IKER MARTÍNEZ GÓMEZ Y NICHOLAS LOUGHRAN CREE

INFORME PRÁCTICA 5: PRUEBAS UNITARIAS DE SOFTWARE

1. Introducción

Con esta práctica se pretende aprender a aplicar refactorizaciones como mecanismo de mantenimiento preventivo de un software heredado.

Además usar practicar el cálculo de métricas de calidad de un producto software.

2. Refactorizaciones

a. Cliente

i. Extract class: Direccion.

Utilizando la herramienta de factorización de Eclipse del mismo nombre, se han extraído los atributos calle, zip y localidad y se ha creado una clase nueva para agrupar esas propiedades, ya que no se utilizan por separado.

ii. Introduce Parameter Object: Direccion.

Utilizando la herramienta de factorización de Eclipse del mismo nombre, se han cambiado los parámetros del constructor calle, zip y localidad por uno de la clase Direccion creada anteriormente.

iii. Extract Method: getSaldoTotal(). Utilizando la herramienta de factorización de Eclipse del mismo nombre, se ha extraído parte del código que determina el tipo de cuenta y su valor.

b. Crédito

i. Extract Method: checkDatoErroneo().

Utilizando la herramienta de factorización de Eclipse del mismo nombre, se ha extraído parte del código que determina si el valor es negativo o no.

ii. Replace Magic Number with Symbolic Constant, cambiamos 0.05 por COMISION.

iii. Extract method for movimientos mensuales, se quitan los bucles for de Retirar y pagoEnEstablecimiento, creando un solo método buscaMovimientoMensual().

iv. Rename atributos.

c. CuentaAhorro

i. Rename Fields: mMovimientos, mFechaDeCaducidadTarjetaDebito, mFechaDeCaducidadTarjetaCredito.

Utilizando la herramienta de factorización de Eclipse del mismo nombre, se ha quitado la letra m de los atributos porque no resulta claro el significado de la letra, si la tiene.

ii. Extract Method: checkSaldo.

Utilizando la herramienta de factorización de Eclipse del mismo nombre, se ha extraído parte del código que determina si el saldo disponible es suficiente.

iii. Extract Method: checkIngresarNegativa, checkRetirarNegativa.

Utilizando la herramienta de factorización de Eclipse del mismo nombre, se ha extraído parte del código que determina si el valor es negativo o no, con su correspondiente mensaje si resultaba ser ingreso o retirada.

d. Débito

i. Extract Method: checkSalarioDisponiblel().

Utilizando la herramienta de factorización de Eclipse del mismo nombre, se ha extraído parte del código que determina el valor del salario disponible.

e. Movimiento

i. Rename atributos y rename métodos para que sean más fácilmente identificables.

f. Valor

i. El super() en el método constructor sobra

3. Situación inicial y final del sistema

Diagrama de clases sin refactorizar.

Diagrama de clases tras refactorizar.

