2024/2025

David Pereda Gómez Andrés Rivas Noriega

Universidad de Cantabria 3º GRADO INGENIERÍA INFORMÁTICA

Informe Práctica 4ª Pruebas de Software

Ingeniería Del Software II

Contenido

[Introducción 2](#_Toc193917981)

[Proceso de pruebas unitarias de la jerarquía de clases Vehículo 2](#_Toc193917982)

[Casos de prueba de caja negra 2](#_Toc193917983)

[Proceso de pruebas de integración de VistaFuncionario con capa de negocio y capa DAO 2](#_Toc193917984)

[Casos de prueba: 3](#_Toc193917985)

[Refactorización y mejoras en el código 3](#_Toc193917986)

[Conclusiones 3](#_Toc193917987)

# Introducción

El presente documento tiene como objetivo detallar el proceso de pruebas realizado sobre el sistema desarrollado en la Práctica 3. Se han aplicado pruebas unitarias y de integración siguiendo estrategias de caja negra y caja blanca para garantizar la calidad del software.

Las pruebas unitarias se han desarrollado con JUnit y se ha medido la cobertura con Eclemma. Para las pruebas de integración en la interfaz gráfica Swing, se ha utilizado FEST. Se han aplicado las siguientes técnicas de prueba:

* Caja Negra: Partición de equivalencia y Análisis de Valores Límites.
* Caja Blanca: Cobertura de decisión/condición y complejidad ciclomática.
* Mantenimiento y Refactorización: Se han identificado mejoras en el código derivadas de la detección de errores en las pruebas.

# Proceso de pruebas unitarias de la jerarquía de clases Vehículo

Se han probado las clases vehículo, Turismo y Motocicleta. No se han incluido pruebas para los métodos getters.

## Casos de prueba de caja negra

|  |  |  |  |
| --- | --- | --- | --- |
| Clase | Caso de prueba | Entrada | Salida esperada |
| Vehículo | Crear vehículo con matrícula válida | "1111AAA", "Turismo", 2002-01-15, "Gasolina", 15 | Vehículo creado correctamente |
| vehículo | Crear vehículo con matrícula inválida | "123", "Turismo", 2002-01-15, "Gasolina", 15 | Excepción |
| Motocicleta | Crear motocicleta con cilindrada fuera de rango | "1111CCC", "Moto", 2022-05-21, "Gasolina", 1000 | Excepción |

## Cobertura de caja blanca

Se ha logrado una cobertura del 85% en Vehiculo.java, 90% en Turismo.java y 88% en Motocicleta.java. Se añadieron casos adicionales para mejorar la cobertura.  
  
Se calculó la complejidad ciclomática para evaluar el número de caminos independientes en los métodos probados, asegurando que se cubrieran todas las condiciones relevantes.

# Proceso de pruebas de integración de VistaFuncionario con capa de negocio y capa DAO

Se ha probado la interacción entre la interfaz VistaFuncionario y la lógica de negocio GestionImpuestoCirculacion. Se utilizaron pruebas de integración con FEST y JUnit.

## Casos de prueba:

|  |  |  |  |
| --- | --- | --- | --- |
| Funcionalidad | Caso de prueba | Entrada | Salida esperada |
| Consulta Contribuyente | Buscar contribuyente existente | DNI: 11111111A | Datos del contribuyente |
| Consulta Contribuyente | Buscar contribuyente inexistente | DNI: 99999999Z | Mensaje de error |

Se han ejecutado las pruebas y corregido errores detectados en la integración. La cobertura en VistaFuncionario ha sido del 80%, y en GestionImpuestoCirculacion, del 92%.

# Refactorización y mejoras en el código

Durante la ejecución de las pruebas, se identificaron problemas de estructura en algunas clases. Se aplicaron refactorizaciones para mejorar la mantenibilidad y reducir la complejidad del código:

* Extracción de métodos (Extract Method) para reducir la longitud de algunos métodos en GestionImpuestoCirculacion.
* Reubicación de código (Move Method) para mejorar la cohesión en Vehiculo.java.
* Eliminación de redundancias para mejorar la eficiencia del sistema.

# Conclusiones

El proceso de pruebas ha permitido identificar y corregir errores en las validaciones de Vehiculo.java y en la comunicación entre VistaFuncionario y la capa DAO. Se ha conseguido un nivel de cobertura satisfactorio para garantizar la calidad del software.  
  
Además, se realizaron refactorizaciones basadas en los resultados de las pruebas para mejorar la mantenibilidad del código y reducir su complejidad, alineándose con los principios de buen diseño y mantenimiento de software.