EVALUACION	Obligatorio	GRUPO	TODOS	FECHA	Marzo 2021
MATERIA	**Algoritmos y Estructuras de Datos 1**				
CARRERA	Analista Programador / Analista en Tecnologías de la información				
CONDICIONES	**- Puntos:** Máximo: 50 Mínimo: 0 **- Fecha máxima de entrega**: **IMPORTANTE** - Los grupos deben estar conformados por hasta 3 personas. - Deben Inscribirse (sacar la “boleta de entrega”). - La entrega será únicamente por gestión.				

Obligatorio: Sistema de Reserva de vuelos

Contenido

1. Introduction

Se desea implementar una plataforma para la búsqueda y reserva de vuelos.

Los vuelos serán registrados en el sistema con un número (que podrá ser único dentro de una aerolínea.

Cada vuelo tiene asociado una ciudad origen, una ciudad destino, una cantidad de asientos (capacidad), la categoría (cantidad de estrellas) y una lista de servicios provistos.

El sistema deberá proveer funcionalidades para que los clientes puedan gestionar sus reservas para los vuelos, realizar búsquedas por ciudad origen, ciudad destino, por ranking o consultar los servicios de estos, entre otras.

Se proveen los siguientes tipos de datos que deberán ser respetados:

Sistema	public class Sistema{ /*Aquí introduzca la información que estime conveniente*/ }

Retorno	public class Retorno{
	enum TipoRet{OK,ERROR, NO_IMPLEMENTADA};
	/*Aquí introduzca la información que estime conveniente*/
	boolean valorBooleano
	int valorEntero;
	String valorString;
	Resultado resultado;
	}

Pueden definirse tipos de datos (clases) auxiliares.

2. Functionalities

2.1. Crear Sistema de Reservas

Firma: Retorno crearSistemaReservas();

Descripción: Crea la estructura necesaria para representar el sistema de reservas.

Retornos posibles
OK – Siempre retorna ok
ERROR – no existen errores posibles
NO-IMPLEMENTADA – Cuando aún no se implementó Es el tipo de retorno por defecto

2.2. Destruir Sistema de Mensajes

Firma: Retorno destruirSistemaReservas(); Descripción: Destruye el sistema de reservas y todos sus elementos, liberando la memoria utilizada.

Retornos posibles
OK – Siempre retorna ok
ERROR – no existen errores posibles
NO-IMPLEMENTADA – • Cuando aún no se implementó Es el tipo de retorno por defecto

2.3. Registrar Ciudad

Firma: Rectorno registrarCiudad(String Ciudad);
Descripción: Registra la ciudad “Ciudad”, del país “País”
Nota: el sistema en un inicio no contempla ciudades de igual nombre en países distintos.

Retornos posibles
OK – • En caso de agregar la ciudad con éxito
ERROR – • 1. En caso de que la ciudad ya exista
NO-IMPLEMENTADA – • Cuando aún no se implementó Es el tipo de retorno por defecto

2.4. Eliminar Ciudad

Firma: Rectorno eliminarCiudad(String Ciudad);
Descripción: Elimina la ciudad “Ciudad”, del país “País”
Nota: Elimina la ciudad siempre y cuando no tenga vuelos asociados, en cuyo caso debe emitir un mensaje.

Retornos posibles
OK – • En caso de eliminar la ciudad con éxito
ERROR – • 1. En caso de que la ciudad no exista
NO-IMPLEMENTADA – • Cuando aún no se implementó Es el tipo de retorno por defecto

2.5. Registrar vuelo.

Firma: Rectorno registrarVuelo(int numero, String aerolinea, String ciudadOrigen, String ciudadDestino, int estrellas, int capacidad, Calendar fechaHoraSalida, int duracion);

Descripción: Registra el vuelo número "numero", de la aerolínea "aerolinea" que sale de la ciudad "ciudadOrigen", con destino a la ciudad "ciudadDestino", cantidad de estrellas "estrellas", con una capacidad de pasajeros "capacidad", fecha de salida "fechaHoraSalida" y una duración "duración".