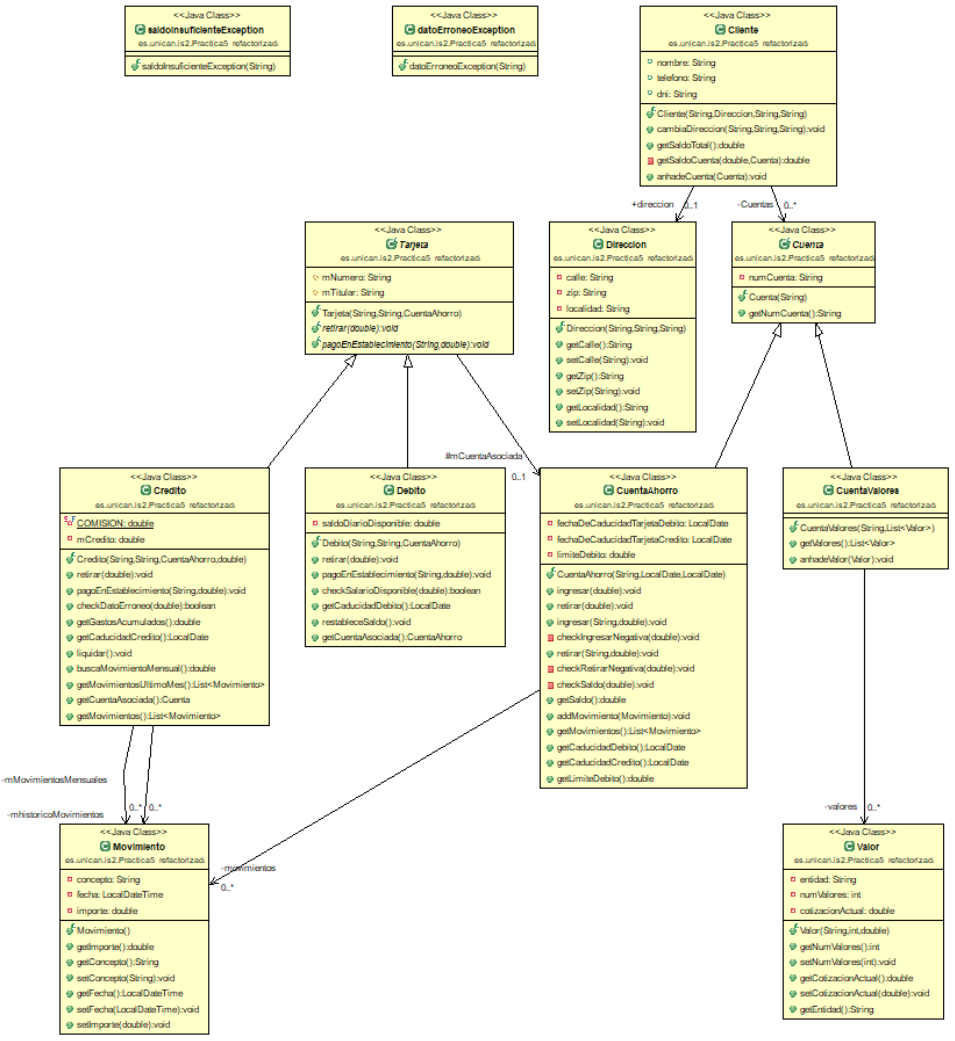

Tabla de valores de las métricas n, WMC, WMCn, DIT y NOC.

	n		WMC		WMCn		DIT		NOC	
Clases	Ini	Fi	Ini	Fi	Ini	Fi	Ini	Fi	Ini	Fi
Cliente	4	5	8	9	2,00	1,80	0	0	0	0
Credito	9	11	16	16	1,78	1,45	1	1	0	0
Cuenta	2	2	2	2	1,00	1,00	0	0	2	2
CuentaAhorro	11	14	18	18	1,64	1,29	1	1	0	0
CuentaValores	3	3	3	3	1,00	1,00	1	1	0	0
Debito	6	7	8	8	1,33	1,14	1	1	0	0
Movimiento	6	6	6	6	1,00	1,00	0	0	0	0
Tarjeta	1	1	1	1	1,00	1,00	0	0	2	2
Valor	6	6	6	6	1,00	1,00	0	0	0	0
Direccion		7		7		1,00		0		0

Tabla de valores de las métricas CBO y CCog.

	CBO		CCog	
Clases	Ini	Fi	Ini	Fi
Cliente	4 Cuenta CuentaAhorro CuentaValores Valor	5 Cuenta CuentaAhorro CuentaValores Valor Direccion	8	5
Credito	6 Movimiento CuentaAhorro saldoInsuficienteException datoErroneoException Tarjeta Cuenta	6 Movimiento CuentaAhorro saldoInsuficienteException datoErroneoException Tarjeta Cuenta	7	5
Cuenta	4 Cliente Credito CuentaAhorro CuentaValores	4 Cliente Credito CuentaAhorro CuentaValores	0	0
CuentaAhorro	8 Cuenta Movimiento datoErroneoException saldoInsuficienteException Cliente Credito Debito Tarjeta	8 Cuenta Movimiento datoErroneoException saldoInsuficienteException Cliente Credito Debito Tarjeta	7	4
CuentaValores	3 Valor Cuenta Cliente	3 Valor Cuenta Cliente	0	0

