UTN - FR Mar del Plata - TSP Laboratorio III - Segundo parcial Comisión 6 - 2024	Nombre y apellido	Nota

Importante

- El nombre del proyecto debe ser el nombre y apellido del alumno.
- Subir al campus el proyecto comprimido.

Sistema de reserva de vuelos

Completar la implementación de un sistema de reservas de vuelos para que funcione correctamente. El código proporcionado compila, pero no realiza ninguna funcionalidad completa. Se han dejado 15 comentarios TODO con indicaciones específicas en el código para guiar la implementación. Debes completar cada uno de estos TODO para cumplir con los requerimientos del sistema. A continuación se detallan los requerimientos y las instrucciones para cada TODO.

Requerimientos específicos

Interfaces

TODO 1: (5 puntos)

Implementar la interfaz Buscable<C> que permita buscar elementos que cumplan con un criterio de búsqueda.
La interfaz debe tener el método matches que recibe un criterio de búsqueda y devuelve true si el elemento cumple con el criterio.

TODO 2: (5 puntos)

Implementar la interfaz Filtrable<F> que permita filtrar elementos que cumplan con un criterio de filtrado.
La interfaz debe tener el método filter que recibe un criterio de filtrado y devuelve true si el elemento cumple con el criterio.

Clases

TODO 3: (10 puntos)

Implementar la clase Pasajero que cumpla con las interfaces Buscable<String>, Comparable<Pasajero> y Filtrable<String>. La clase debe tener los siguientes atributos:

- String pasaporte: representa el número de pasaporte del pasajero (por ejemplo: AAA123456).
- String nombre: representa el nombre completo del pasajero.

Los pasaportes y nombres deben ser almacenados en mayúsculas. Dos pasajeros son iguales si tienen el mismo pasaporte. Los pasajeros se pueden buscar por pasaporte y filtrar por nombre. Se pueden comparar por nombre.

TODO 4: (10 puntos)

Implementar la clase Vuelo que cumpla con las interfaces Buscable<String>, Comparable<Vuelo> y Filtrable<Destino>. La clase debe tener los siguientes atributos:

- String codigoVuelo: representa el código del vuelo.
- Destino origen: representa el origen del vuelo.
- Destino destino: representa el destino del vuelo.
- String horarioSalida: representa el horario de salida del vuelo.
- String horarioLlegada: representa el horario de llegada del vuelo.
- EstadoVuelo estado: representa el estado del vuelo.

La clase debe tener los siguientes métodos:

- void retrasado(String nuevoHorario): cambia el estado del vuelo a retrasado y modifica el horario de salida.
- void cancelado(): cambia el estado del vuelo a cancelado.
- void despegado(): cambia el estado del vuelo a despegado.

El vuelo se crea por defecto en estado programado. Los vuelos se pueden buscar por código de vuelo y filtrar por destino. Se pueden comparar por destino. El formato del horario de salida y de llegada debe ser dd/MM/yyyy HH:mm.

TODO 5: (10 puntos)

Implementar la clase Reserva que cumpla con las interfaces Buscable<String>, Comparable<Reserva> y Filtrable<Pasajero>. La clase debe tener los siguientes atributos:

- String idReserva: representa el identificador de la reserva.
- Pasajero pasajero: representa el pasajero que realiza la reserva.
- Vuelo vuelo: representa el vuelo que se reserva.
- ClaseAsiento claseAsiento: representa la clase de asiento que se reserva.
- EstadoReserva estado: representa el estado de la reserva.

La clase debe tener los siguientes métodos:

- void cambiarAsiento(ClaseAsiento nuevaClase): permite cambiar la clase de asiento de la reserva.
- void cancelar(): permite cambiar el estado de la reserva a cancelada.
- void confirmar(): permite cambiar el estado de la reserva a confirmada.

La reserva se crea por defecto en estado pendiente. El formato del ID de reserva debe ser R{contador}{vuelo.getHorarioSalida().getNano()}, donde contador es un número que se incrementa en 1 por cada reserva, comenzando en 0. Las reservas se pueden buscar por ID de reserva y filtrar por pasajero. Se pueden comparar por ID de reserva.