Nota: Al inicio los vuelos registrados no contarán con ningún servicio.

Retornos posibles
OK – • En caso de que el vuelo haya sido ingresado correctamente en el sistema
ERROR – • 1. En caso de que "estrellas" sea menor a 1 o mayor a 5. • 2. En caso de que "capacidad" o "duración" sean menores a 0. • 3. En caso de que ya exista un vuelo con número "numero" de la aerolínea "aerolínea". • 4. En caso de que ya haya un vuelo entre el origen y el destino • 5. En caso de que no exista ciudad origen o ciudad destino en la lista de ciudades.
NO-IMPLEMENTADA – • Cuando aún no se implementó Es el tipo de retorno por defecto

2.6. Ingresar Servicio.

Firma: Retorno ingresarServicio(String aerolinea, int numero, String servicio);
Descripción: Ingresa el servicio "servicio" al número de vuelo "numero" de la aerolínea "aerolinea".
Nota 1: No se hará ningún control de unicidad sobre los servicios.
Nota 2: El ingreso de los servicios deberá realizarse en el menor tiempo posible.

Retornos posibles
OK – • En caso de que el servicio "servicio" haya sido ingresado correctamente en el número de vuelo "numero" de la aerolínea "aerolinea". en el sistema
ERROR – • 1. En caso de que no exista un número de vuelo "numero" registrado dentro de la aerolínea "aerolinea".
NO-IMPLEMENTADA – • Cuando aún no se implementó Es el tipo de retorno por defecto

2.7. Borrar Servicio.

Firma: TipoRet borrarServicio(String aerolinea, int numero, String servicio);
Descripción: Borra la primera ocurrencia del servicio "servicio" del número de vuelo "numero" de la aerolínea "aerolinea".

Retornos posibles
OK – • En caso de que el servicio "servicio" haya sido borrado correctamente del número de vuelo "numero" de la aerolínea "aerolinea"
ERROR – • 1. En caso de que no exista un número de vuelo "numero" registrado dentro de la aerolínea "aerolinea". • 2. En caso de que no exista el servicio "servicio" registrado dentro de la aerolínea "aerolinea".
NO-IMPLEMENTADA – • Cuando aún no se implementó Es el tipo de retorno por defecto

2.8. Ingresar Comentario.

Firma:
Retorno ingresarComentario(String aerolinea, int numero, String comentario, int ranking);
Descripción: Ingresa el comentario "comentario" al número de vuelo "numero" de la aerolínea "aerolinea". Cada comentario llevará una calificación para la aerolínea. El ranking general de la aerolínea quedará definido por el promedio de todas sus calificaciones.
Nota: El ranking será representado con un número entre 0 y 5 inclusive.

Retornos posibles	
OK –	• En caso que el comentario "comentario" haya sido ingresado correctamente al número de vuelo "numero" de la aerolínea "aerolinea
ERROR –	• 1. En caso de que el ranking sea menor a 0 o mayor a 5. • 2. En caso de que no exista el número de vuelo "numero" de la aerolínea "aerolinea".
NO-IMPLEMENTADA –	• Cuando aún no se implementó Es el tipo de retorno por defecto

2.9. Realizar Reserva.

Firma: Retorno realizarReserva(int cliente, int numero, String aerolinea);

Descripción: Realiza la reserva de un pasaje para el vuelo número "numero" de la aerolínea "aerolinea". En caso de que no haya cupos disponibles, el cliente "cliente" quedará en lista de espera. El cliente podrá realizar más de una reserva.

