



INFORME PRÁCTICA 5B: CALIDAD DE PRODUCTO

1. Introducción

Este informe tiene como objetivo analizar y corregir la calidad del código desarrollado en la anterior práctica. Se revisaron aspectos como los criterios de calidad externa e interna en todo el código con el fin de identificar posibles mejoras. A partir del análisis, se aplicaron correcciones diseñando y haciendo uso de un plan de mejora cuyo tiempo de ejecución sea menor de una hora.

2. Análisis superficial de la mejora obtenida con la refactorización

Inicial:

Clase	Incidencias	Calidad	CleanCode	Tiempo	Nº
saldoInsuficienteException	Nombre de clase comienza por minúscula	Mantenibilidad	Consistencia	5 min	1
datoErroneoException	Nombre de clase comienza por minúscula	Mantenibilidad	Consistencia	5 min	1
CuentaValores	Reemplaza especificación <TipoVariable> por <>	Mantenibilidad	Intencionalidad	1 min	1
Tarjeta	Declarar variables en líneas diferentes	Mantenibilidad	Consistencia	2 min	1
	Cambia el constructor de public a protected	Mantenibilidad	Intencionalidad	2 min	1
Valor	Añade un override a hashCode() (ya que hay uno a equals)	Fiabilidad	Intencionalidad	15 min	1
	Equals() debería verificar el tipo del argumento	Fiabilidad	Intencionalidad	5 min	1
	Se puede lanzar un NullPointerException en equals()	Fiabilidad	Intencionalidad	10 min	1
Movimiento	Añade un override a hashCode() (ya que hay uno a equals)	Fiabilidad	Intencionalidad	15 min	1
	Equals() debería verificar el tipo del argumento	Fiabilidad	Intencionalidad	5 min	1
	Se puede lanzar un NullPointerException en equals()	Fiabilidad	Intencionalidad	10 min	1
CuentaAhorro	Variable "Movimientos" debería comenzar por minúscula	Mantenibilidad	Consistencia	2 min	1
	Reemplaza especificación <TipoVariable> por <>	Mantenibilidad	Intencionalidad	1 min	1
	Cast innecesario a	Mantenibilidad	Intencionalidad	5 min	1



	Movimiento				
Credito	La variable "credito" no debería tener el mismo nombre que su clase	Mantenibilidad	Consistencia	10 min	1
	Variable "MovimientosMensuales" debería comenzar por minúscula	Mantenibilidad	Consistencia	2 min	1
	Cast innecesario a Movimiento	Mantenibilidad	Intencionalidad	5 min	2
CuentaValoresTest	Quita public de los métodos	Mantenibilidad	Intencionalidad	2 min	4
	Usa assertEquals en vez de assertTrue	Mantenibilidad	Intencionalidad	2 min	4
Cliente	Cambia las variables públicas a privadas o constantes	Mantenibilidad	Adaptabilidad	10 min	6
	Variable "Cuentas" debería comenzar por minúscula	Mantenibilidad	Consistencia	2 min	1
	Reemplaza especificación <TipoVariable> por <>	Mantenibilidad	Intencionalidad	1 min	2
CuentaAhorroTest	Quita public de los métodos	Mantenibilidad	Intencionalidad	2 min	10
	Argumentos introducidos en assertEquals deberían ser escritos en el orden correcto (valor esperado, valor recibido)	Mantenibilidad	Intencionalidad	2 min	8
	Usa assertEquals en vez de assertTrue	Mantenibilidad	Intencionalidad	2 min	17
	Borra o rellena bloques de catch vacíos	Mantenibilidad	Intencionalidad	5 min	6
Total				285 min	76

Refactorizado:

Clase	Incidencias	Calidad	CleanCode	Tiempo	Nº
saldoInsuficienteException	Nombre de clase comienza por minúscula	Mantenibilidad	Consistencia	5 min	1
datoErroneoException	Nombre de clase comienza por minúscula	Mantenibilidad	Consistencia	5 min	1
ValidacionCantidades	Añadir un constructor privado, con el fin de hacerla una clase de utilidad	Mantenibilidad	Intencionalidad	5 min	1



