



**UADY**  
FACULTAD DE  
MATEMÁTICAS

**Universidad Autónoma de Yucatán**  
Facultad de Matemáticas  
Licenciatura en Ingeniería de Software

**Mantenimiento de Software**  
M.I.T. Carlos Benito Mojica Ruiz

**Casos de Prueba**  
Versión 1.2

**Integrantes de Equipo:**

Canul Ordoñez, Josué Israel  
Garcilazo Cuevas, Mónica  
Leo Fernández, José Carlos  
Pool Flores, Endrick Alfredo  
Rodríguez Coral, Samuel David

## CONTROL DE DOCUMENTACIÓN

Título:	Casos de Prueba para el Proyecto “Contador Azul”
Referencia:	Canul, J. I., Garcilazo, M., Leo, J. C., Pool, E. A., & Rodríguez, S. D. (2025, febrero 24). <i>Casos de prueba para el proyecto “Contador Azul”</i>
Autor:	Canul, J. I., Garcilazo, M., Leo, J. C., Pool, E. A., & Rodríguez, S. D.
Fecha:	24 de febrero del 2025

## HISTÓRICO DE VERSIONES

Versión	Fecha	Estado	Responsable	Nombre de archivo
1.0	24/02/2025	A	Samuel David Rodríguez Coral	Casos de Prueba (V1.0).docx
1.1	26/02/2025	A	Samuel David Rodríguez Coral	Casos de Prueba (V1.1).docx
1.2	01/03/25	A	Samuel David Rodríguez Coral	Casos de Prueba (V1.2).docx

## HISTÓRICO DE CAMBIOS

Versión	Fecha	Cambios
1.0	24/02/2025	Definición de los casos de prueba.
1.1	26/02/2025	Reporte de las pruebas realizadas.
1.2	01/03/25	Actualización de pruebas.

# Índice

<b>Caso de Prueba 1.....</b>	<b>3</b>
Estándar de Conteo: Identificación de líneas que terminan en paréntesis.....	3
<b>Caso de Prueba 2.....</b>	<b>4</b>
Estándar de Conteo: Identificación de líneas con corchete abierto como único elemento “{” .....	4
<b>Caso de Prueba 3.....</b>	<b>5</b>
Estándar de Conteo: Identificación de líneas con multi-instancia.....	5
<b>Caso de Prueba 4.....</b>	<b>6</b>
Estándar de Conteo: Identificación líneas donde no se respeta la estructura de cierre de llaves.....	6
<b>Caso de Prueba 5.....</b>	<b>7</b>
Prueba Unitaria: Funcionamiento de CodeLineAnalyzer.....	7
<b>Caso de Prueba 6.....</b>	<b>8</b>
Prueba Unitaria: Funcionamiento de ProjectAnalyzer.....	8
<b>Caso de Prueba 7.....</b>	<b>9</b>
Prueba de Integración: Funcionamiento de CodeProcessor.....	9
<b>Reporte de Pruebas.....</b>	<b>10</b>

# Caso de Prueba 1

## **Estándar de Conteo: Identificación de líneas que terminan en paréntesis**

Identificador caso de prueba: CP-001

Autor: Samuel David Rodriguez Coral

Fecha de creación: [24/02/25]

Función por probar: Detección de líneas inválidas durante el procesamiento de archivos

Objetivo: Asegurar que el programa durante la lectura de archivos identifica líneas no válidas debido a que terminan con un paréntesis.

Descripción: Cuando el programa se encuentra leyendo un archivo .java y encuentra una línea que finaliza con un paréntesis el programa se detendrá y lanzará una excepción.

Criterios de éxito: El programa es detenido por una excepción `InvalidLineFormatException` en caso de existir alguna línea que no se considere válida, en caso contrario se espera que el programa concluya indicando la cantidad de líneas físicas y lógicas contenidas en cada archivo procesado.

Criterios de falla: El programa concluye su funcionamiento sin haber lanzado una excepción en un código que contiene líneas inválidas.

Post condiciones: El programa tuvo la reacción esperada según si el contenido de alguno de los archivos leídos tenía líneas válidas o no.