Retornos posibles	
OK –	• En caso de que la reserva haya sido efectuada correctamente en el número de vuelo "numero" de la aerolínea "aerolinea". • En caso de que la reserva haya quedado en lista de espera
ERROR –	• 1. En caso de que no exista un numero de vuelo "numero" registrado en la aerolínea "aerolinea".
NO-IMPLEMENTADA –	• Cuando aún no se implementó Es el tipo de retorno por defecto

2.10. Cancelar Reserva

Firma: Retorno cancelarReserva(int cliente, int numero, String aerolinea);

Descripción: Cancela la reserva de un cliente "cliente" para el vuelo número "numero" de la aerolínea "aerolinea". En caso de que la cancelación se lleve a cabo correctamente y haya clientes en lista de espera, el cupo liberado será ocupado por el primer cliente de la lista. La búsqueda de las reservas se efectuará primero dentro de las reservas de vuelos y luego en la lista de espera. En caso de que el cliente "cliente" tenga más de una reserva para el numero de vuelo "numero" de la aerolínea "aerolinea" se cancelará solo la primera reserva encontrada en el orden establecido.

Retornos posibles	
OK –	• En caso de que la cancelación del número de vuelo "numero" de la aerolínea "aerolinea" se lleve a cabo correctamente para el cliente "cliente".

ERROR –	• 1. En caso de que no exista un numero de vuelo “numero” registrado en la aerolínea “aerolinea”. • 2. En caso de que el cliente “cliente” no tenga ninguna reserva hecha en el número de vuelo “numero” registrado en la aerolínea “aerolinea”.
NO-IMPLEMENTADA –	• Cuando aún no se implementó Es el tipo de retorno por defecto

2.11. Listado de Servicios.

Firma: Retorno listarServicios(int numero, String aerolinea);

Descripción: Lista todos los servicios prestados por la aerolínea “aerolinea” en el número de vuelo “numero” con el siguiente formato:

Servicios de la Aerolínea < Aerolínea > para el vuelo número <numero >

1 - <Servicio 1>

- - - - - - -

N - <Servicio N>

En caso de que no haya servicios registrados en la aerolínea “aerolinea”, el sistema deberá imprimir: No existen servicios registrados en la Aerolínea vuelo número <numero >

Retornos posibles	
OK –	• En caso que se imprima el listado correctamente.
ERROR –	• 1. En caso de que no exista un numero de vuelo “numero” registrado en la aerolínea “aerolinea”.
NO-IMPLEMENTADA –	• Cuando aún no se implementó Es el tipo de retorno por defecto

2.12. Listado de vuelos por aerolínea .

Firma: Retorno listarVuelosAerolinea(String aerolinea);
Descripción: Lista todos los vuelos registrados en la aerolínea “aerolinea” ordenados por Ranking con el siguiente formato.
Vuelos de la aerolínea <Aerolinea>
<Ciudad Origen1>
<Numero> <Ciudad destino><Estrellas> <Ranking>
…
<Numero> <Ciudad destino><Estrellas> <Ranking>
En caso de que no haya ningún número de vuelo “numero” registrado en la aerolínea “aerolinea” el sistema deberá imprimir:
No existen vuelos registrados.
Nota: La impresión de este listado (por ser muy utilizado) deberá realizarse en el menor tiempo posible.

Retornos posibles	
OK –	• Siempre retorna OK.
ERROR –	• 1. No hay errores posibles.

NO-IMPLEMENTADA – • Cuando aún no se implementó Es el tipo de retorno por defecto

2.13. Listado de Aerolíneas por Ranking

Firma: TipoRet listarAerolineasRanking();
Descripción: Lista todos las Aerolíneas registradas en el sistema ordenadas por ranking descendente.
El formato de impresión deberá ser el siguiente:
Aerolínea ordenados por ranking
<Aerolinea> - <Ranking>
…
<Aerolinea> - <Ranking>
En caso de que no exista ningúna aerolínea registrada en el sistema, se deberá imprimir: No hay registros de aerolíneas en el sistema.