CuentaValores	Reemplaza especificación <TipoVariable> por <>	Mantenibilidad	Intencionalidad	1 min	1
Cuenta	Cambia el constructor de public a protected	Mantenibilidad	Intencionalidad	2 min	1
Tarjeta	Declarar variables en líneas diferentes	Mantenibilidad	Consistencia	2 min	1
	Cambia el constructor de public a protected	Mantenibilidad	Intencionalidad	2 min	1
Debito	Borra fechaCaducidad (sin utilizar)	Mantenibilidad	Intencionalidad	5 min	1
	Declara fechaCaducidad como variable local en los métodos que la utilizan	Mantenibilidad	Intencionalidad	5 min	1
Cliente	Reemplaza especificación <TipoVariable> por <>	Mantenibilidad	Intencionalidad	1 min	2
ValidacionCantidadesTest	Quita public de los métodos	Mantenibilidad	Intencionalidad	2 min	3
Valor	Añade un override a hashCode() (ya que hay uno a equals)	Fiabilidad	Intencionalidad	15 min	1
	Equals() debería verificar el tipo del argumento	Fiabilidad	Intencionalidad	5 min	1
	Se puede lanzar un NullPointerException en equals()	Fiabilidad	Intencionalidad	10 min	1
Movimiento	Añade un override a hashCode() (ya que hay uno a equals)	Fiabilidad	Intencionalidad	15 min	1
	Equals() debería verificar el tipo del argumento	Fiabilidad	Intencionalidad	5 min	1
	Se puede lanzar un NullPointerException en equals()	Fiabilidad	Intencionalidad	10 min	1
CuentaAhorro	Borra import no utilizado java.time.LocalDate	Mantenibilidad	Intencionalidad	1 min	1
	Variable "Movimientos" debería comenzar por minúscula	Mantenibilidad	Consistencia	2 min	1
	Reemplaza especificación <TipoVariable> por <>	Mantenibilidad	Intencionalidad	1 min	1
	Cast innecesario a Movimiento	Mantenibilidad	Intencionalidad	5 min	1
Credito	La variable "credito" no debería tener el mismo nombre que su clase	Mantenibilidad	Consistencia	10 min	1
	Borra fechaCaducidad (sin utilizar)	Mantenibilidad	Intencionalidad	5 min	1
	Declara fechaCaducidad	Mantenibilidad	Intencionalidad	5 min	1



	como variable local en los métodos que la utilizan				
	Cast innecesario a Movimiento	Mantenibilidad	Intencionalidad	5 min	1
CuentaValoresTest	Borra import no utilizado java.beans.Transient	Mantenibilidad	Intencionalidad	1 min	1
	Quita public de los métodos	Mantenibilidad	Intencionalidad	2 min	5
	Usa assertEquals en vez de assertTrue	Mantenibilidad	Intencionalidad	2 min	6
CuentaAhorroTest	Borra import no utilizado org.junit.jupiter.api.Assertions.assertNull	Mantenibilidad	Intencionalidad	1 min	1
	Borra import no utilizado java.time.LocalDate	Mantenibilidad	Intencionalidad	1 min	1
	Quita public de los métodos	Mantenibilidad	Intencionalidad	2 min	9
	Argumentos introducidos en assertEquals deberían ser escritos en el orden correcto (valor esperado, valor recibido)	Mantenibilidad	Intencionalidad	2 min	7
	Usa assertEquals en vez de assertTrue	Mantenibilidad	Intencionalidad	2 min	17
	Borra o rellena bloques de catch vacíos	Mantenibilidad	Intencionalidad	5 min	6
Total				255 min	81

Comparación de número de incidencias por atributo de calidad/CleanCode:

