PRÁCTICA 4: PRUEBAS DE SOFTWARE

1. Objetivos

- Aprender a diseñar casos de prueba unitarios aplicando técnicas de caja negra y caja blanca para un sistema software orientado a objetos.
- Aprender a ejecutar pruebas unitarias utilizando un framework de pruebas, en este caso formado por las herramientas *JUnit* y *Eclemma*.
- Aprender a ejecutar pruebas de integración en aplicaciones basadas en interfaces gráficas SWING usando la librería *FEST*.

2. Actividades

1. El alumno atenderá a las explicaciones del profesor, que mostrará en el aula el funcionamiento básico de *JUnit*, *Eclemma* y *FEST*.

2. El alumno deberá llevar a cabo el proceso (parcial) de diseño y ejecución de pruebas unitarias y de integración para el sistema cuya especificación parcial se muestra en el anexo. Para las pruebas de caja negra se debe aplicar criterio de cobertura 1-wise, y para las pruebas de caja blanca cobertura de decisión/condición. Los pasos a seguir para llevar a cabo el proceso son los siguientes:

   2.1) Pruebas unitarias de la clase `Seguro`.

   i. Diseñar casos de prueba para la clase `Seguro` utilizando las técnicas de partición equivalente y AVL.
   ii. Escribir el código de la clase `SeguroTest` utilizando JUnit (para ello basta con escribir el esqueleto de la clase Seguro, sin necesidad de implementarla).
   iii. Completar el código de la clase Seguro.
   iv. Ejecutar la clase de prueba y corregir defectos.
   v. Cuando todas las pruebas de caja negra sean correctas, comprobar la cobertura.
   vi. Diseñar nuevos casos de prueba para alcanzar la cobertura deseada en caso de no haberse alcanzado con las pruebas de caja negra.

   Nota 1: En la implementación se recomienda usar la clase `LocalDate` para el manejo de fechas pues simplifica bastante el desarrollo respecto a la utilización de la tradicional clase `Date` de Java. En el siguiente enlace tenéis un tutorial con el manejo básico de esta clase: http://tutorials.jenkov.com/java-date-time/localdate.html

   2.2) Pruebas de integración de las clases `SegurosGUI` y `Seguro`.

   Diseñar y aplicar pruebas de integración para comprobar el funcionamiento conjunto de la clase `SegurosGUI` con la clase `Seguro`. La clase `SegurosGUI` es una interfaz gráfica muy sencilla que permite realizar el cálculo rápido del precio de un seguro. Su funcionamiento básico se explica en el anexo.

i. Indicar qué casos de prueba se considera necesario aplicar para dicha prueba de integración.
ii. Implementar en JUnit TRES de los casos de prueba indicados utilizando FEST.
iii. Ejecutar los casos de prueba y corregir errores.

2.3) Pruebas unitarias de la clase `ListaOrdenada`.

Para su futura integración con el resto de la aplicación, es necesario implementar el proceso de pruebas unitarias de la clase `ListaOrdenada`.

i. Diseñar casos de prueba de caja negra basándose en la interfaz de la clase, `IListaOrdenada`, que se puede consultar en Moodle.
ii. Implementar los casos de prueba definidos usando JUnit.
iii. Ejecutar dichos casos de prueba utilizando la implementación proporcionada en forma de archivo .jar (`ListaOrdenada.jar`) y detectar posibles errores.
Nota: Para saber cómo usar un jar predefinido desde un proyecto Maven consultar el seminario de Maven (transparencia 15).
iv. Indicar al profesor en qué métodos se encuentran los errores detectados y pedir el código fuente de la clase.
v. Corregir los defectos encontrados.
vi. Ejecutar pruebas de caja blanca comprobando la cobertura alcanzada.
vii. En caso de que la cobertura no sea suficiente, diseñar nuevos casos de prueba.