Retornos posibles
OK – • Siempre retorna OK.
ERROR – • 1. No hay errores posibles.
NO-IMPLEMENTADA – • Cuando aún no se implementó Es el tipo de retorno por defecto

2.14. Listado de Comentarios

Firma: TipoRet listarComentarios(int numero, String aerolinea);
Descripción: Lista todos comentarios ingresados para el numero de vuelo “numero” de la aerolínea “aerolínea”. Los comentarios más recientes deberán aparecer primeros en el listado.
El formato deberá ser el siguiente:
N - <Comentario N> - <RankingN>
…
1 - <Comentario 1> - <Ranking1>
En caso de que no existan comentarios para el numero de vuelo “numero” de la aerolínea “aerolinea” el sistema deberá imprimir:
No se han agregado comentarios al número <Numero> de la aerolínea <Aerolinea>

Retornos posibles
OK – • En caso de que el listado se haya imprimido correctamente.
ERROR – • 1. En caso de que no exista el número de vuelo “numero” registrado en la aerolínea “aerolinea”..
NO-IMPLEMENTADA – • Cuando aún no se implementó Es el tipo de retorno por defecto

2.15. Lista de Espera

Firma: Retorno listarEspera(int numero, String aerolinea);
Descripción: Lista los clientes en lista de espera para el numero de vuelo “numero” dentro de la aerolínea “aerolinea”. Los clientes deberán imprimirse en el orden que serán considerados para tomar un posible asiento libre. Se considerará primero al cliente que esté hace más tiempo en la lista de espera.
El formato del listado deberá ser el siguiente:
Lista de espera para el número de vuelo <Numero> de la aerolínea <Aerolinea>

1 - <CedulaCliente1>
…
N - <CedulaClienteN>
En caso de que no existan clientes en la lista de espera se deberá imprimir en pantalla: No existen reservas pendientes para el numero de vuelo <Numero> de la aerolínea <Aerolinea>

Retornos posibles	
OK –	• En caso de que el listado se haya imprimido correctamente.
ERROR –	• 1. En caso de que no exista el número de vuelo “numero” registrado en la aerolínea “aerolinea”..
NO-IMPLEMENTADA –	• Cuando aún no se implementó Es el tipo de retorno por defecto

2.16. Cuadro de Duraciones de los vuelos entre ciudades.

Firma: Retorno mostrarDuraciones();
Descripción: Muestra por pantalla una tabla conteniendo las duraciones de todos los viajes entre las ciudades registradas en el sistema. Solamente se mostrarán las duraciones entre ciudades vecinas (un vuelo directo las une), o a las que puedo llegar pasando por una sola ciudad (dos vuelos).

Retornos posibles	
OK –	• En caso de que el listado se haya imprimido correctamente.
ERROR –	• No hay errores posibles
NO-IMPLEMENTADA –	• Cuando aún no se implementó Es el tipo de retorno por defecto

El siguiente es un ejemplo de impresión.
El formato de impresión de cada celda podrá ser:

- <Minutos> (en caso de vuelo directo)
- <Minutos> / <Ciudad por la que debo pasar para llegar> (en caso de vuelo con una escala)

Por ejemplo, para ir de Montevideo a San Pablo, puedo hacerlo directamente por un único vuelo de xxxx minutos.

Otra opción es un vuelo con una escala. Como no hay vuelo directo entre Montevideo y Nueva York, se puede ir de Montevideo a Rio con una duración de xx minutos y de Rio a Nueva York con una duración de yy minutos, por lo que lo que sumando el largo de ambos tramos serían xx+yy minutos.

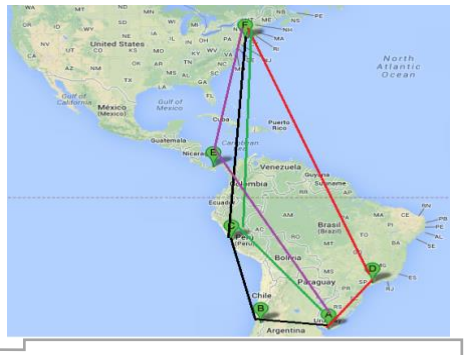
Nota: en este ejemplo, se asumirá que los vuelos entre las ciudades están registrados bidireccionalmente. Recordar que, al registrar un vuelo, sólo se agrega de origen a destino, y no en ambas direcciones.

