AeroDescuentos

Se está desarrollando para una aerolínea su módulo de liquidación de tiquetes aéreos. Para el mismo, se tiene una función que aplica descuentos a la tarifa base del vuelo dependiendo del tiempo de antelación de la reserva y la edad del pasajero. Los descuentos SON ACUMULABLES.

*Normativa 005, sobre los descuentos:*

* ***15%*** *de descuento sacando el billete con antelación superior a 20 días.*
* ***5%*** *a los pasajeros con edad inferior a 18 años y* ***8%*** *a los pasajeros con edad superior a 65 años.*

La siguiente es la especificación de la función que se usará en el módulo del cálculo de los descuentos:

/\*\*

calcular la tarifa de cada billete según el trayecto, la antelación en la que se obtiene el billete y la edad del pasajero, de acuerdo con la normativa 005.

@param tarifaBase valor base del vuelo

@param diasAntelacion dias de antelación del vuelo @param edad - edad del pasajero

@throws ExcepcionParametrosInvalidos [XXXXXXXXXXXXXXXXXXXXXXXXXXX]

\*\*/

public long calculoTarifa(long tarifaBase, int diasAntelacion, int edad)

1. De acuerdo con lo indicado, y teniendo en cuenta que NO hay precondiciones, en qué casos se debería arrojar una excepción de tipo ExcepcionParametrosInvalidos?. Agregue esto a la especificación.
   * La excepción debe arrojarse cuando:
     + tarifaBase, diasAntelacion o edad sean valores negativos
     + edad sea 0

/\*\*

calcular la tarifa de cada billete según el trayecto, la antelación en la que se obtiene el billete y la edad del pasajero, de acuerdo con la normativa 005.

@param tarifaBase valor base del vuelo, debe ser mayor o igual que 0

@param diasAntelacion dias de antelación del vuelo, debe ser mayor o igual que 0

@param edad - edad del pasajero, debe ser mayor que 0

@throws ExcepcionParametrosInvalidos [XXXXXXXXXXXXXXXXXXXXXXXXXXX]

\*\*/

public long calculoTarifa(long tarifaBase, int diasAntelacion, int edad)

1. En la siguiente tabla enumere un conjunto de clases de equivalencia que -según usted- creen una buena división del conjunto de datos de entrada de la función anterior:

|  |  |  |
| --- | --- | --- |
| Número | Clase de equivalencia (en lenguaje natural o matemático). | Resultado correcto /  incorrecto. |
| tarifaBase |  | Correcto |
| tarifaBase |  | Incorrecto |
| diasAntelacion |  | Correcto |
| diasAntelacion |  | Correcto |
| diasAntelacion |  | Incorrecto |
| edad |  | Incorrecto |
| edad |  | Correcto |
| edad |  | Correcto |
| edad |  | Correcto |

1. Para cada clase de equivalencia, defina un caso de prueba específico, definiendo: parámetros de entrada y resultados esperados.

@Test

    public void validateNegBase() {

        double result = 0;

        double base = -15;

        int diasAntelación = 15;

        int edad = 50;

        result = CalculadorDescuentos.calculoTarifa(base, diasAntelación, edad);

        Assert.assertNotEquals(15, result, 0);

        Assert.assertFalse(base > 0);

    }

    @Test

    public void validateNegDias() {

        double result = 0;

        double base = 15;

        int diasAntelación = -15;

        int edad = 50;

        result = CalculadorDescuentos.calculoTarifa(base, diasAntelación, edad);

        Assert.assertEquals(15, result, 0);

        Assert.assertFalse(diasAntelación > 0);

    }

    @Test

    public void validateNegEdad() {

        double result = 0;

        double base = 15;

        int diasAntelación = 15;

        int edad = -50;

        result = CalculadorDescuentos.calculoTarifa(base, diasAntelación, edad);

        Assert.assertNotEquals(15, result, 0);

        Assert.assertFalse(edad > 0);

    }

    @Test

    public void validateSuperEdad() {

        double result = 0;

        double base = 15;

        int diasAntelación = 15;

        int edad = 5000;

        result = CalculadorDescuentos.calculoTarifa(base, diasAntelación, edad);

        System.out.println(result);

        Assert.assertEquals(13.8, result, 0);

        Assert.assertTrue(edad > 0);

    }

1. A partir de las clases de equivalencia identificadas en el punto 2, identifique las condiciones límite o de frontera de las mismas.
2. Para cada una de las condiciones de frontera anteriores, defina casos de prueba específicos.

/\*\*

     \* Tests en Frontera

     \*/

    @Test

    public void validateBaseLimite() {

        double result = 0;

        double base = 0;

        int diasAntelación = 15;

        int edad = 50;

        result = CalculadorDescuentos.calculoTarifa(base, diasAntelación, edad);

        Assert.assertEquals(0, result, 0);

        Assert.assertEquals(0, base, 0);

        base++;

        result = CalculadorDescuentos.calculoTarifa(base, diasAntelación, edad);

        Assert.assertEquals(1, result, 0);

        Assert.assertTrue(base > 0);

    }

    @Test

    public void validateDiasLimite() {

        double result = 0;

        double base = 15;

        int diasAntelación = 20;

        int edad = 50;

        result = CalculadorDescuentos.calculoTarifa(base, diasAntelación, edad);

        Assert.assertEquals(15, result, 0);

        Assert.assertEquals(20, diasAntelación, 0);

        diasAntelación++;

        result = CalculadorDescuentos.calculoTarifa(base, diasAntelación, edad);

        Assert.assertEquals(12.75, result, 0);

        Assert.assertTrue(diasAntelación > 20);

    }

    @Test

    public void validateEdadLimite() {

        double result = 0;

        double base = 15;

        int diasAntelación = 15;

        int edad = 17;

        result = CalculadorDescuentos.calculoTarifa(base, diasAntelación, edad);

        Assert.assertEquals(14.25, result, 0);

        Assert.assertTrue(edad < 18);

        edad++;

        result = CalculadorDescuentos.calculoTarifa(base, diasAntelación, edad);

        Assert.assertEquals(15, result, 0);

        Assert.assertEquals(18, edad);

        edad = 65;

        result = CalculadorDescuentos.calculoTarifa(base, diasAntelación, edad);

        Assert.assertEquals(15, result, 0);

        Assert.assertEquals(65, edad);

        edad++;

        result = CalculadorDescuentos.calculoTarifa(base, diasAntelación, edad);

        Assert.assertEquals(13.8, result, 0);

        Assert.assertTrue(edad > 65);

    }