3. Criterios de Evaluación y Aclaraciones

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

- Proyecto Maven completo, con todas las clases involucradas (tanto de aplicación como de prueba), debidamente exportado.
- El .jar correspondiente a la aplicación completa.
- Un informe que resuma el proceso de pruebas que se ha seguido, identificando las técnicas utilizadas y los casos de prueba definidos. Se proporciona, a través de la plataforma *Moodle*, una plantilla para la elaboración de dicho informe.

Se valorará:

a) El contenido y formato del informe (15%).
b) La corrección del proceso de pruebas de la clase Seguro (40%).
c) La corrección del proceso de pruebas de la clase SegurosGUI (15%).
d) La corrección del proceso de pruebas de la clase ListaOrdenada (30%).

Patricia López Martínez

CASO ESTUDIO

Se trata de una aplicación que gestiona los seguros de coches de una aseguradora. El modelo de implementación parcial de la aplicación es el que se muestra en la Figura 1.

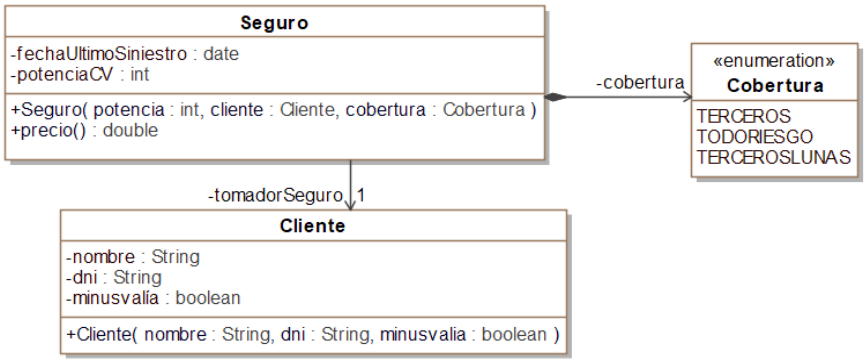

Figura 1. Diagrama de clases parcial de la aplicación de gestión de la aseguradora

El precio del seguro se calcula de acuerdo a las siguientes reglas:

1. Nivel de cobertura contratado => Cada nivel de cobertura tiene un precio base: 1000€ para el seguro a todo riesgo, 600€ para el seguro a terceros + lunas y 400€ para terceros simple.
2. Potencia del coche => En función de la potencia del coche, se aplica un porcentaje de subida al precio base. Los porcentajes son del 5% para coches de 90 a 110 CV (ambos valores incluidos) y del 20% para coches de más de 110CV.
3. Nivel de siniestralidad => Si se ha tenido un siniestro en el último año, se aplica una subida de 200€ al seguro. Si el último siniestro se ha tenido entre uno y tres años atrás se suman 50€ al precio del seguro.
4. Grado minusvalía del tomador del seguro => A las personas con minusvalía se les aplica un descuento del 25% en el precio total del seguro (precio base + subida por potencia + subida por siniestros).

La interfaz gráfica tiene el aspecto que se muestra en la figura 2 y funciona del siguiente modo: en el campo precio se muestra el precio del seguro cuando la ejecución es correcta o los siguientes mensajes de error:

- "¡Dato de entrada erróneo!": Cuando se invoca el método precio con algún dato no válido.
- "Formato de fecha no válido": Cuando el formato de la fecha no se ajusta al patrón dd/mm/yyyy.

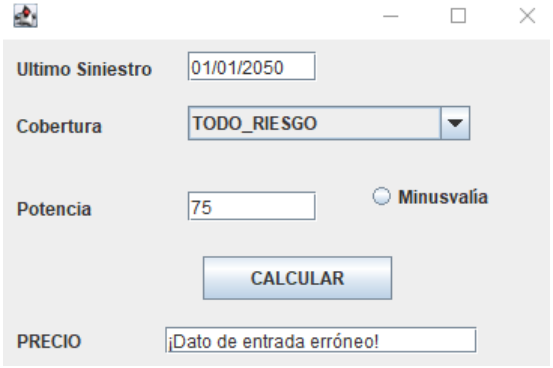

Figura 2. Interfaz gráfica para el cálculo rápido de un seguro