un false.** |
|  | **3** |  | **El sistema imprime**  **"Error: Line 3 file InvalidFile.java contains multiple executable statements."** |
| **Post condiciones** | El sistema notifica sobre el error al usuario y no realiza el conteo de líneas físicas y lógicas del archivo. | | |

*Tabla 8 - Resultado del caso de prueba 2.4*

| **Número de ejecución** | 1 | |
| --- | --- | --- |
| **Estatus (Pasó / No Pasó)** | Pasó | |
| **Responsable de la ejecucuión** | Fernando Joachín Prieto | |
| **Fecha de ejecución** | 23/02/2025 | |
| **Salida** | Error: Line 3 in file InvalidFile.java contains multiple executable statements. | |
| **Faltas detectadas** | | |
| **Descripción falta** | | **Severidad (Alta, Media, Baja)** |
| N/A | | |

### 

## 2.5. Validación de llaves del estilo K&R

*Tabla 9 - Caso de prueba 2.5*

| **Caso de prueba** | Validación de llaves del estilo K&R | | |
| --- | --- | --- | --- |
| **Identificador caso de prueba** | CP15\_ValidacionLlavesEstiloK&R | | |
| **Función probar** | FileFormatValidator.isValidBracesStyle | | |
| **Objetivo** | Verificar que el método isValidBracesStyle retorne true para líneas con llaves al estilo K&R. | | |
| **Descripción** | Se prueba el método isValidBracesStyle con líneas de código para asegurar que las llaves sigan el estilo K&R. | | |
| **Criterios de éxito** | El método retorna true para líneas con llaves que siguen el estilo K&R. | | |
| **Criterios de falla** | El método retorna false para líneas con llaves que siguen el estilo K&R. | | |
| **Precondiciones** | N/A | | |
| **Necesidades para el caso de prueba** | Un objeto javaFile con lo siguiente:  Una lista con el contenido del archivo a contar pero con líneas códigos que siguen el estilo K&R.  List<String> validBraceFile = Arrays.asList(  "public class Test {",  " public static void main(String[] args) {",  " System.out.println(\"Hello, World!\");",  " }",  "}"  );  El nombre del archivo con la extensión .java ValidFile.java | | |
| **Autor** | Fernando Joachin Prieto | | |
| **Fecha de creación** | 23/02/2025 | | |
| **Flujo del caso de prueba** | **No paso** | **Usuario del sistema** | **Sistema** |
| **1** | **El usuario llamada al método isValidFileFormat y le pasa el nombre del archivo y la lista del contenido con las líneas a contar** | **El sistema ejecuta isValidFileFormat con los parametros recibidos y llama a isValidBracesStyle para evaluar el estilo de las llaves siguen el estilo K&R.** |
| **2** |  | **El sistema retorna un true.** |
| **Post condiciones** | El sistema permite el conteo de lineas fisicas y logicas del archivo. | | |

*Tabla 10 - Resultado del caso de prueba 2.5*

| **Número de ejecución** | 1 | |
| --- | --- | --- |
| **Estatus (Pasó / No Pasó)** | Pasó | |
| **Responsable de la ejecucuión** | Fernando Joachín Prieto | |
| **Fecha de ejecución** | 23/02/2025 | |
| **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. Validación de llaves del estilo Allman