	Inicial	Refactorizado
Mantenibilidad	70	75
Fiabilidad	6	6
Usabilidad	0	0
Consistencia	7	5
Intencionalidad	63	76
Adaptabilidad	6	0
Responsabilidad	0	0

Comparación principales métricas de complejidad:

	Inicial		Refactorizado	
Métricas	WMC	CCog	WMC	CCog
Cliente	16	9	12	1



Credito	16	8	14	3
Cuenta	2	0	2	0
CuentaAhorro	20	7	9	1
CuentaValores	5	3	7	4
datoErroneoException	1	0	1	0
Debito	8	2	9	0
Movimiento	9	1	10	1
saldoInsuficienteException	1	0	1	0
Tarjeta	1	0	1	0
ValidacionCantidades	-	-	4	2
Valor	8	1	8	1

3. Análisis profundo del proyecto refactorizado

Los principales problemas de calidad del producto suponen mucho de la implementación de las pruebas y el método de uso de AssertTrue en vez de un Assert especial para el caso de uso haciendo fácilmente casi la mitad de los errores encontrados errores en el desarrollo de las pruebas como la definición public de los mismo y alguno del orden de los argumentos. Siendo los problemas relacionados con las pruebas un total de 56 con un tiempo requerido estimado de 2 horas y 7 minutos para la corrección de estos.

Search for rules...

(Java) JUnit assertTrue/assertFals... 23
(Java) JUnit5 test classes and met... 17
(Java) Assertion arguments should ... 7
(Java) Nested blocks of code shoul... 6

El error dentro de las pruebas que requiere más tiempo para su corrección es remover código relacionado con try catch de 5 minutos. Demostrando ser errores relativamente sencillos de corregir.

src/test/java/es/unican/is2/CuentaAhorroTest.java

☐ Swap these 2 arguments so they are in the correct order: expected value, actual value. Intentionality
Maintainability junit tests ... +
Open Not assigned L37 • 2min effort • 3 days ago

☐ Swap these 2 arguments so they are in the correct order: expected value, actual value. Intentionality
Maintainability junit tests ... +
Open Not assigned L38 • 2min effort • 3 days ago

☐ Use assertEquals instead. Intentionality
Maintainability junit tests +
Open Not assigned L48 • 2min effort • 3 days ago



src/test/java/es/unican/is2/CuentaAhorroTest.java

☐ [Remove this 'public' modifier.](#) Intentionality
Maintainability junit tests +
Fixed Not assigned 2min effort • 3 days ago

☐ [Remove this 'public' modifier.](#) Intentionality
Maintainability junit tests +
Fixed Not assigned 2min effort • 3 days ago

Por otro lado, el código principal cuenta con 25 errores de los cuales 8 son de mediana severidad y los 17 restantes de baja severidad, siendo los más numerosos de intencionalidad 7 mientras uno solo de consistencia. Con un tiempo total requerido de 2 horas 8 minutos.

☐ Bulk Change Select issues Navigate to issue 4 issues 4min effort

src/main/java/es/unican/is2/Cliente.java

☐ [Replace the type specification in this constructor call with the diamond operator \("<>"\).](#) Intentionality
Maintainability clumsy +
Fixed Not assigned 1min effort • 3 days ago

Se obtienen que entre los diversos criterios la mantenibilidad es la mayor con 19 casos, 6 medianos y 13 leves. La regla más rota por parte es la del uso innecesario de tipo al inicializar un constructor. Que se cuenta dentro de 4 casos dos dentro de cliente y una en CuentaAhorro y otra en CuentaValores con un tiempo total de 4 minutos. Mientras que las más importante es la de renombrar el campo credito con un tiempo estimado de 10 minutos.

☐ Bulk Change Select issues Navigate to issue 6 issues 29min effort

src/main/java/es/unican/is2/Credito.java

☐ [Rename field "credito"](#) Consistency
Maintainability brain-overload +
Fixed Not assigned 10min effort • 3 days ago

Mientras tanto el criterio de fiabilidad cuenta con 6 problemas entre las clases Movimiento y Valor teniendo las dos los mismos problemas con el mismo tiempo entre ambas sumando 1 hora.



