Practica 4 Ingeniería de Software II

En este ejercicio nos centraremos en las pruebas de Conformidad y Fiabilidad del programa mediante el uso de pruebas automatizadas del framework de JUnit.

Para comenzar hemos diseñado los casos de prueba de la caja negra, en este caso hemos utilizado la técnica de Partición equivalente y el análisis de valores limites

Columna1	Clase de equivalencia	Casos de prueba valido	Clases de equivalencia invalidas
Potencia	>0	(0,inf)	<0
Cobertura	Terceros	TERCEROS	null
	Terceros + Lunas	TERCEROS_LUNAS	cualquier otro valor
	Todo Riesgo	TODO_RIESGO	
Cliente	Cliente valido		Null

Potencia	Cobertura	Cliente	Valor
100	TERCEROSLUNAS	null	null
-1	TERCEROSLUNAS	cliente	null
0	TODORIESGO	null	null
100	null	cliente	null
100	TODORIESGO	cliente	Se crea el cliente

Para la caja blanca hemos analizado los valores interesantes, que permitan un análisis de los limites de la función precio y así poder comprobar los fallos.

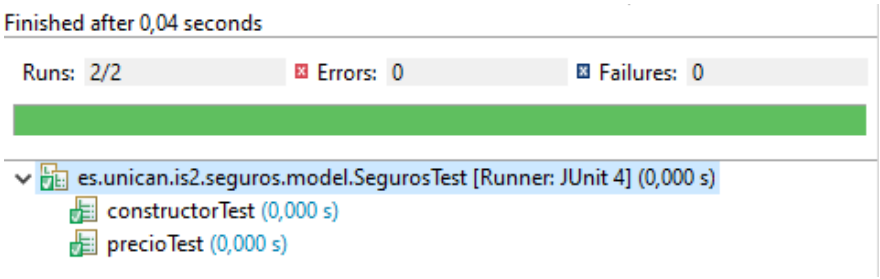
Columna1	Clase de equivalenc	Casos de prueba valic	Clases de equivalencia invalidas	VALOR
Potencia	<90	(0,90)	<0	1
	[90,110]	[90,110]		1,05
	>110	(110,inf)		1,2
Cobertura	terceros	TERCEROS	null	400
	Terceros + Lunas	TERCEROSLUNAS	cualquier otro valor	600
	Todo Riesgo	TODORIESGO		1000
Siniestro	Sin Siniestro	(3,inf)	<0	0
	Entre 1 y 3 años	[1,3]		50
	Hace menos de 1 año	[0,1)		200
Minusvalia	Sin minusvalia	False	null	1
	Con Minusvalia	True	null	0,75

Potencia	Cobertura	Siniestro	Minusvalia	Valor
1	TERCEROSLUNAS	Sin Siniestro	False	600
50	TERCEROSLUNAS	Menos de 1 año	False	800
89	TERCEROS	Menos de 1 año	True	450
90	TERCEROSLUNAS	Menos de 1 año	True	622,5
110	TODORIESGO	Entre 1 y 3 años	False	1100
110	TERCEROS	Menos de 1 año	True	465
111	TERCEROS	Menos de 1 año	True	510
120	TODORIESGO	Entre 1 y 3 años	False	1250

Jose Lesarri Fernandez
Sergio Alegria Ocampo

Gracias a estas pruebas realizadas, nos pudimos dar cuenta de que uno de los if dentro del método que calcula el precio, se había programado como “<=” cuando realmente la especificación era “<” por lo que asignaba un precio incorrecto al seguro.

El finalizar las pruebas de JUnit en relación tanto al constructor como al método que calcula el precio, obtenemos



Lista Ordenada:

En las pruebas de clase ordenada, hemos realizado test para iniciar y crear unas listas, comprobar el tamaño de ella, la posibilidad de añadir elementos y comprobar si el tamaño aumenta y por ultimo, nos dimos cuenta de que el clear lo que hacia realmente es llamar al lista.clone() en vez de hacer lista.clear() por lo que no funcionaba correctamente.

Jose Lesarri Fernandez
Sergio Alegria Ocampo