*Tabla 11 - Caso de prueba 2.6*

| **Caso de prueba** | Validación de llaves del estilo Allman | | |
| --- | --- | --- | --- |
| **Identificador caso de prueba** | CP16\_ValidacionLlavesEstiloAllman | | |
| **Función probar** | FileFormatValidator.isValidBracesStyle | | |
| **Objetivo** | Verificar que el método isValidBracesStyle retorne false para líneas con llaves al estilo Allman. | | |
| **Descripción** | Se prueba el método isValidBracesStyle con líneas de código para asegurar que las llaves del estilo Allman no sean aceptadas. | | |
| **Criterios de éxito** | El método retorna false para líneas con llaves que siguen el estilo Allman. | | |
| **Criterios de falla** | El método retorna true para líneas con llaves que siguen el estilo Allman. | | |
| **Precondiciones** | N/A | | |
| **Necesidades para el caso de prueba** | Un objeto javaFile con lo siguiente:  Una lista con el contenido del archivo a contar pero con líneas códigos que siguen el estilo Allman.  List<String> invalidBraceFile = Arrays.asList(  "public class Test {",  " public static void main(String[] args)",  "{",  " System.out.println(\"Hello, World!\");",  " }",  "}"  );  El nombre del archivo con la extensión .java InvalidFile.java | | |
| **Autor** | Fernando Joachin Prieto | | |
| **Fecha de creación** | 23/02/2025 | | |
| **Flujo del caso de prueba** | **No paso** | **Usuario del sistema** | **Sistema** |
| **1** | **El usuario llamada al método isValidFileFormat y le pasa el nombre del archivo y la lista del contenido con las líneas a contar** | **El sistema ejecuta isValidFileFormat con los parametros recibidos y llama a isValidBracesStyle para evaluar el estilo de las llaves siguen el estilo K&R.** |
| **2** |  | **El sistema retorna un false.** |
|  | **3** |  | **El sistema imprime**  **“Error: Line 3 in file InvalidFile.java has incorrect brace style."** |
| **Post condiciones** | El sistema notifica sobre el error al usuario y no realiza el conteo de líneas físicas y lógicas del archivo. | | |

*Tabla 12 - Resultado del caso de prueba 2.6*

| **Número de ejecución** | 1 | |
| --- | --- | --- |
| **Estatus (Pasó / No Pasó)** | Pasó | |
| **Responsable de la ejecucuión** | Fernando Joachín Prieto | |
| **Fecha de ejecución** | 23/02/2025 | |
| **Salida** | Error: Line 3 in file InvalidFile.java has incorrect brace style. | |
| **Faltas detectadas** | | |
| **Descripción falta** | | **Severidad (Alta, Media, Baja)** |
| N/A | | |

### 

## 2.7. Validación de importaciones explícitas

*Tabla 13 - Caso de prueba 2.7*

| **Caso de prueba** | Validación de importaciones explícitas | | |
| --- | --- | --- | --- |
| **Identificador caso de prueba** | CP17\_ValidaciónImportacionesExplícitas | | |
| **Función probar** | FileFormatValidator.isValidImportStatement | | |
| **Objetivo** | Verificar que el método isValidImportStatement retorne true con importaciones explícitas. | | |
| **Descripción** | Se prueba el método isValidImportStatement para asegurar que solamente permite importaciones explícitas | | |
| **Criterios de éxito** | El método retorna true solamente con importaciones explícitas | | |
| **Criterios de falla** | El método retorna false para importaciones de comodines (\*) | | |
| **Precondiciones** | N/A | | |
| **Necesidades para el caso de prueba** | Un objeto javaFile con lo siguiente:  Una lista con el contenido del archivo a contar pero solamente con importaciones explícitas.  List<String> explicitImportFile = Arrays.asList(  "import java.util.List;",  "public class Test {",  " public static void main(String[] args) {",  " System.out.println(\"Hello, World!\");",  " }",  "}"  );.  El nombre del archivo con la extensión .java ValidFile.java | | |
| **Autor** | Fernando Joachin Prieto | | |
| **Fecha de creación** | 23/02/2025 | | |
| **Flujo del caso de prueba** | **No paso** | **Usuario del sistema** | **Sistema** |
| **1** | **El usuario llamada al método isValidFileFormat y le pasa el nombre del archivo y la lista del contenido con las líneas a contar** | **El sistema ejecuta isValidFileFormat con los parametros recibidos y llama a isValidImportStatement para validar que solamente haya importaciones explícitas.** |
| **2** |  | **El sistema retorna un true.** |
| **Post condiciones** | El sistema permite el conteo de lineas fisicas y logicas del archivo. | | |

