PRÁCTICA 5A: REFACTORIZACIONES

1. Objetivos

- Aprender a aplicar refactorizaciones como mecanismo de mantenimiento preventivo de un software heredado.
- En la medida de lo posible, aplicar las refactorizaciones de forma semiautomática utilizando para ello las herramientas que proporciona Eclipse.
- Practicar el cálculo de métricas de calidad de un producto software.

2. Actividades

1. Descargar el código fuente del sistema heredado, disponible en Moodle en el archivo `Practica5.zip`

2. Comprender y analizar la funcionalidad y el código del sistema heredado. Para ello:

a. Leer la siguiente descripción de la aplicación:

La aplicación sirve para gestionar las ventas de una tienda. La aplicación todavía no está completa, por el momento a través de la aplicación sólo se puede añadir una nueva venta a uno de los vendedores de la tienda (identificado por un Id) así como consultar el vendedor (o vendedores) que mayor importe de ventas acumulado tienen en el mes así como el listado de vendedores ordenados por dicho importe de ventas acumulado. Por cada venta realizada los vendedores reciben una comisión, que varía según el tipo de vendedor. Solo los vendedores en plantilla reciben dicha comisión: si es un vendedor Junior la comisión es del 0.5%, mientras que los Senior reciben una comisión del 1%. Los vendedores en prácticas, que todavía no forman parte de la plantilla estable de la tienda, no reciben comisión.

La carga de datos de los vendedores y datos actuales de la tienda se realiza a través de un fichero de texto que se puede asignar a la clase Tienda en su constructor (se proporciona un fichero de ejemplo denominado datosTienda.txt). La aplicación asume que este fichero está en C:\temp, si se quiere almacenar en otro sitio, es necesario asignar el path absoluto adecuado en la clase GestionComisiones.

La interfaz gráfica es muy sencilla y por el momento está implementada usando el paquete fundamentos.

b. Consultar el modelo UML de la aplicación, que se encuentra disponible en Moodle en formar de modelo StarUML.

c. Analizar el propio código.

3. Obtener las métricas WMC, WMCn, CBO, DIT, NOC, CCog y CCogn de todas las clases de la aplicación (excepto de las excepciones si es que las hay).

Nota: El modo de calcular las métricas de complejidad (WMC y CCog) se debe indicar en el propio código con comentarios (como en los problemas del tema 4) para poder evaluar el

proceso de cálculo de dicha métrica. En el caso del CBO se debe proporcionar tanto el valor numérico como una enumeración de las clases que contribuyen a dicho valor (para el cálculo del CBO sí se han de tener en cuenta las excepciones si es que existen).

4. Aplicar un proceso de refactorización a la aplicación. Se pueden aplicar las refactorizaciones vistas en clase de teoría u otras que se encuentren en la bibliografía. Además de las refactorizaciones propiamente dichas, se pueden aplicar otras mejoras que se consideren necesarias para mejorar la calidad del código, como por ejemplo, mejoras dirigidas a la optimización o mejora de rendimiento.

   Nota: Para llevar a cabo este paso crear un proyecto Maven independiente, para así realizar después la entrega de ambos proyectos, cada uno con sus propias métricas. Recordad cambiar el nombre del artifactID en el pom del nuevo proyecto.

   Cuando se realizan tareas de mantenimiento (como la aplicación de refactorizaciones) es muy importante comprobar que los cambios realizados no introducen nuevos errores en el código. Por ello, después de cada refactorización aplicada, se han de ejecutar el conjunto de pruebas de regresión que se hayan establecido. A consecuencia de las refactorizaciones introducidas, puede ser necesario modificar el código de las clases de prueba (no los casos de prueba en sí, sino el modo de implementarlos), o incluso añadir nuevos métodos de prueba si aparecen nuevos métodos en las clases, o incluso nuevas clases de prueba, si aparecen nuevas clases. En esta práctica, será suficiente con mantener la cobertura de las pruebas alcanzada en el código dado (100% de las clases Vendedor, VendedorEnPlantilla y VendedorEnPracticas) en el proyecto refactorizado.

5. Análisis de la mejora obtenida
   a. Realizar el diagrama de clases final una vez aplicadas las refactorizaciones.
   b. Obtener de nuevo las métricas WMC, WMCn, CBO, DIT, NOC, CCog y CCogn.
   c. Analizar y razonar si el cambio en las métricas refleja de manera clara las mejoras realizadas en el código.

3. Criterios de Evaluación y Aclaraciones

Se deberá entregar un fichero comprimido que contenga los siguientes elementos:

- Proyectos Maven completos (sin refactorizar y refactorizado) debidamente exportados. En el código de ambos proyectos deberá estar indicado el modo en que se han calculado las métricas WMC y CCog.
- Archivo jar ejecutable correspondiente al proyecto refactorizado.
- Un informe que describa:
  - La lista de refactorizaciones aplicadas, en orden y con una pequeña explicación de cada una.
  - La situación inicial y final del sistema, incluyendo sus correspondientes diagramas de clases y valores para las métricas.

- El análisis razonado sobre si las diferencias en los valores obtenidos para las métricas reflejan las mejoras de calidad introducidas en el código.

Juan Rivas
Patricia López
Adolfo Garandal