# Caso de Prueba 2

## **Estándar de Conteo: Identificación de líneas con corchete abierto como único elemento “{“**

Identificador caso de prueba: CP-002

Autor: Samuel David Rodriguez Coral

Fecha de creación: [24/02/25]

Función por probar: Detección de líneas inválidas durante el procesamiento de archivos

Objetivo: Asegurar que el programa durante la lectura de archivos identifica líneas no válidas y debido a que solo contienen un corchete abierto” { ” como único elemento

Descripción: Cuando el programa se encuentra leyendo un archivo .java y encuentra una línea que solo contiene un corchete abierto como único elemento “ { “ el programa se detendrá y lanzará una excepción.

Criterios de éxito: El programa es detenido por una excepción InvalidLineFormatException en caso de existir alguna línea que no se considere válida, en caso contrario se espera que el programa concluya indicando la cantidad de líneas físicas y lógicas contenidas en cada archivo procesado.

Criterios de falla: El programa concluye su funcionamiento sin haber lanzado una excepción en un código que contiene líneas inválidas.

Post condiciones: El programa tuvo la reacción esperada según si el contenido de alguno de los archivos leídos tenía líneas válidas o no.

# Caso de Prueba 3

## **Estándar de Conteo: Identificación de líneas con multi-instancia**

Identificador caso de prueba: CP-003

Autor: Samuel David Rodriguez Coral

Fecha de creación: [24/02/25]

Función por probar: Detección de líneas inválidas durante el procesamiento de archivos

Objetivo: Asegurar que el programa durante la lectura de archivos identifica líneas no válidas debido a la declaración o instancia de múltiples variables en una sola línea.

Descripción: Cuando el programa se encuentra leyendo un archivo .java y se encuentre en una misma línea instanciación o declaración de múltiples variables el programa se detendrá y lanzará una excepción.

Criterios de éxito: El programa es detenido por una excepción InvalidLineFormatException en caso de existir alguna línea que no se considere válida, en caso contrario se espera que el programa concluya indicando la cantidad de líneas físicas y lógicas contenidas en cada archivo procesado.

Criterios de falla: El programa concluye su funcionamiento sin haber lanzado una excepción en un código que contiene líneas inválidas.

Post condiciones: El programa tuvo la reacción esperada según si el contenido de alguno de los archivos leídos tenía líneas válidas o no.

# Caso de Prueba 4

## **Estándar de Conteo: Identificación líneas donde no se respeta la estructura de cierre de llaves**

Identificador caso de prueba: CP-004

Autor: Samuel David Rodriguez Coral

Fecha de creación: [24/02/25]

Función por probar: Detección de líneas inválidas durante el procesamiento de archivos

Objetivo: Asegurar que el programa durante la lectura de archivos identifica líneas no válidas debido a que no se respeta la estructura de cierre de llaves

Descripción: Cuando el programa se encuentra leyendo un archivo .java y se encuentre con alguna línea que donde no se respete la estructura de cierre de llaves el programa se detendrá y lanzará una excepción.

Criterios de éxito: El programa es detenido por una excepción InvalidLineFormatException en caso de existir alguna línea que no se considere válida, en caso contrario se espera que el programa concluya indicando la cantidad de líneas físicas y lógicas contenidas en cada archivo procesado.

Criterios de falla: El programa concluye su funcionamiento sin haber lanzado una excepción en un código que contiene líneas inválidas.

Post condiciones: El programa tuvo la reacción esperada según si el contenido de alguno de los archivos leídos tenía líneas válidas o no.

# Caso de Prueba 5

## **Prueba Unitaria: Funcionamiento de CodeLineAnalyzer**

Identificador caso de prueba: CP-005

Autor: Samuel David Rodriguez Coral

Fecha de creación: [24/02/25]

Función por probar: Correcta clasificación y conteo de líneas lógicas y físicas durante la lectura del código.

Objetivo: Asegurar que el programa durante la lectura de archivos identifica y contabiliza de manera correcta las líneas lógicas y físicas, apegándose a lo establecido en el estándar de conteo.