*Tabla 14 - Resultado del caso de prueba 2.7*

| **Número de ejecución** | 1 | |
| --- | --- | --- |
| **Estatus (Pasó / No Pasó)** | Pasó | |
| **Responsable de la ejecucuión** | Fernando Joachín Prieto | |
| **Fecha de ejecución** | 23/02/2025 | |
| **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. Validación de importaciones comodines

*Tabla 15 - Caso de prueba 2.8*

| **Caso de prueba** | Validación de importaciones comodines | | |
| --- | --- | --- | --- |
| **Identificador caso de prueba** | CP18\_ValidaciónImportacionesComodines | | |
| **Función probar** | FileFormatValidator.isValidImportStatement | | |
| **Objetivo** | Verificar que el método isValidImportStatement retorne false con importaciones comodines (\*). | | |
| **Descripción** | Se prueba el método isValidImportStatement para asegurar que no permite importaciones comodines (\*). | | |
| **Criterios de éxito** | El método retorna false con importaciones comodines (\*). | | |
| **Criterios de falla** | El método retorna true para importaciones de comodines (\*). | | |
| **Precondiciones** | N/A | | |
| **Necesidades para el caso de prueba** | Un objeto javaFile con lo siguiente:  Una lista con el contenido del archivo a contar pero con importaciones comodines.  List<String> wildcardImportFile = Arrays.asList(  "import java.util.\*;",  "public class Test {",  " public static void main(String[] args) {",  " System.out.println(\"Hello, World!\");",  " }",  "}"  );  El nombre del archivo con la extensión .java InvalidFile.java | | |
| **Autor** | Fernando Joachin Prieto | | |
| **Fecha de creación** | 23/02/2025 | | |
| **Flujo del caso de prueba** | **No paso** | **Usuario del sistema** | **Sistema** |
| **1** | **El usuario llamada al método isValidFileFormat y le pasa el nombre del archivo y la lista del contenido con las líneas a contar** | **El sistema ejecuta isValidFileFormat con los parametros recibidos y llama a isValidImportStatement para validar que solamente haya importaciones explícitas.** |
| **2** |  | **El sistema retorna un false.** |
|  | **3** |  | **El sistema imprime**  **"Error: Line 1 in file InvalidFile.java contains a wildcard import."** |
| **Post condiciones** | El sistema notifica sobre el error al usuario y no realiza el conteo de líneas físicas y lógicas del archivo. | | |

*Tabla 16 - Resultado del caso de prueba 2.8*

| **Número de ejecución** | 1 | |
| --- | --- | --- |
| **Estatus (Pasó / No Pasó)** | Pasó | |
| **Responsable de la ejecucuión** | Fernando Joachín Prieto | |
| **Fecha de ejecución** | 23/02/2025 | |
| **Salida** | Error: Line 1 in file InvalidFile.java contains a wildcard import. | |
| **Faltas detectadas** | | |
| **Descripción falta** | | **Severidad (Alta, Media, Baja)** |
| N/A | | |

### 

## 2.9. Validación de anotaciones correctas

*Tabla 17 - Caso de prueba 2.9*

| **Caso de prueba** | Validación de anotaciones correctas | | |
| --- | --- | --- | --- |
| **Identificador caso de prueba** | CP19\_ValidaciónAnotacionesCorrectas | | |
| **Función probar** | FileFormatValidator.isValidAnnotationFormat | | |
| **Objetivo** | Verificar que el método isValidAnnotationFormatr retorne true con el formato adecuado de las anotaciones | | |
| **Descripción** | Se prueba el método isValidAnnotationFormatpara asegurar que solamente permite anotaciones con el formato correcto. | | |
| **Criterios de éxito** | El método retorna true solamente con el formato de anotaciones correcto. | | |
| **Criterios de falla** | El método retorna false con el formato de anotaciones correcto. | | |
| **Precondiciones** | N/A | | |
| **Necesidades para el caso de prueba** | Un objeto javaFile con lo siguiente:  Una lista con el contenido del archivo a contar pero con anotaciones que siguen el formato establecido.  List<String> correctAnnotationFile = Arrays.asList(  "@Override",  "public void method() {",  " System.out.println(\"Hello, World!\");",  "}"  );  El nombre del archivo con la extensión .java ValidFile.java | | |
| **Autor** | Fernando Joachin Prieto | | |
| **Fecha de creación** | 23/02/2025 | | |
| **Flujo del caso de prueba** | **No paso** | **Usuario del sistema** | **Sistema** |
| **1** | **El usuario llamada al método isValidFileFormat y le pasa el nombre del archivo y la lista del contenido con las líneas a contar** | **El sistema ejecuta isValidFileFormat con los parametros recibidos y llama a isValidAnnotationFormat para validar que las anotaciones se apeguen al formato establecido.** |
| **2** |  | **El sistema retorna un true.** |
| **Post condiciones** | El sistema permite el conteo de lineas fisicas y logicas del archivo. | | |

