INFORME PRÁCTICA 4: PRUEBAS UNITARIAS DE SOFTWARE

1. Introducción

El objetivo de esta práctica ha sido crear el conjunto de pruebas correspondientes a las clases seguro, cliente, interfaz y lista ordenada. Para realizarlas se han seguido los métodos de caja negra y posteriormente el uso de la cobertura para comprobar que se habían conseguido cubrir todos los casos.

Como se ha mencionado, las fases que se han seguido para realizar la practica han sido las siguientes;

1- Realización de las pruebas de caja negra, a partir de la tabla que se crea para cada una de las clases, se crean las diferentes pruebas para el correcto funcionamiento de esta.
2- Posteriormente, se corrigen los errores que hayan podido surgir en la realización del código.
3- Se ejecutan los tests con cobertura para las pruebas de caja blanca, con esto obtenemos lo que se ha recorrido de los métodos y que métodos se han recorrido(existen métodos que al crear nuevos constructores no se utilizan).
4- Si existe alguna parte del método sin recorrer se corrige y se crea un nuevo test, valido o no, que cubra esa necesidad.

Para llevar a cabo los test correspondientes a caja negra se ha utilizado la instrucción mvn test para correrlos y verificar su funcionamiento. En el caso de la caja blanca, se ha utilizado la herramienta de eclipse de ejecutar con cobertura.

2. Proceso de pruebas unitarias de la clase Seguro

Para las pruebas de caja negra se ha seguido la siguiente tabla

	Clases validas		Clases no validas	
Potencia	1	0<x<90		
	2	90<=x<110	10	<0
	3	x>=110		
Año	4	0<x<365		
	5	365<=x<730	11	<0
	6	x>=730		(>=fecha actual)
		(dias post-contratacion)		
Cobertura	7	TERCEROS		
	8	TERCEROSLUNAS	12	Null
	9	TODORIESGO		

Los valores que se han utlilizado y las combinaciones restantes son las siguientes

	Valores seleccionados
1	70, 89, 90
2	91,100,109, 110
3	111, 120
4	0, 200, 364
5	365, 550, 729
6	730, 745
7	TERCEROS
8	TERCEROSLUNAS
9	TODORIESGO

	Combinaciones creadas
1	70, 745, TERCEROS
2	89, 745, TERCEROSLUNAS
3	90, 730, TODORIESGO
4	91, 729, TERCEROS
5	100, 550, TERCEROSLUNAS
6	109, 365, TODORIESGO
7	110,364,TERCEROS
8	111, 200, TERCEROSLUNAS
9	120,0,TODORIESGO

Tras aplicar la cobertura proporcionada por el eclipse vemos que se han recorrido todas las opciones posibles de los métodos por lo que no hace falta realizar tests extras.

3. Proceso de pruebas de integración de la clase Cliente

Para la caja negra se ha utilizado la siguiente tabla:

	Clases validas		Clases no validas	
Minusvalía	1	True	10	Null
	2	False		
Seguros Contratados	3	>=0	5	Null

Se han utilizado los siguientes valores y combinaciones:

	Valores seleccionados
1	True
2	False
3	1,2

	Combinaciones creadas
1	True,1
2	False,1
3	True,2

Se han creado los test conforme a esto.

Al aplicar la cobertura vemos que el método getseguros no se recorría, por lo que creamos un caso no valido en el que no se añadan seguros y se intente operar con la lista de seguros.

4. Proceso de pruebas de integración de la interfaz

Para realizar las pruebas correspondientes a caja negra de la interfaz se ha creado un test valido en el que se introduce un DNI del xml, y se comprueba que devuelva los datos

correspondientes al mismo y se ha creado un test no valido en el que se introduce un dni erróneo y comprueba que no se ha devuelto ningún dato.

Tras aplicar cobertura desde eclipse vemos que todo VistaAgente esta cubierto por los test realizados antes.

Errores encontrados: En vistaAgente se pide el nombre del cliente en vez del DNI en la casilla donde se pregunta por este último.

5. Proceso de pruebas unitarias de la clase ListaOrdenada

Se han creado los diferentes tests para cada uno de los métodos de la lista ordenada, teniendo en cuenta el funcionamiento de ellos y sus casos.

Errores encontrados:

Al ejecutar los tests que corroboran que se añaden los elementos correctamente observamos que en el método add() no se están almacenando ordenados y simplemente los almacena según entran. Para corregirlo se cambia el "<"del compare to de la siguiente línea por ">", de tal manera que ahora sí que almacenara correctamente los valores.

```
while (indice < lista.size() && elemento.compareTo(lista.get(indice)) > 0)
```

El siguiente error que encontramos está en el método clear(), al ejecutar los test vemos que el tamaño de la lista sigue siendo el mismo y no 0. Para corregirlo cambiamos lista.clone() por el método que la borraría que es lista.clear().

```
lista.clear();
```

Tras crear los test correspondientes a caja negra y corregir los errores, comprobamos la cobertura de caja blanca. Podemos observar que se han comprobado todos los métodos y no es necesario crear ningún método extra más.