LINT REPORT

Repositorio:
https://github.com/marsamber/Acme-Toolkits

Miembros (Grupo E3.05):
Gonzalo Martínez Martínez (gonmarmar5@alum.us.es)
Álvaro Miranda Pozo (alvmirpoz@alum.us.es)
Pedro Parrilla Bascón (pedparbas@alum.us.es)
Marta Sampedro Bernal (marsamber@alum.us.es)
Álvaro Vázquez Ortiz (alvvazort@alum.us.es)
María Castro Bonilla (marcasbon@alum.us.es)

Índice

Resumen Ejecutivo

Con este documento tratamos de mostrar los bad smells que encontramos mediante el análisis con SonarLint. Usaremos una tabla plantilla que rellenaremos indicando la información del bad smell y si se llevó alguna acción a cabo.

Historial de versiones

Versión	Fecha	Descripción
1.0	22-05-2022	Creación del documento
1.1	22-05-2022	Correcciones y avances
1.2	23-05-2022	Correcciones del follow up y finalización del documento

Introducción

El uso de SonarLint se organiza mediante una tabla de la siguiente forma:

- Será de una forma estructurada en tabla para que se vea de forma más visual e intuitiva.
- En la primera columna tendremos el ID del bad smell.
- En la segunda columna encontramos una breve descripción.
- Tras esto, en la tercera columna se muestra si la corrección propuesta por SonaLint se lleva a cabo o no.
- En la última columna adjuntamos las fotos de código que veamos necesaria, como por ejemplo el antes y después

Contenido

Title	Imports innecesarios deberían borrarse (java:S1128)
Descripción	Las importaciones de un archivo deberían ser manejadas por el Entorno de Desarrollo Integrado (IDE), no manualmente por el desarrollador. Las importaciones no utilizadas e inútiles no deberían aparecer si ese es el caso. Dejarlas reduce la legibilidad del código, ya que su presencia puede resultar confusa.
Corrección (S/N)	Sí
Fotos(no necesaria)	

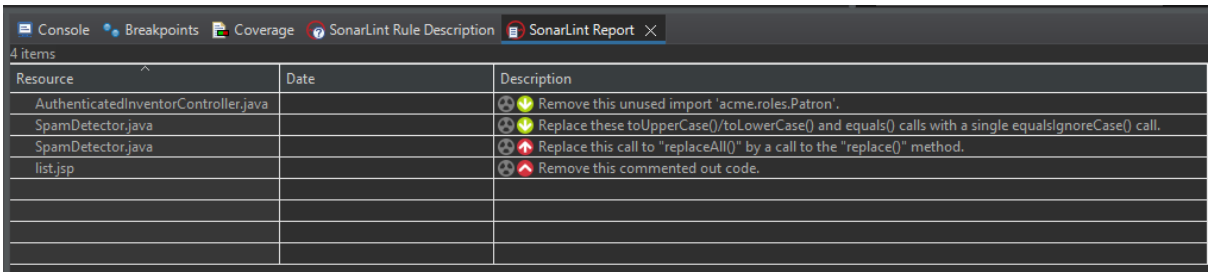
Title	Las comparaciones de cadenas insensibles a mayúsculas y minúsculas deben realizarse sin conversores a mayúsculas ni minúsculas intermedios (java:S1157)
Descripción	El uso de toLowerCase() o toUpperCase() para realizar comparaciones sin distinción de mayúsculas y minúsculas es ineficiente porque requiere la creación de objetos String temporales e intermedios.
Corrección (S/N)	Sí
Fotos(no necesaria)	

Title	Debería preferirse "String#replace" a "String#replaceAll" (java:S5361)
Descripción	La implementación subyacente de String::replaceAll llama al método java.util.regex.Pattern.compile() cada vez que se llama, incluso si el primer argumento no es una expresión regular. Esto tiene un coste de rendimiento significativo y, por lo tanto, debe utilizarse con cuidado. Cuando se utiliza String::replaceAll, el primer argumento debe ser una expresión regular real. Si no es el caso, String::replace hace exactamente lo mismo que String::replaceAll sin el inconveniente de rendimiento de la expresión regular. Esta regla plantea un problema para cada String::replaceAll utilizado con una cadena como primer parámetro que no contenga un carácter o patrón regex especial.
Corrección (S/N)	Sí
Fotos(no necesaria)	

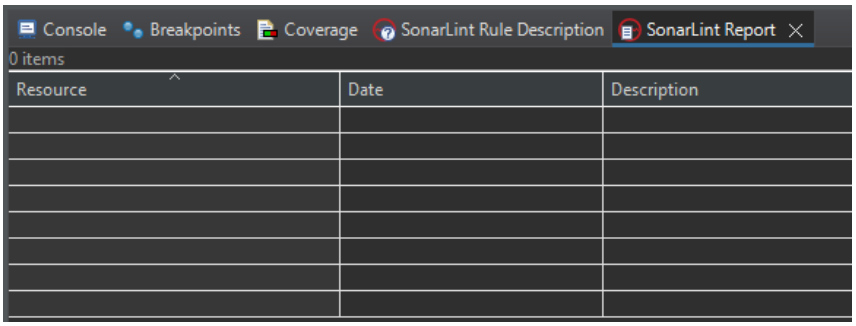
Title	Las secciones de código no deben comentarse (Web:AvoidCommentedOutCodeCheck)
Descripción	Los programadores no deben comentar el código, ya que esto abulta los programas y reduce la legibilidad. El código no utilizado debe eliminarse y puede recuperarse del historial de control de código fuente si es necesario.
Corrección (S/N)	Sí
Fotos(no necesaria)	

Fotos

SonaLint antes:



SonarLint después:



Conclusión

Los problemas de Sonalint los solventamos rápidamente ya que eran fallos tontos que no tardamos en resolver.

Bibliografía

Intencionadamente en blanco.