Descripción: Cuando el programa se encuentra leyendo un archivo .java y se encuentre con alguna línea la clasificará como lógica o física según lo definido en el estándar de conteo.

Criterios de éxito: El programa concluye indicando la cantidad de líneas físicas y lógicas contenidas en cada archivo procesado.

Criterios de falla: El programa concluye su funcionamiento al haber lanzado una excepción o la respuesta es incorrecta.

Post condiciones: El programa tuvo la reacción esperada según si el contenido de alguno de los archivos leídos tenía líneas válidas o no.



# Caso de Prueba 6

## **Prueba Unitaria: Funcionamiento de ProjectAnalyzer**

Identificador caso de prueba: CP-006

Autor: Samuel David Rodriguez Coral

Fecha de creación: [24/02/25]

Función por probar: Correcta detección de los archivos que el programa puede procesar

Objetivo: Asegurar que el programa durante la lectura de archivos únicamente lee los archivos cuya extensión es .java

Descripción: Cuando el programa se encuentra leyendo un archivo .java y se encuentre con alguna línea la clasificará correctamente como lógica o física y contabilizará su aparición. En caso de que se le presente un archivo con una extensión diferente a .java dicho archivo no será procesado por el programa y será omitido.

Criterios de éxito: El programa concluye indicando la cantidad de líneas físicas y lógicas contenidas en cada archivo procesado. En caso de pasarle un archivo con una extensión distinta, el programa omite dicho archivo.

Criterios de falla: El programa realiza la lectura de una archivo cuya extensión es distinta a .java

Post condiciones: El programa concluye enseñando la cantidad de lineas fisicas y logicas identificadas en el archivo leído, en caso de haberle presentado un archivo con una extensión diferente el programa no entrega nada.

# Caso de Prueba 7

## **Prueba de Integración: Funcionamiento de CodeProcessor**

Identificador caso de prueba: CP-007

Autor: Samuel David Rodriguez Coral

Fecha de creación: [24/02/25]

Función por probar: Funcionamiento correcto de la clase CodeProcessor al estar integrado con CodeLineAnalyzer

Objetivo: Asegurar que el programa durante la lectura de archivos identifica y contabiliza de manera correcta las líneas lógicas y físicas.

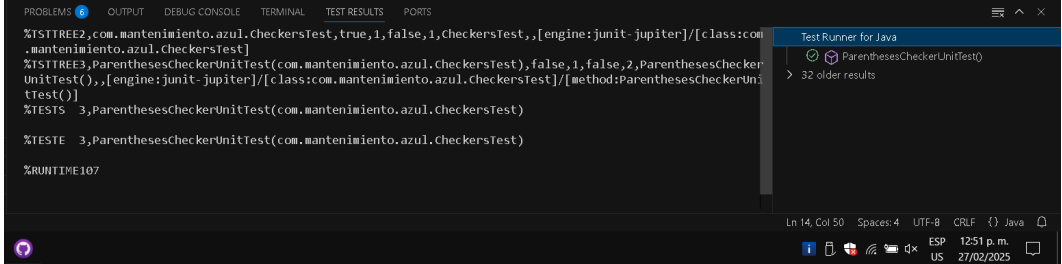
Descripción: Cuando el programa se encuentra leyendo un archivo .java y se encuentre con alguna línea la clasificará correctamente como lógica o física y contabilizará su aparición.

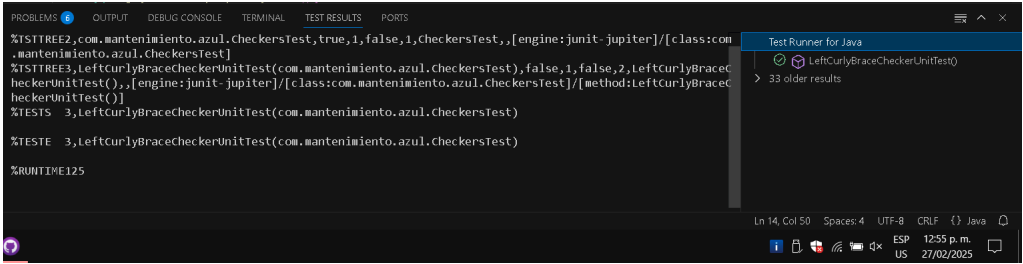
Criterios de éxito: El programa concluye indicando la cantidad de líneas físicas y lógicas contenidas en cada archivo procesado.

