Universidad Nacional de General Sarmiento

TP FINAL PROGRA 2

COM 5

Integrantes: Nicolas Marnoni - Ian Bokser

IREP

2. registrarCliente(int dni, String nombre, String telefono)

dni debe ser positivo y único.

nombre no puede ser nulo ni vacío.

telefono no puede ser nulo ni vacío.

3. registrarAeropuerto(String nombre, String pais, String provincia, String direccion)

nombre debe ser único.

nombre, pais, provincia y direccion no deben ser nulos ni vacíos.

4. registrarVueloPublicoNacional

origen y destino deben ser aeropuertos ya registrados.

Tanto origen como destino deben tener pais = "Argentina".

fecha debe ser válida y futura.

precios y cantAsientos deben tener longitud igual a 2.

cantAsientos[0] y cantAsientos[1] deben ser positivos.

tripulantes debe ser mayor o igual a 1.

valorRefrigerio debe ser mayor o igual a 0.

5. registrarVueloPublicoInternacional

origen y destino deben ser aeropuertos ya registrados.

fecha debe ser válida y futura.

precios y cantAsientos deben tener longitud igual a 3.

Los elementos de cantAsientos deben ser positivos.

tripulantes debe ser mayor o igual a 1.

valorRefrigerio debe ser mayor o igual a 0.

Si escalas no está vacío, cada aeropuerto en escalas debe estar registrado.

6. VenderVueloPrivado

origen y destino deben ser aeropuertos registrados.

fecha debe ser válida y futura.

precio debe ser positivo.

dniComprador debe corresponder a un cliente registrado.

Todos los acompaniantes deben tener DNIs positivos.

7. asientosDisponibles

codVuelo debe corresponder a un vuelo registrado.

Cada clave en el mapa debe ser un número de asiento válido (no ocupado).

Los valores deben ser las clases correspondientes ("Turista", "Ejecutivo", "Primera").

8 y 9. venderPasaje

dni debe corresponder a un cliente registrado.

codVuelo debe corresponder a un vuelo registrado.

nroAsiento debe estar disponible para el vuelo.

El asiento debe ser marcado como ocupado si aOcupar = true.

10. consultarVuelosSimilares

origen y destino deben ser aeropuertos registrados.

Fecha debe ser válida.

El rango de búsqueda incluye hasta 7 días después de la fecha indicada.

11. cancelarPasaje

Si se usa la versión 12-A, la cancelación debe realizarse en O(1)O(1)O(1).

El dni debe ser válido y el codVuelo debe estar registrado.

El asiento debe ser liberado.

12. cancelarVuelo

codVuelo debe corresponder a un vuelo registrado.

Los pasajes reprogramados deben cumplir con las restricciones de clase y disponibilidad.

Los pasajes no reprogramados deben ser eliminados.

13. totalRecaudado

destino debe corresponder a un aeropuerto registrado.

El cálculo debe realizarse en O(1)O(1)O(1).

14. detalleDeVuelo

codVuelo debe corresponder a un vuelo registrado.

El detalle debe coincidir con el formato especificado.

Aerolínea

Esta clase gestiona todos los aspectos de la operación de una aerolínea. Permite registrar clientes, aeropuertos y vuelos, tanto públicos como privados. También maneja la venta y cancelación de pasajes, y puede realizar consultas de disponibilidad de asientos y vuelos similares a una fecha determinada.

Stringbuilder

Donde se implementó Stringbuilder en la funcion “detalleDeVuelo” donde se utilizó para para construir el detalle y devolver un String

Iteradores

Se utilizo iteradores en diferentes sectores del código, como por ejemplo utilice Foreach para recorrer los asientos y así poder ponerlos disponibles para que los clientes lo puedan comprar. Tambien lo utilizamos en “consultarVuelosSimilares” para recorrer todos los vuelos para asi poder encontrar otros vuelos similares.

Herencia y Polimorfismo

Se hace la herencia mediante la clase base abstracta Vuelo, de la que heredan las clases VueloNacional, VueloInternacional y VueloPrivado. Estas subclases tienen definidos en Vuelo atributos y métodos comunes, y luego modifican o dezpliegan comportamientos específicos por el tipo de vuelo.

El polimorfismo se extiende a los métodos como el getDetalle() o costoRefrigerio(), que permiten que cada subclase defina su propia implementación, así como su uso a nivel general mediante referencias del tipo Vuelo.

Sobreescritura, Sobrecarga, e Interfaces

Sobreescritura: Los métodos “getDetalle” y “costoRefrigerio” se sobrescriben en las subclases de Vuelo para adaptar la lógica a su comportamiento según tipo de vuelo.

Sobrecarga: El método “cancelarPasaje” de la interfaz IAerolinea presenta sobrecarga, tiene dos versiones de implementación: una que recibe un DNI y un código de vuelo, y otra que hace uso de un DNI y el código de pasaje.

Interfaces: La clase Aerolinea implementa la interfaz IAerolinea, lo que permite que todos los métodos que se definan en ella se implementen al menos de forma uniforme entre todas las aerolíneas y en pro de la abstracción.

Clases y Métodos Abstractos

La clase Vuelo es una clase abstracta, esta define atributos y métodos comunes, aunque no puede ser instancia en forma directa. Incluye métodos abstractos, como getDetalle(), que obligan a las subclases a implementar su respectiva lógica.