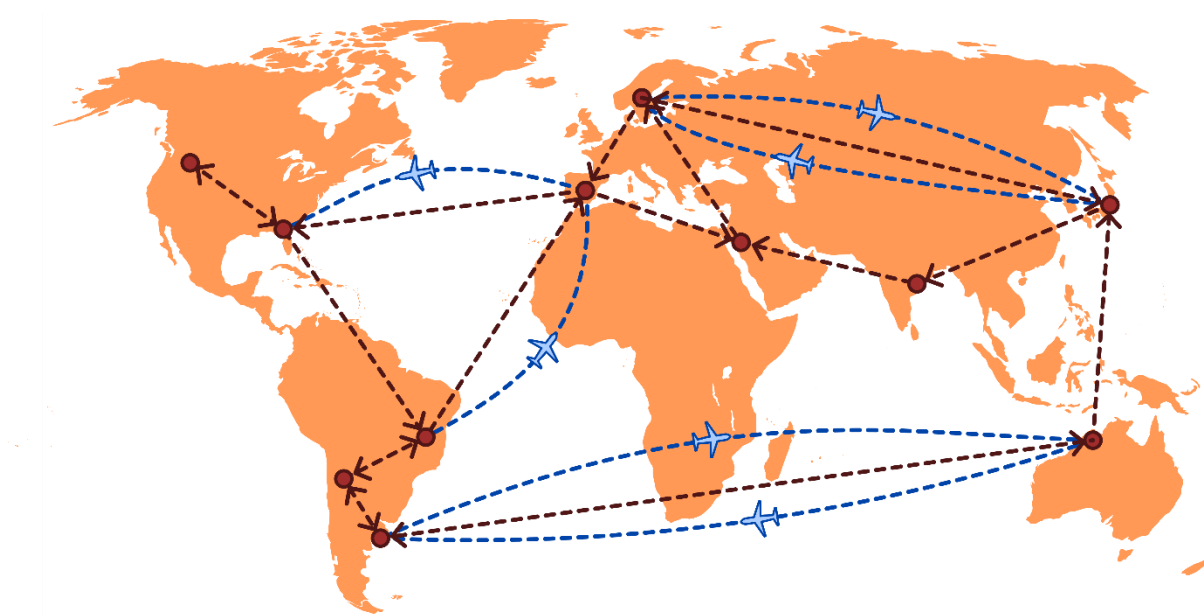


EVALUACION	Obligatorio	GRUPO	Todos	FECHA	Marzo 2024 – V2
MATERIA	Algoritmos y Estructuras de Datos 2				
CARRERA	Analista Programador – Analista en TI				
CONDICIONES	<ul style="list-style-type: none"> <li>• Puntos: Máximo: 35    Mínimo: 1</li> <li>• Fecha máxima de entrega: <b>30/05/2024</b></li> <li>• LA ENTREGA SE REALIZA EN FORMA ONLINE EN ARCHIVO NO MAYOR A 40MB EN FORMATO ZIP, RAR.</li> <li>• <b>IMPORTANTE:</b> <ul style="list-style-type: none"> <li>○ Inscribirse</li> <li>○ Formar grupos de hasta <b>2 personas del mismo dictado (grupo)</b>.</li> <li>○ Subir el trabajo a Gestión antes de la hora indicada, ver hoja al final del documento: “RECORDATORIO”</li> </ul> </li> </ul>				

## Introducción

Se desea implementar un programa para ingresar y consultar información sobre los pasajeros, aeropuertos, aerolíneas, conexiones y vuelos con las cotas de tiempo requeridas para cada operación.



Todas las operaciones deberán devolver una instancia de la clase Retorno. Dicha clase contiene:

- Un **resultado**, que especifica si la operación se pudo realizar correctamente (OK), o si ocurrió algún error (según el número de error).
- Un **valorEntero**, para las operaciones que retornen un número entero.
- Un **valorString**, para las operaciones que retornen un String, o un valor más complejo, por ejemplo, un listado, el cual será formateado según lo indicado en los ejemplos.

```
public class Retorno {  
    public enum Resultado {  
        OK,  
        ERROR_1, ERROR_2, ERROR_3, ERROR_4, ERROR_5, ERROR_6, ERROR_7,  
        NO_IMPLEMENTADA  
    }  
  
    private Resultado resultado;  
    private final int valorEntero;  
    private String valorString;  
  
    . . .  
}
```

Se provee: una interfaz llamada **Sistema**, la cual no podrá ser modificada en ningún sentido, y una clase **ImplementacionSistema** que la implementa, donde su equipo deberá completar la implementación de las operaciones solicitadas.

Consideraciones:

- La clase sistema NO PODRÁ SER UN SINGLETON. Debe ser una clase **instanciable**.
- Pueden definirse tipos de datos (clases) auxiliares.
- Se provee un proyecto base con la estructura de las clases.

## Funcionalidades

### 01. Inicializar Sistema

Retorno **inicializarSistema**(int maxAeropuertos, int maxAerolineas);

Descripción: Inicializa las estructuras necesarias para representar el sistema especificado, capaz de registrar como máximo la cantidad *maxAeropuertos* de aeropuertos diferentes en el sistema como así también la cantidad máxima de aerolíneas habilitadas para brindar servicio.

Restricción de eficiencia: no tiene.

Retornos posibles	
OK	Si el sistema fue inicializado exitosamente.
ERROR	<ol style="list-style-type: none"><li>1. Si <i>maxAeropuertos</i> es menor o igual a 5.</li><li>2. Si <i>maxAerolineas</i> es menor o igual a 3.</li></ol>
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

## 02. Registrar pasajero

Retorno **registrarPasajero**(String cedula, String nombre, String teléfono, Categoria categoria);

Descripción: Registra el pasajero con sus datos. La cédula es su identificador único.

Restricción de eficiencia: Esta operación deberá realizarse en orden  **$O(\log n)$**  promedio.

Retornos posibles	
OK	Si el pasajero fue registrado exitosamente.
ERROR	<ol style="list-style-type: none"> <li>1. Si alguno de los parámetros es vacío o <i>null</i>.</li> <li>2. Si la cédula no es una cédula con formato válido.</li> <li>3. Si ya existe un pasajero registrado con esa cédula.</li> </ol>
NO_IMPLEMENTADA	Cuando aún no se implementó.

Restricción: (Investigación) Se requiere el uso de expresiones regulares para lograr validar el formato de la cédula: N.NNN.NNN-N o NNN.NNN-N (No se debe realizar la validación del dígito verificador de la cédula). Además, se debe validar que el primer dígito sea positivo (1-9) por lo que las cedulas que comienzan en 0 son inválidas.

Las categorías posibles son:

- PLATINO
- FRECUENTE
- ESTANDAR

Ejemplos:

Resultado: OK

```
registrarPasajero("1.345.345-4", "Guillermo", "555-054888", