1- Pruebas unitarias de la clase Seguro: 2 métodos, partición equivalente, AVL.

-Constructor:

-Precio(): según lo que se indica.

2- Implementar la clase SeguroTest con sus métodos. Test Driven Development.

3- Implementar la clase Seguro.

4- Ejecución de la clase de prueba y corrección de fallos. Caja Negra.

5- Si todas las pruebas de caja negra son correctas, se comprueba la cobertura (Caja Blanca).

6- Pruebas de integración de la GUI con la clase Seguro usando Fest.

7- Pruebas unitarias de la clase ListaOrdenada e identificar en qué métodos están los fallos.

8- Corregir esos fallos.

PASO 1:

**Método precio():**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parámetros** | **Clases válidas** | **Valores** | **Clases no válidas** | **Valores** |
| Nivel de cobertura | 1. Todo riesgo.  2. Terceros + lunas.  3. Terceros simple. | Todo riesgo  Terceros + lunas  Terceros simple | 11. !={Todo riesgo, terceros + lunas, terceros simple} | Terceros ampliados |
| Potencia | 4. [90, 110]  5. (110, ∞) | 90  100  110  ------  200 | 12. < 0  13. !Número | -1  AAA |
| Fecha último siniestro | 6. [hoy – 1 año, hoy]  7. [hoy – 3 años, hoy – 1 año)  8. (-∞, hoy – 3 años) | Hoy – 1 año  Hoy – 6 meses  Hoy  ------  Hoy – 3 años  Hoy – 2 años  ------  Hoy – 5 años | 14. > hoy  15. !fecha | Cualquier fecha no válida (fecha con mes 13)  Mañana |
| Grado minusvalía | 9. True.  10. False. | True  False | - | - |

Casos de prueba válidos:

(Todo riesgo, 90, hoy – 1 año, True): 937,5

(Terceros + lunas, 100, hoy – 6 meses, False): 830

(Terceros simple, 110, hoy, True): 465

(Todo riesgo, 200, hoy – 3 años, False): 1250

(Terceros + lunas, 200, hoy – 2 años, False): 770

(Terceros simple, 90, hoy – 5 años, False): 420

Casos de prueba no válidos:

(Terceros ampliados, 110, hoy, True): DatoIncorrectoException

(Todo riesgo, -1, hoy – 1 año, False): DatoIncorrectoException

(Terceros + lunas, AAA, hoy – 6 meses, True): NumberFormatException

(Terceros simple, 200, ’03/13/2021’, False): DatoIncorrectoException

(Todo riesgo, 110, mañana, False): DatoIncorrectoException

**Método Seguro():**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parámetros** | **Clases válidas** | **Valores** | **Clases no válidas** | **Valores** |
| Potencia | 1. [90, 110]  2. (110, ∞) | 90  100  110  ------  200 | 7. < 0  8. !Número | -1  AAA |
| Cliente | 3. No null | - | 9. Null | - |
| Cobertura | 4. Todo riesgo.  5. Terceros + lunas.  6. Terceros simple. | Todo riesgo  Terceros + lunas  Terceros simple | 10. !={Todo riesgo, terceros + lunas, terceros simple} | Terceros ampliados |

Casos de prueba válidos:

(90, No null, Todo riesgo)

(100, No null, Terceros + lunas)

(110, No null, Terceros simple)

(200, No null, Todo riesgo)

Casos de prueba no válidos:

(-1, No null, Todo riesgo): DatoIncorrectoException

(AAA, No null, Todo riesgo): NumberFormatException

(90, Null, Terceros + lunas): NullPointerException

(100,No null, Terceros ampliados): DatoIncorrectoException

**6)**

Realizaremos pruebas de integridad para comprobar el funcionamiento conjunto entre la clase SegurosGUI y Seguros.

Según está indicado, la aplicación debe devolver el precio del seguro según los parámetros seleccionados. Cuando todos los campos sean válidos, devolvemos el precio. Cuando únicamente la fecha sea incorrecta, notificarlo. También, cuando otro de los campos esté mal puesto, mostrarlo por pantalla.

Después de haber verificado mediante caja negra que el código funciona como esperado, nos vale con un caso de prueba válido que devuelva el precio del seguro en la interfaz. Además, nos harán falta dos casos de prueba no válidos para demostrar que se notifica al usuario cuando alguno de los campos está mal puesto.

-Caso de prueba válido:

(TODORIESGO, 90, hoy - 1 año, True): 937,5

-Casos de prueba no válidos:

(TERCEROSLUNAS, -1, hoy - 2 años, False): “¡Dato de entrada incorrecto!”

(TERCEROS, 40, mañana, True): “¡Formato de fecha no válido!”

Para realizar esta operación, lo automatizaremos usando la herramienta FEST. Se precisa especificarlo en el pom.xml como dependencia de la siguiente forma:

<dependency>

<groupId>org.easytesting</groupId>

<artifactId>fest-swing</artifactId>

<version>1.2.1</version>

</dependency>

**7- Pruebas unitarias de la clase ListaOrdenada e identificar en qué métodos están los fallos.**

Caja negra:

Para todos los métodos que se prueban, también hay que tener en cuenta el estado de la lista. Por comodidad lo pondré en este comentario, ya que es para todos igual: lista vacía, lista con un elemento y lista con elementos.

Método get(int índice):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parámetros** | **Clases válidas** | **Valores** | **Clases no válidas** | **Valores** |
| Índice | 1. [0, lista.size() -1] | 0  4  Lista.size() - 1 | 2. < 0  3. > lista.size() | -1  Lista.size() + 1 |

|  |  |
| --- | --- |
| **Casos válidos** | **Casos no válidos** |
| (1, []): 1  (3, [1,2,3,4]): 4  (0, [1]): 1  (2, [1,2,3,4]): 3 | (2, []): IndexOutOfBoundsException  (-1, [1,2,3,4]): IndexOutOfBoundsException  (4, [1,2,3,4]): IndexOutOfBoundsException |

Método add(E elemento):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parámetros** | **Clases válidas** | **Valores** | **Clases no válidas** | **Valores** |
| elemento | 1. !null |  | 2. null |  |

|  |  |
| --- | --- |
| **Casos válidos** | **Casos no válidos** |
| (1, []): [1]  (3, [1,2,3,4]): [1,2,3,3,4]  (1, [2]): [1,2]  (5, [1,2,3,4]): [1,2,3,4,5] | (null, []): NullPointerException  (null, [1]): NullPointerException  (null, [1,2,3,4]): NullPointerException |

Método remove(int índice):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parámetros** | **Clases válidas** | **Valores** | **Clases no válidas** | **Valores** |
| Índice | 1. [0, lista.size() -1] | 0  4  Lista.size() - 1 | 2. < 0  3. > lista.size() | -1  Lista.size() + 1 |

|  |  |
| --- | --- |
| **Casos válidos** | **Casos no válidos** |
| (0, [1]): (1, [])  (3, [1,2,3,4]): (4, [1,2,3])  (2, [1,2,3,4]): (3, [1,2,4]) | (2, []): IndexOutOfBoundsException  (-1, [1,2,3,4]): IndexOutOfBoundsException  (4, [1,2,3,4]): IndexOutOfBoundsException |

Método size():

|  |  |
| --- | --- |
| **Casos válidos** | **Casos no válidos** |
| ([]): 0  ([1]): 1  ([1,2,3,4]): 4 |  |

Método clear():

|  |  |
| --- | --- |
| **Casos válidos** | **Casos no válidos** |
| ([]): ([])  ([1]): ([])  ([1,2,3,4]): ([]) |  |