☐ Bulk Change

Select issues Navigate to issue 6 issues 1h effort

src/main/java/es/unican/is2/Movimiento.java

☐ This class overrides "equals()" and should therefore also override "hashCode()".

Intentionality

Reliability

cwe cert +

☐ Open ☐ Not assigned

L42 • 15min effort • 3 days ago

☐ Add a type test to this method.

Intentionality

Reliability

No tags +

☐ Open ☐ Not assigned

L42 • 5min effort • 3 days ago

☐ A "NullPointerException" could be thrown; "other" is nullable here.

Intentionality

Reliability

cwe symbolic-execution ... +

☐ Open ☐ Not assigned

L44 • 10min effort • 3 days ago

4. Plan de mejora

Las del código principalmente deben de centrarse en el código principal, ya que este los errores de estos afectan más el funcionamiento del proyecto mientras que los problemas de los test tienden a rondar el uso correcto de la herramienta, lo cual puede considerarse de menos importancia, ya que estos funcionan, pero no son necesariamente óptimos. El criterio de priorización ha sido la mayor severidad seguido por el tiempo requerido, si los tiempos son iguales y el error es el mismo o muy parecido se intenta tomar como una categoría y se suman los tiempos.

En el código principal deberá de mejorarse en la clase Valor y la clase Movimiento la implementación del método aquel ya que podría darse el caso de que el objeto dado sea nulo provocando errores e incluso la detención del programa. Cada uno de los metodos tardara 10 minutos según SonarQube, juntos son 20 minutos.

Seguido está el uso equivocado de la variable nombrada crédito en la clase con el mismo nombre. Esto puede llegar a causar mal entendidos sobre el uso de la variable causando problemas de mantenimiento. Esta resolución tardara 10 minutos según SonarQube.

Remover la variable fechaCaducidad de la clase Debito y Credito, así mismo como los métodos que la utilizan, esto también requiere de una revisión al resto de las clases como Usuario. El tiempo estimado es de 5 minutos en cada clase, con un total de 10 minutos.

En la clase utilitaria ValidacionCantidades no se ha establecido un método constructor para evitar su incorrecta instanciación. Tiempo requerido 5 minutos.



Las clases abstractas no deberían tener un constructor publico ya que este debería de estar disponible solo para clases hijas por lo que se cambia la clase Tarjeta y la clase Cuenta. El tiempo requerido es de 2 minutos individualmente. Siendo 4 minutos por la tarea en conjunto.

Todos los errores anteriores siendo de severidad media, se debería de poder resolver en un lapso de 49 minutos. Luego los errores de poca severidad del código principal se ignoran los relacionados con las clases hash debido a la insuficiencia de tiempo para poder centrarnos en cambiar los nombres de las excepciones para que estas estén dentro del estándar. Siendo dos clases debería de tomar 5 minutos cada una, 10 minutos en total. Y para el minuto faltante se puede optimizar algún import innecesario como eliminar el import de LocalDate de CuentaAhorro.

Tiempo total estimado: 60 minutos

Prioridad	Clase/Archivo	Incidencia	Severidad	Tiempo (min)	Tiempo Acumulado (min)
1	Valor y Movimiento	Validación de null en equals()	Alta	10 cada clase	20
2	Credito	Variable "credito" con mismo nombre que la clase	Media	10	30
3	Debito y Credito	Eliminar fechaCaducidad (variable sin uso)	Media	5 cada clase	40
4	ValidacionCantidades	Añadir constructor privado (clase utilitaria)	Media	5	45
5	Tarjeta y Cuenta	Cambiar constructor de public a protected	Media	2 cada clase	49
Reserva	-	Ajustes finos (ej: imports innecesarios)	-	1	50
Baja Prioridad					
6	saldoInsuficienteException y datoErroneoException	Nombres de clases en minúscula	Baja	5 cada clase	60

5. Resolución de incidencias

Se han corregido todas las incidencias indicadas anteriormente en el plan de acción.



