**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 días 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.

/\*\*

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 – Días de antelación del vuelo

@param edad – Edad del pasajero

@throws excepcionParametrosInvalidos

@exception tarifaBaseInvalida – Se genera si tarifaBase es menor o igual a cero

@exception diasDeAntelacionInvalidos – Se genera si diasAntelacion es menor que cero

@expcetion edadInvalida – Se genera si la edad es menor que cero

/\*\*

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:

|  |  |  |
| --- | --- | --- |
| **Numero** | **Clase de equivalencia (Lenguaje natural o Matemático)** | **Resultado (Correcto/Incorrecto)** |
|  | 0 <= edad < 18 ^ tarifaBase > 0 ^ diasAntelacion => 0 | Correcto |
|  | edad > 18 ^ tarifaBase > 0 ^ diasAntelacion => 0 | Correcto |
|  | edad < 0 v edad > 125 | Incorrecto |
|  | tarifaBase <= 0 | Incorrecto |
|  | diasAntelacion < 0 | Incorrecto |

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

entrada y resultados esperados.

|  |  |  |
| --- | --- | --- |
| **Clase de Equivalencia** | **Parámetros** | **Resultado Esperado** |
|  | calculoTarifa(450, 3, 10) | 427.5 |
|  | calculoTarifa(115, 0, 20) | 115 |
|  | calculoTarifa(5, 2, -20) | edadInvalida |
|  | calculoTarifa(0, 5, 18) | tarifaBaseInvalida |
|  | calculoTarifa(1, -2, 0) | diasAntelacionInvalidos |

1. A partir de las clases de equivalencia identificadas en el punto 2, identifique las condiciones límite o de frontera de las mismas.

|  |  |
| --- | --- |
| **Clase de Equivalencia** | **Límite** |
|  | edad ϵ { 0, 1, 17, 18} ^ tarifaBase ϵ {1, 2} ^ diasAntelacion ϵ {0, 1} |
|  | edad ϵ { 19, 20} ^ tarifaBase ϵ {1, 2} ^ diasAntelacion ϵ {0, 1} |
|  | edad ϵ { -2, -1, 126, 127} |
|  | tarifaBase ϵ {-1, 0} |
|  | diasAntelacion ϵ {-2, -1} |

1. Para cada una de las condiciones de frontera anteriores, defina casos de prueba específicos.

|  |  |  |
| --- | --- | --- |
| **Clase de Equivalencia** | **Condición** | **Prueba** |
|  | edad ϵ { 0, 1, 17, 18} ^ tarifaBase ϵ {1, 2} ^ diasAntelacion ϵ {0, 1} | calculoTarifa(1, 1, 0) |
|  | edad ϵ { 19, 20} ^ tarifaBase ϵ {1, 2} ^ diasAntelacion ϵ {0, 1} | calculoTarifa(2, 0, 20) |
|  | edad ϵ { -2, -1, 126, 127} | calculoTarifa(2, 0,-2) calculoTarifa(1, 1, 127) |
|  | tarifaBase ϵ {-1, 0} | calculoTarifa(-1, 0, 3) |
|  | diasAntelacion ϵ {-2, -1} | calculoTarifa(1, -2, 0) |