RESERVAS DE VUELOS

Se va a gestionar la reserva de billetes de avión para una aerolínea. No contemplaremos overbooking, es decir, se reservan como mucho tantos billetes como asientos tiene el avión.

(0.25 puntos) Enums

Enum TipoTarifa {OPTIMA, CONFORT, FLEXIBLE}
Enum TipoAsiento {TURISTA, BUSINESS}

(0.5 puntos) Clase Persona (abstracta)

Clase básica con información: Long id, nombre, apellidos, dniPasaporte, email, dirección, ciudad, país, teléfono.
Constructor vacío, constructor con todos los parámetros, getters, setters y toString. Equals por dniPasaporte.

(0.5 puntos) Clase Pasajero hereda de Persona

Añade atributos: String preferenciasComida, String restriccionesMedicas, Asiento asiento.
Constructor con todos los parámetros, Constructor copia, getters, setters y toString.

(0.75 puntos) Clase Asiento (abstracta)

Añade atributos: Long id, Pasajero pasajero, Double precioBase, Integer fila, String letra, TipoAsiento tipo
Constructor (sin pasajero, ni tipo de asiento), getters, setters, toString (ojo a Pasajero – Asiento), equals por id.
Métodos extra:

- **getCodigo**(): devuelve un String con la fila y la letra del asiento.
- **calcularPrecio**(): devuelve el precio del asiento. Método abstracto.

(0.5 puntos) Clase AsientoTurista
Añade atributos: boolean ventana
Constructor (el tipo lo pone a Turista), getters, setters, toString (codigo, precio, tipo, tiene ventana, dni Pasajero)
Métodos extra:

- **calcularPrecio**(): devuelve el precio del asiento. Suma 10€ si es un asiento de ventana.

(0.5 puntos) Clase AsientoBusiness
Añade atributos: boolean comida
Constructor (el tipo lo pone a Business), getters, setters, toString (codigo, precio, tipo, tiene comida, dni Pasajero)
Métodos extra:

- **calcularPrecio**(): devuelve el precio del asiento, suma 25% al precioBase. Si incluye comida suma 30€.

(2 puntos) Clase Vuelo

Añade atributos: String codigo, String origen, String destino, LocalDate fecha, LocalTime hora, Double precioBase, Integer asientosDisponibles, ArrayList<Asiento> asientos.
La propiedad *asientosDisponibles* es el número de asientos del avión.
Constructor, getters, setters, toString, equals por codigoVuelo.
En el constructor deberás crear todos esos asientos y meterlos en la colección asientos, que haya un 30% Business y un 70% Turista. A cada asiento se le pone de precioBase, el precioBase del vuelo.

Métodos extra:

- **int verificarDisponibilidad**(TipoAsiento tipoAsiento): devuelve el número de asientos disponibles del tipo indicado (Business o Turiesta) que no tienen Pasajero asignado. No hay que usar *instanceof*, hay un método getTipo().
- **Asiento buscarAsientoDisponible**(TipoAsiento tipoAsiento): devuelve el primer asiento libre en el vuelo del tipo indicado. No hay que usar *instanceof*, hay un método getTipo().
- **boolean ocuparAsiento**(Pasajero, Asiento): si ese asiento está disponible, se pone que el Pasajero tiene ese Asiento, y que el Asiento es ocupado por el Pasajero.

- **liberarAsiento**(Asiento): se quita el Pasajero del Asiento, y del Pasajero su Asiento queda null (si el asiento tenía pasajero previamente).
- **diasFaltanVuelo**(): devuelve los días que faltan para el vuelo.
- **ArrayList<Pasajero> getPasajeros():** devuelve la lista de pasajeros del vuelo.

(2 puntos) Clase ReservaVuelo
Añade atributos: Long id, Vuelo vuelo, ArrayList<Pasajero> pasajeros, TipoTarifa tipoT, TipoAsiento tipoA
Constructor, getters, setters, toString (mostrará el id, el código del vuelo, también los días que faltan para el Vuelo, no imprime pasajeros ni asientos, llama a imprimirBilletes).
El id no debe aparecer en el constructor, debe hacerse con un campo estático, cada reserva incrementa el valor.

- addPasajero, removePasajero
- **ArrayList<Asiento> getAsientos**(): devuelve los asientos asignados a pasajeros
- **reservaAsiento**(Pasajero pasajero): Primero verifica que haya disponibilidad de Asientos del tipo indicado, si hay disponibilidad, buscaremos un asiento libre del tipo y ***ocupará*** el asiento en el vuelo, añadirá el pasajero a la lista de pasajeros de la reserva. **IMPORTANTE**: aquí debemos **clonar** el objeto Pasajero llamando a su ***constructor copia***, antes de ocupar el asiento, y a ocupar() se le pasa el pasajero clonado y se añade también a la lista de pasajeros el pasajero clonado. Es decir, cada reserva lleva un objeto pasajero diferente, si fuera el mismo, el asiento del pasajero se modificaría cuando se haga otra reserva.
- **calcularPrecioTotal**(): devuelve la suma del precio de los asientos de la reserva. Según el tipo de tarifa, sumará un valor al precio de cada asiento individual: si es Optima suma un 10%, Confort suma un 15% y Flexible suma un 30%
- **imprimirBilletes**(): muestra la información de cada pasajero, junto su asiento y el precio total de la reserva.

(2 puntos) Clase Atrapame
Añade atributos: ArrayList<Vuelo> vuelos, ArrayList<ReservaVuelo> reservas.
Constructor, getVuelos, getReservas
Métodos:

- **addVuelo**(Vuelo vuelo): añade un vuelo a la colección.
- Boolean **crearReserva**(Vuelo, ArrayList<Pasajero>, TipoTarifa, TipoAsiento): comprueba que haya tantos asientos disponibles del tipo de asiento como números de pasajeros hay en la reserva. Si no hay devuelve false. Si sí hay, crear la reserva y **añade** a la reserva cada pasajero. Añade la reserva a la lista de reservas.
- **cancelarReserva**(Long id): antes de eliminar la reserva de la colección, debe entrar a su vuelo y **liberar** los asientos que tenían asignados en el vuelo.
- **ArrayList<Vuelo> buscarVuelos(String destino)**
- **ArrayList<Vuelo> buscarVuelos(LocalDate fecha)**
- **ArrayList<Vuelo> buscarVuelos(LocalDate fecha, String destino)**
- **ArrayList<Reserva> buscarReservas(String dni)**

(1 punto) Clase Principal
Crea un main en el que añadas 1000 vuelos (en un for).
Además, para uno de esos vuelos añade 5 reservas con diferentes pasajeros.
Imprime el vuelo.
Imprime las reservas.
Muestra las reservas de un dni.
Muestra todos los vuelos a un destino que elijas en una determinada fecha.

(0.5 puntos extra) Pregunta final
Cómo mejorarías saber los pasajeros de un vuelo. Explícalo en un fichero de texto junto al proyecto.
Qué pasa si en lugar de 1000 vuelos creas 100000 o un millón. Dime cuánta memoria te ocupa el programa cuando te crea tantos vuelos.

SE VALORARÁ LA LIMPIEZA DEL CÓDIGO, LA TABULACIÓN, LOS CONCEPTOS DE POO, LOS COMENTARIOS.

DEBES ENTREGAR EL CÓDIGO DEL PROYECTO JUNTO CON EL FICHERO JAR GENERADO Y EL JAVADOC. PUEDES USAR UN REPOSITORIO GITHUB PARA LA ENTREGA.