- Para la de mayor prioridad, se estableció según el ejemplo dado por sonarQube que si el objeto recibido es null que retorne falso.
- Para la de la variable de Credito, buscamos en el código todas las referencias a dicha variable, y nos aseguramos de que estaban todas en la misma clase, por lo que cambiamos el nombre a cantidadCredito en esas tres instancias.
- Para la eliminación de los campos fechaCaducidad, se empezó eliminando la propia variable, luego revisando los usos de esta que es el establecerFechaCaducidad. Se revisó que esta no editara nada más y se eliminó. Seguidamente se revisaron los archivos con referencia a esa función y al confirmar que las llamadas a esta función no realizan ningún cambio se eliminaron también para confirmar el funcionamiento correcto de la clase Cliente en este caso.
- Para ValidacionCantidades, añadimos un constructor privado, sin inicializar nada en este, con el único fin de evitar la instanciación.
- En las clases Tarjeta y Cuenta, simplemente cambiamos la declaración de los constructores a protected, sin necesidad de más cambios.
- Por último, arreglamos las declaraciones de saldoInsuficienteException y datoErroneoException, buscando todas las referencias a dichos nombres, además de los nombres de los archivos .java, y cambiándolos a SaldoInsuficienteException y DatoErroneoException, respectivamente.
- Borramos los importes innecesarios esparcidos por todo el código con el tiempo restante.

6. Análisis superficial final

Clase	Incidencias	Calidad	CleanCode	Tiempo	Nº
Valor	Añade un override a hashCode() (ya que hay uno a equals)	Fiabilidad	Intencionalidad	15 min	1
Movimiento	Añade un override a hashCode() (ya que hay uno a equals)	Fiabilidad	Intencionalidad	15 min	1
CuentaValoresTest	Usa assertEquals en vez de assertTrue	Mantenibilidad	Intencionalidad	2 min	6
CuentaAhorroTest	Argumentos introducidos en assertEquals deberían ser escritos en el orden correcto (valor esperado,	Mantenibilidad	Intencionalidad	2 min	7



	valor recibido)				
	Usa assertEquals en vez de assertTrue	Mantenibilidad	Intencionalidad	2 min	17
	Borra o rellena bloques de catch vacíos	Mantenibilidad	Intencionalidad	5 min	6
Total				120 min	38

Comparación de número de incidencias por atributo de calidad/CleanCode:

	Refactorizado	Mejorado
Mantenibilidad	75	36
Fiabilidad	6	2
Usabilidad	0	0
Consistencia	5	0
Intencionalidad	76	38
Adaptabilidad	0	0
Responsabilidad	0	0

	Refactorizado		Mejorado	
Métricas	WMC	CCog	WMC	CCog
Cliente	12	1	12	1
Credito	14	3	13	3
Cuenta	2	0	2	0
CuentaAhorro	9	1	9	1
CuentaValores	7	4	7	4
datoErroneoException	1	0	1	0
Debito	9	0	8	0
Movimiento	10	1	12	3
saldoInsuficienteException	1	0	1	0
Tarjeta	1	0	1	0
ValidacionCantidades	4	2	5	2
Valor	8	1	10	3

Se puede observar un decremento drástico del número de incidencias de calidad que posee el código. Como era de esperar, la mayoría de las incidencias resueltas fueron las basadas en Mantenibilidad e Intencionalidad, debido a que son el mayor número de incidencias que hay en el código. Por otro lado, las métricas no tienen un cambio muy notable, más que algunas empeoran ligeramente con el fin de mejorar la calidad del código. En resumen, podemos afirmar que el plan de mejora proporciona los resultados esperados de este, sobre todo gracias a que ha habido muchas más mejoras que empeoramientos en el código.