

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. Debe ser un valor positivo mayor que cero.
 * @param diasAntelacion dias de antelación del vuelo. Debe ser un valor positivo mayor que cero.
 * @param edad - edad del pasajero. Debe ser un valor positivo mayor que cero y menor que 135.
 * @return La tarifa calculada con los descuentos aplicados.
 * @throws ExcepcionParametrosInvalidos Si alguno de los parámetros (tarifaBase, diasAntelacion, edad) es
 * negativo o cero.
 */
public long calculoTarifa(long tarifaBase, int diasAntelacion, int edad) throws
ExcepcionParametrosInvalidos;
```

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.
-> Editado en la especificación
2. 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.
1	$\text{diasAntelacion} < 20, 18 < \text{edad} < 65$	Sin descuento = Correcto
2	$\text{diasAntelacion} > 20, 18 < \text{edad} < 65$	Con 15% Descuento = Correcto
3	$\text{diasAntelacion} > 20, 0 < \text{edad} < 18$	Con 20% Descuento = Correcto
4	$\text{diasAntelacion} > 20, \text{edad} > 65$	Con 23% Descuento = Correcto
5	$\text{diasAntelacion} < 20, 0 < \text{edad} < 18$	Con 5% Descuento = Correcto
6	$\text{diasAntelacion} < 20, 0 < \text{edad} > 65$	Con 8% Descuento = Correcto
7	$\text{diasAntelacion} < 0$	Incorrecto
6	$\text{TarifaB} < 0$	Incorrecto
7	$\text{edad} < 0 \parallel \text{edad} > 135$	Incorrecto

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

```
public class TarifasTest {  
    new *  
    @Test  
    public void testSinDescuento() {  
        double tarifaBase = 100000;  
        int diasAntelacion = 15;  
        int edad = 30;  
  
        double tarifaFinal = CalculadorDescuentos.calcularTarifa(tarifaBase, diasAntelacion, edad);  
  
        Assert.assertEquals(tarifaBase, tarifaFinal, delta: 0.01); // No hay descuento  
    }  
    new *  
    @Test  
    public void testDescuento15Porcentaje() {  
        double tarifaBase = 100000;  
        int diasAntelacion = 25;  
        int edad = 30;  
  
        double tarifaFinal = CalculadorDescuentos.calcularTarifa(tarifaBase, diasAntelacion, edad);  
        double tarifaEsperada = tarifaBase * 0.85;  
  
        Assert.assertEquals(tarifaEsperada, tarifaFinal, delta: 0.01);  
    }  
}
```

```
@Test  
public void testDescuento20Porcentaje() {  
    double tarifaBase = 100000;  
    int diasAntelacion = 25;  
    int edad = 15;  
  
    double tarifaFinal = CalculadorDescuentos.calcularTarifa(tarifaBase, diasAntelacion, edad);  
    double tarifaEsperada = tarifaBase * 0.80;  
  
    Assert.assertEquals(tarifaEsperada, tarifaFinal, delta: 0.01);  
}  
  
new *  
@Test  
public void testDescuento23Porcentaje() {  
    double tarifaBase = 100000;  
    int diasAntelacion = 25;  
    int edad = 67;  
  
    double tarifaFinal = CalculadorDescuentos.calcularTarifa(tarifaBase, diasAntelacion, edad);  
    double tarifaEsperada = tarifaBase * 0.77;  
  
    Assert.assertEquals(tarifaEsperada, tarifaFinal, delta: 0.01);  
}
```

```

@Test
public void testDescuento5Porciento() {
    double tarifaBase = 100000;
    int diasAntelacion = 10;
    int edad = 16;

    double tarifaFinal = CalculadorDescuentos.calcularTarifa(tarifaBase, diasAntelacion, edad);
    double tarifaEsperada = tarifaBase * 0.95; // Descuento del 5%

    Assert.assertEquals(tarifaEsperada, tarifaFinal, delta: 0.01);
}

new *
@Test
public void testDescuento8Porciento() {
    double tarifaBase = 100000;
    int diasAntelacion = 15;
    int edad = 70;

    double tarifaFinal = CalculadorDescuentos.calcularTarifa(tarifaBase, diasAntelacion, edad);
    double tarifaEsperada = tarifaBase * 0.92; // Descuento del 8%

    Assert.assertEquals(tarifaEsperada, tarifaFinal, delta: 0.01);
}

```

```

@Test
public void testDescuento8PorcientoConEdadLimite() {
    double tarifaBase = 100000;
    int diasAntelacion = 10;
    int edad = 66;

    double tarifaFinal = CalculadorDescuentos.calcularTarifa(tarifaBase, diasAntelacion, edad);
    double tarifaEsperada = tarifaBase * 0.92; // Descuento del 8%

    Assert.assertEquals(tarifaEsperada, tarifaFinal, delta: 0.01);
}

```

```

/*
@Test
public void testTarifaBaseNegativa() {
    double tarifaBase = -100000;
    int diasAntelacion = 15;
    int edad = 30;

    try {
        CalculadorDescuentos.calcularTarifa(tarifaBase, diasAntelacion, edad);
    } catch (ExcepcionParametrosInvalidos e) {
        Assert.assertEquals(e.getMessage(), ExcepcionParametrosInvalidos.TARIFA_BASE);
    }
}

@Test
public void testDiasNegativos() {
    double tarifaBase = 100000;
    int diasAntelacion = -15;
    int edad = 30;
    try {
        CalculadorDescuentos.calcularTarifa(tarifaBase, diasAntelacion, edad);
    } catch (ExcepcionParametrosInvalidos e) {
        Assert.assertEquals(e.getMessage(), ExcepcionParametrosInvalidos.DIAS_NEGATIVOS);
    }
}
}

```

```

@Test
public void testDiasNegativos() {
    double tarifaBase = 100000;
    int diasAntelacion = 15;
    int edad = -30;
    try {
        CalculadorDescuentos.calcularTarifa(tarifaBase, diasAntelacion, edad);
    } catch (ExcepcionParametrosInvalidos e) {
        Assert.assertEquals(e.getMessage(), ExcepcionParametrosInvalidos.EDAD_INVALIDA);
    }
}
}

```

4. A partir de las clases de equivalencia identificadas en el punto 2, identifique las condiciones límite o de frontera de las mismas.
 - Identificamos los valores límites de la edad en 0, 18 y 65
 - Identificamos el valor límite de los días de antelación en 20
 - Identificamos que para la tarifa básica debe ser mayor que 0
5. Para cada una de las condiciones de frontera anteriores, defina casos de prueba específicos.