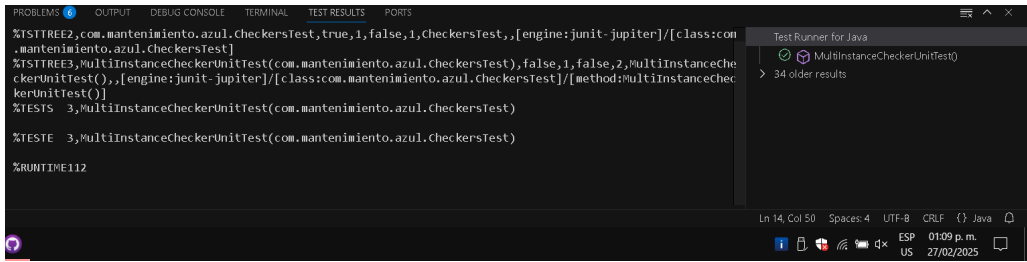
Criterios de falla: El programa concluye su funcionamiento al haber lanzado una excepción o la respuesta es incorrecta.

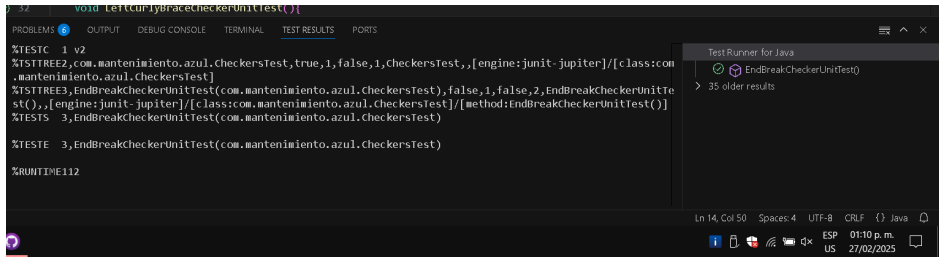
Post condiciones: El programa tuvo la reacción esperada según si el contenido de alguno de los archivos leídos tenía líneas válidas o no.


# Reporte de Pruebas


Proyecto: Contador Azul	Versión: 1.0
Tipo de prueba: Unitaria	Autor de la prueba: Samuel David Rodriguez Coral
Como se probará: Se usa CodeProcessor y se le hace procesar un archivo java el cual contiene las líneas que se espera active la excepción correspondiente al caso de prueba	
Ubicación de la prueba: Github del proyecto, rama de Test	
Requisitos para la prueba: Maven instalado	
Clase donde se hace la prueba: CheckerTest.java	
Resultados de la prueba: Exitoso	
Evidencia:	
	
Fecha de la prueba: 27/02/25	

Proyecto: Contador Azul	Versión: 1.0
Tipo de prueba: Unitaria	Autor de la prueba: Samuel David Rodriguez Coral
Como se probará: Se usa CodeProcessor y se le hace procesar un archivo java el cual contiene las líneas que se espera active la excepción correspondiente al caso de prueba	
Ubicación de la prueba: Github del proyecto, rama de Test	
Requisitos para la prueba: Maven instalado	
Clase donde se hace la prueba: CheckerTest.java	
Resultados de la prueba: Exitoso	
Evidencia:	
	