*Tabla 18 - Resultado del caso de prueba 2.9*

| **Número de ejecución** | 1 | |
| --- | --- | --- |
| **Estatus (Pasó / No Pasó)** | Pasó | |
| **Responsable de la ejecucuión** | Fernando Joachín Prieto | |
| **Fecha de ejecución** | 23/02/2025 | |
| **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. Validación de anotaciones incorrectas

*Tabla 19 - Caso de prueba 2.10*

| **Caso de prueba** | Validación de anotaciones incorrectas | | |
| --- | --- | --- | --- |
| **Identificador caso de prueba** | CP20\_ValidaciónAnotacionesIncorrectas | | |
| **Función probar** | FileFormatValidator.isValidAnnotationFormat | | |
| **Objetivo** | Verificar que el método isValidAnnotationFormatr retorne false con el formato inadecuado de las anotaciones | | |
| **Descripción** | Se prueba el método isValidAnnotationFormatpara asegurar que no permite anotaciones con el formato incorrecto. | | |
| **Criterios de éxito** | El método retorna false con el formato de anotaciones incorrecto. | | |
| **Criterios de falla** | El método retorna true con el formato de anotaciones incorrecto. | | |
| **Precondiciones** | N/A | | |
| **Necesidades para el caso de prueba** | Un objeto javaFile con lo siguiente:  Una lista con el contenido del archivo a contar pero con anotaciones que no siguen el formato establecido.  List<String> annotationWithDeclarationFile = Arrays.asList(  "@Override public void method() {",  " System.out.println(\"Hello, World!\");",  "}"  );  El nombre del archivo con la extensión .java InvalidFile.java | | |
| **Autor** | Fernando Joachin Prieto | | |
| **Fecha de creación** | 23/02/2025 | | |
| **Flujo del caso de prueba** | **No paso** | **Usuario del sistema** | **Sistema** |
| **1** | **El usuario llamada al método isValidFileFormat y le pasa el nombre del archivo y la lista del contenido con las líneas a contar** | **El sistema ejecuta isValidFileFormat con los parametros recibidos y llama a isValidAnnotationFormat para validar que las anotaciones se apeguen al formato establecido.** |
| **2** |  | **El sistema retorna un true.** |
|  | **3** |  | **El sistema imprime**  **“Error: Line 1 in file" InvalidFile.java has incorrect annotation formatting”.** |
| **Post condiciones** | El sistema notifica sobre el error al usuario y no realiza el conteo de líneas físicas y lógicas del archivo. | | |

*Tabla 20 - Resultado del caso de prueba 2.10*

| **Número de ejecución** | 1 | |
| --- | --- | --- |
| **Estatus (Pasó / No Pasó)** | Pasó | |
| **Responsable de la ejecucuión** | Fernando Joachín Prieto | |
| **Fecha de ejecución** | 23/02/2025 | |
| **Salida** | Error: Line 1 in file InvalidFile.java has incorrect annotation formatting. | |
| **Faltas detectadas** | | |
| **Descripción falta** | | **Severidad (Alta, Media, Baja)** |
| N/A | | |