TODO 6: (15 puntos)

Implementar la clase Gestor<T> que permita agregar, buscar, filtrar y listar elementos de tipo T. T debe implementar las interfaces Buscable, Filtrable y Comparable.
El gestor debe almacenar los elementos en un TreeSet.
Utiliza el método matches de la interfaz Buscable y el método filter de la interfaz Filtrable. Para limitar el tipo T a elementos que implementen las interfaces Buscable, Filtrable y Comparable, se puede utilizar la cláusula & en la definición de la clase:

```
public class Gestor<T extends Interface1 & Interface2 & Interface3> { }
```

TODO 7: (15 puntos)

Implementar la clase Sistema.
La clase Sistema debe contener los métodos necesarios para agregar, buscar y filtrar vuelos, pasajeros y reservas.
Debe contener métodos para crear, confirmar, cancelar y cambiar la clase de asiento de una reserva.
Los elementos deben ser almacenados en instancias de la clase Gestor. El método que busca un pasajero debe lanzar una excepción si el pasaporte no existe. La excepción debe ser de tipo PasajeroNoEncontradoException e informar el pasaporte que no se encontró.

TODO 8: (2 puntos)

Generar datos de prueba.
Agregar 10 vuelos con distintos destinos y fechas de salida. Para generar fechas en el futuro, se pueden utilizar los métodos plusDays, plusWeeks y plusMonths de la clase LocalDateTime.
Agregar 2 pasajeros con distintos nombres y pasaportes.
Agregar una reserva para cada pasajero en un vuelo distinto.

Funcionalidad

TODO 9: (3 puntos)

Obtener la lista de vuelos filtrados por destino y mostrarlos en la lista de vuelos.
Llamar al método de la clase Sistema que filtra los vuelos por destino.
Llamar al método mostrarVuelos para mostrar los vuelos en la lista.

TODO 10: (3 puntos)

Completar el método init en DetalleVuelo.
El método debe completar los campos del formulario con los datos del vuelo recibido como parámetro. Campos a completar: numeroVuelo, comboOrigen, comboDestino, horarioSalida, horarioLlegada, comboEstadoVuelo.

TODO 11: (3 puntos)

Completar el método init en DialogReservaVuelo.
El método debe completar los campos del formulario con los datos del vuelo recibido como parámetro. Campos a completar: numeroVuelo, comboOrigen, comboDestino, horarioSalida, horarioLlegada.

TODO 12: (10 puntos)

Completar la acción del botón Buscar.
El método debe buscar un pasajero en el sistema a partir del pasaporte ingresado.
Si el pasajero no existe, debe mostrar un mensaje de error.
Si el pasajero existe, debe completar los campos pasaporte y nombreCompleto con los datos del pasajero.

TODO 13: (3 puntos)

Completar la acción del botón Reservar.
El método debe crear una reserva en el sistema a partir de los datos ingresados en el formulario.

Si la reserva se crea correctamente, debe mostrar un mensaje con el código de la reserva y dar la opción de copiarlo al portapapeles.

TODO 14: (3 puntos)

Obtener la reserva por código y mostrarla en el panel de detalle de reserva.
Llamar al método de la clase Sistema que busca la reserva por código.
Llamar al método mostrarReserva para mostrar la reserva en el panel.

TODO 15: (3 puntos)

Mostrar la reserva en el panel de detalle de reserva.
Mostrar el código de la reserva, origen, destino, horario de salida, horario de llegada, pasajero y estado de la reserva.

Evaluación

Número	Descripción del TODO	Puntos	Puntos obtenidos
TODO 1	Implementar la interfaz Buscable<C>	5	
TODO 2	Implementar la interfaz Filtrable<F>	5	
TODO 3	Implementar la clase Pasajero que cumpla con Buscable<String>, Comparable<Pasajero> y Filtrable<String>	10	
TODO 4	Implementar la clase Vuelo que cumpla con Buscable<String>, Comparable<Vuelo> y Filtrable<Destino>	10	
TODO 5	Implementar la clase Reserva que cumpla con Buscable<String>, Comparable<Reserva> y Filtrable<Pasajero>	10	
TODO 6	Implementar la clase Gestor<T>	15	
TODO 7	Implementar la clase Sistema	15	
TODO 8	Generar datos de prueba	2	
TODO 9	Obtener la lista de vuelos filtrados por destino y mostrarlos en la lista de vuelos	3	
TODO 10	Completar el método init en DetalleVuelo	3	
TODO 11	Completar el método init en DialogReservaVuelo	3	
TODO 12	Completar la acción del botón Buscar	10	
TODO 13	Completar la acción del botón Reservar	3	
TODO 14	Obtener la reserva por código y mostrarla en el panel de detalle de reserva	3	
TODO 15	Mostrar la reserva en el panel de detalle de reserva	3	
		Total	