Debito	4 CuentaAhorro Tarjeta saldoInsuficienteException datoErroneoException	4 CuentaAhorro Tarjeta saldoInsuficienteException datoErroneoException	2	1
Movimiento	2 Credito CuentaAhorro	2 Credito CuentaAhorro	0	0
Tarjeta	5 CuentaAhorro saldoInsuficienteException datoErroneoException Debito Credito	5 CuentaAhorro saldoInsuficienteException datoErroneoException Debito Credito	0	0
Valor	2 CuentaValores Cliente	2 CuentaValores Cliente	0	0
Direccion		1 Cliente		0

4. Análisis de las mejoras en los valores de las métricas

Se intentó utilizar las siguientes refactorizaciones pero se observó que no mejoraron los valores de las métricas como se quería.

- Se intentó juntar los métodos ingresar y retirar (con y sin concepto) en uno solo porque parecían redundantes y se hubiesen observado buenas mejoras en la CCog y en la WMCn. Sin embargo, esta refactorización afectaba al funcionamiento de la aplicación: no diferenciaba entre ingresar y retirar; y no saltaba error si se quería ingresar un valor negativo.
- Se había considerado hacer la refactorización “Pull Up Method”, del método getCuentaAsociada() en las clases Credito y Debito, pero esto incrementa un poco los valores de la métrica WMCn de cada clase. Por esto se decidió no utilizar esta refactorización.

Por otro lado, las siguientes refactorizaciones si tuvieron efectos positivos sobre los valores de las métricas.

a. Cliente

i. El CBO ha incrementado debido a que se ha hecho un "Extract Class" creando la nueva clase Direccion, resultando en un pequeño incremento en el acoplamiento de clases.

ii. Aplicar la herramienta "Extract Method" creando el método getSaldoTotal() ha resultado en una mejora en cuanto a la complejidad ciclomática normalizada (WMCn disminuida de 2 a 1,8) y en cuanto a la complejidad cognitiva (CCog disminuida de 8 a 5), menos compleja en ambos casos. La mejora en WMCn se debe al incremento del número de métodos(de 4 a 5) y el incremento pequeño de el WMC(de 8 a 9).

b. Crédito

i. Aplicar la herramienta "Extract Method" creando los métodos checkDatoErroneo () y buscaMovimientoMensual() ha resultado en una mejora en cuanto a la complejidad ciclomática normalizada (WMCn disminuida de 1,78 a 1,45) y en cuanto a la complejidad cognitiva (CCog disminuida de 7 a 5), menos compleja en ambos casos. La mejora en WMCn se debe al incremento del número de métodos(de 9 a 11).

c. CuentaAhorro

i. Aplicar la herramienta "Extract Method" creando los métodos checkSaldo(), checkIngresarNegativa() y checkRetirarNegativa() ha resultado en una mejora en cuanto a la complejidad ciclomática normalizada (WMCn disminuida de 1,64 a 1,29) y en cuanto a la complejidad cognitiva (CCog disminuida de 7 a 4), menos compleja en ambos casos. La mejora en WMCn se debe al incremento del número de métodos(de 11 a 14).

d. Débito

i. Aplicar la herramienta "Extract Method" creando el método checkSalarioDisponible() ha resultado en una mejora en cuanto a la complejidad ciclomática normalizada (WMCn disminuida de 1,33 a 1,14) y en cuanto a la complejidad cognitiva (CCog disminuida de 2 a 1), menos compleja en ambos casos. La mejora en WMCn se debe al incremento del número de métodos(de 6 a 7).

Se observa que la mayor parte de las mejoras se deben a la refactorización "Extract Method". Esta refactorización disminuye tanto la complejidad ciclomática normalizada, WMCn, y la complejidad cognitiva, CCog. Las otras refactorizaciones utilizadas ayudan a la legibilidad del código y a la comprensión de él, pero no repercuten en las métricas.