Origen/destino	Montevideo	BsAires	San Pablo	Rio	Santiago	Nueva York	Miami	Londres
Montevideo			xxxx	xx		Xx + yy / Rio		
BsAires								
San Pablo	xxxx							
Rio	xx					yy		
Santiago								
Nueva York	Xx + yy / Rio			yy				
Miami								
Londres								

2.17. Cargar mapa con duraciones de los vuelos para calcular el mejor camino.

Dada la siguiente matriz en la que se almacena las distancias que existen entre las ciudades que se muestran en el ejemplo.

	A Montevideo	B Santiago	C Lima	D San Pablo	E Panamá	F New York
A	0	10	25	15	30	0
B	10	0	20	0	0	0
C	25	20	0	0	0	40
D	15	0	0	0	0	45
E	30	0	0	0	0	25
F	0	0	40	45	25	0



Montevideo, Lima, New York

Montevideo, Santiago, New York

Montevideo, San Pablo, New York

Montevideo, panamá, New York

Se desea implementar los siguientes métodos:

2.18. Cargar Matriz de Distancias

Firma: `Retorno CargarDistancias(int[][] Mapa, String ciudadOrigen, String Ciudad destino, int duracion);`

Nota: La matriz debe ser creada en función de la cantidad de ciudades definidas en el sistema. Debemos definir en la creación del sistema un parámetro cantidad de ciudades para luego validar y crear la matriz en función de esta cantidad.

2.19. Camino más corto

Firma: `Retorno BuscarCamino ( int [][] Mapa, string origen, string destino)`

Retorna una lista 

Se debe retornar el camino más corto para llegar de un origen a un destino restringiendo la búsqueda a caminos que solo tengan una ciudad intermedia.

Nota: Las casillas que tienen valor distinto de 0 son ciudades que tienen conexión y las casillas que tienen valor 0 no tienen conexión.

Defina las estructuras que considere necesarias para resolver el problema.

3. Documentación

Se pide que la documentación incluya:

1. Las pre y post condiciones de los métodos solicitados.
2. Diagrama de la estructura de datos que se implementó para representar el sistema de reservas junto con una breve explicación indicando por qué eligió dichas estructuras.
3. El conjunto de pruebas diseñadas y ejecutadas y el resultado obtenido de esta ejecución con un screenshot de la pantalla mostrando el resultado de cada uno de los tests.
4. Los nombres de cada @Test (caso de prueba) debe ser descriptivos, dejando claro que es lo que se pretende testear.
5. No se aceptará Tests con nombres Test1, Test2, etc.

Información importante

 Puntaje mínimo/máximo: 0/50
 Los obligatorios se realizan por equipos de hasta 3 **estudiantes.**
 Plazo máximo de segunda entrega (con boleta): **----------.**
 Se deberán respetar los formatos de impresión dados para las operaciones que imprimen en consola.
 El resto de las operaciones no deben imprimir nada en consola.
 El sistema no debe requerir ningún tipo de interacción con el usuario por consola.
 Es obligación del estudiante mantenerse al tanto de las aclaraciones que se realicen en clase o a través
del foro de aulas.
 Se la selección adecuada de las estructuras para modelar el problema y la eficiencia en
cada una de las operaciones es muy importante.
 Deberá aplicar la metodología vista en el curso.
 Para la presentación de la documentación se publicará en aulas.ort.edu.uy un template. (El uso de este template es obligatorio).
 El proyecto será implementado en lenguaje JAVA sobre una interfaz que debe respetar las firmas que se describen en el documento. (El uso de esta interfaz es obligatorio y no puede ser modificada).