INFORME PRÁCTICA 5B: MÉTRICAS DE CALIDAD

1. INTRODUCCIÓN

En esta práctica se pretende verificar la calidad de varios productos software haciendo uso de SonarQube. Con esto analizaremos datos como código clonado, bugs o vulnerabilidades y el esfuerzo necesario para su resolución.

2. ANÁLISIS SUPERFICIAL

-> Versión inicial

Número total de incidencias: 106

- Ordenadas por tipo: 2 bug y 104 code smell
- Ordenadas por severidad: 14 high, 57 medium y 35 low

Tiempo total para corregir el proyecto: 7 h y 32 min

Principales métricas de complejidad:

- Complejidad ciclomática: 71
- Complejidad cognitiva: 74

-> Versión refactorizada

Número total de incidencias: 86

- Ordenadas por tipo: 2 bug y 84 code smell
- Ordenadas por severidad: 5 high, 53 medium y 28 low

Tiempo total para corregir el proyecto: 5h y 9 min

Principales métricas de complejidad:

- Complejidad ciclomática: 56
- Complejidad cognitiva: 41

3. ANÁLISIS PROFUNDO DEL CODIGO REFACTORIZADO

Para empezar, analizamos el código y vemos que en la clase tienda hay un 8,4% de duplicación de código.

Además, observamos que la mayor parte de las incidencias de mantenimiento se encuentran en los test, principalmente en el test VendedorEnPlantillaTest. El tiempo para corregir estas clases es de 1h y 28min para la clase VendedorEnPlantillaTest y 32 min para VendedorEnPracticas.

Analizando las incidencias de mantenimiento del main vemos que la clase con un mayor número de incidencias es la clase Tienda. Y el tiempo para corregir esta clase es de 1h y 10 min.

Ahora analizaremos las incidencias por severidad:

La incidencia con severidad alta que más se repite es: “Change this "try" to a try-with-resources”, se encuentra en la clase Tienda y tiene un tiempo de resolución de 15 min.

Las incidencias con severidad media que más se repiten son:

- “Rename "nombre" which hides the field declared at line 22”, en la clase Tienda y con un tiempo de resolución de 5 min.

- “Change the visibility of this constructor to "protected"”, en las clases Vendedor y VendedorEnPlantilla, con un tiempo de resolución de 2 min.

- “Swap these 2 arguments so they are in the correct order: expected value, actual value” en las clases VendedorEnPlantillaTest y VendedorEnPracticasTest, con un tiempo de resolucion de 2 min.

- “Use assertEquals instead” en las clases VendedorEnPlantillaTest y VendedorEnPracticasTest, con un tiempo de resolucion de 2 min.

- “Use assertNotEquals instead” en las clases VendedorEnPlantillaTest y VendedorEnPracticasTest, con un tiempo de resolucion de 2 min

Las incidencias con severidad baja que más se repiten son:

- “Rename this package name to match the regular expression '^[a-z_] + (\. [a-z_] [a-z0-9_] *) *$'” con un tiempo de resolucion de 10 min.
- “Replace the type specification in this constructor call with the diamond operator ("<>")” con un tiempo de resolucion de 1 min.
- “Remove this 'public' modifier” con un tiempo de resolución de 2 min.

4. PLAN DE MEJORA DE CALIDAD

Para considerar: el plan de mejora de calidad que realizaremos se basará en corregir las incidencias con errores de severidad alta. En esto es en lo que nos hemos basado para elegir las incidencias a solucionar. Debido a que solucionar todas las incidencias de severidad alta nos suponía un esfuerzo de 60 minutos no hemos tenido que elegir entre ellas.

Índice	Prioridad	Severidad	Descripcion	Tipo	Localización	Effort
1	1	High	Añadir una condición de finalización a un bucle	Bug	Línea 36, Clase GestionComisiones	15 min

2	1	High	Añadir un caso por defecto al switch	Code Smell	Línea 40, Clase GestionComisiones	5 min
3	3	High	Crear una constante en vez de duplicar el mismo “ERROR” literalmente 4 veces	Code Smell	Línea 66, Clase GestionComisiones	10 min
4	2	High	Cambiar un try por un try-with-resources	Code Smell	Línea 113, Clase Tienda	15 min
5	2	High	Cambiar un try por un try-with-resources	Code Smell	Línea 173, Clase Tienda	15 min
						60 min

Ahora describiremos como hemos solucionado cada una de las incidencias (identificadas por su índice):

1. Añadir una condición de finalización a un bucle.

Para solucionar esto hemos añadido al default del switch una línea en la que se pone a false una variable booleana que hemos creado e inicializado a true y que además hemos convertido en el condicional del while.

2. Añadir un caso por defecto al switch.

Para solucionar esto inicialmente habíamos creado un default con un System.out.println pero observamos que seguía manteniéndose la incidencia, aunque de severidad media y por lo tanto para eliminarlas completamente hemos declarado un logger que realice en el default del switch un logger.warning().

3. Crear una constante en vez de duplicar el mismo “ERROR” literalmente 4 veces

Para solucionar esto hemos creado una constante `private static final String ERROR = "ERROR"` para sustituir los “ERROR” por ERROR.

4. Cambiar un try por un try-with-resources

Para solucionar esto hemos cambiado el try y su contenido, hemos cambiado la declaración del *PrintWritter out* a la declaración del try de la siguiente manera:

try (PrintWriter out = new PrintWriter(new FileWriter(datos))) {

…}

5. Cambiar un try por un try-with-resources

Para solucionar esto hemos cambiado el try y su contenido, hemos cambiado la declaración del *Scanner in* a la declaración del try de la siguiente manera:

try (Scanner in = new Scanner(new FileReader(datos))) {

...}

5. RESOLUCION DE INCIDENCIAS DE CALIDAD

A la hora de ejecutar los cambios propuestos en el plan de mejora de calidad, podemos afirmar que se han realizado todos los cambios propuestos con éxito.

Cada una de las incidencias se ha resuelto tal y como se describe en el apartado anterior.

6. ANÁLISIS DE MEJORA DE LA CALIDAD OBTENIDA

-> Versión final después de solucionar incidencias

Número total de incidencias: 76

- Ordenadas por tipo: 1 bug y 75 code smell
- Ordenadas por severidad: 0 high, 53 medium y 23 low

Tiempo total para corregir el proyecto: 4 h y 4 min

Principales métricas de complejidad:

- Complejidad ciclomática: 64
- Complejidad cognitiva: 39