Fecha de la prueba: 27/02/25	

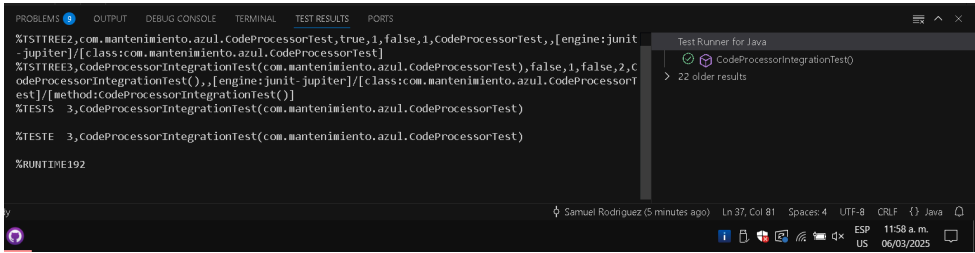
Proyecto: Contador Azul	Versión: 1.0
Tipo de prueba: Unitaria	Autor de la prueba: Samuel David Rodriguez Coral
Caso de prueba a evaluar: CP-03	
Como se probará: Se usa CodeProcessor y se le hace procesar un archivo java el cual contiene las líneas que se espera active la excepción correspondiente al caso de prueba	
Ubicación de la prueba: Github del proyecto, rama de Test	
Requisitos para la prueba: Maven instalado	
Clase donde se hace la prueba: CheckerTest.java	
Resultados de la prueba: Exitoso	
Evidencia:	
	
Fecha de la prueba: 27/02/25	

Proyecto: Contador Azul	Versión: 1.0
Tipo de prueba: Unitaria	Autor de la prueba: Samuel David Rodriguez Coral
Caso de prueba a evaluar: CP-04	
Como se probará: Se usa CodeProcessor y se le hace procesar un archivo java el cual contiene las líneas que se espera active la excepción correspondiente al caso de prueba	
Ubicación de la prueba: Github del proyecto, rama de Test	
Requisitos para la prueba: Maven instalado	
Clase donde se hace la prueba: CheckerTest.java	
Resultados de la prueba: Exitoso	
Evidencia:	
	
Fecha de la prueba: 27/02/25	

Proyecto: Contador Azul	Versión: 1.1
Tipo de prueba: Unitaria	Autor de la prueba: Samuel David Rodriguez Coral
Caso de prueba a evaluar: CP-05	
Como se probará: Se usa CodeLineAnalyzer y se le da como parámetro un Path el cual corresponde a la dirección del archivo RegularFile.java de la cual se hizo el conteo manual de lineas fisicas y logicas y se compara los resultados esperados con los resultados obtenidos al usar el método analyzeFile de la clase CodeLineAnalyzer, esperando que ambos resultados coinciden.	
Ubicación de la prueba: Github del proyecto, rama de Test	
Requisitos para la prueba: Maven instalado	
Clase donde se hace la prueba: CodeLineAnalyzerTest.java	
Resultados de la prueba: Exitoso	
Evidencia:	
	
Fecha de la prueba: 05/03/25	

Proyecto: Contador Azul	Versión: 1.0
Tipo de prueba: Unitaria	Autor de la prueba: Samuel David Rodriguez Coral
Caso de prueba a evaluar: CP-06	
Como se probará: Se usa ProjectAnalyzer y se le pasa la direccion de el archivo WrongFile.py, luego se hará uso del método analyzeProject de la clase ProjectAnalyzer, se espera que el programa retorne una lista vacía, ya que solo analiza archivos que con la extensión .java.	
Ubicación de la prueba: Github del proyecto, rama de Test	
Requisitos para la prueba: Maven instalado	
Clase donde se hace la prueba: ProjectAnalyzerTest.java	
Resultados de la prueba: Exitoso	
Evidencia:	
	
Fecha de la prueba: 27/02/25	



Proyecto: Contador Azul	Versión: 1.1
Tipo de prueba: Integración	Autor de la prueba: Samuel David Rodriguez Coral
Caso de prueba a evaluar: CP-07	
<p>Como se probará: Se usa CodeProcessor y se instancian los parametros que el metodo processFile solicita, se espera que al darle leer un archivo de extension .java este sea procesado sin problemas, y que el conteo de lineas fisicas y logicas coincida con el conteo manual hecho para el archivo RegularFile.java el cual fue usado como sujeto de pruebas.</p>	
Ubicación de la prueba: Github del proyecto, rama de Test	
Requisitos para la prueba: Maven instalado	
Clase donde se hace la prueba: CodeProcessorTest.java	
Resultados de la prueba: Exitoso	
Evidencia:	
	
Fecha de la prueba: 05/03/25	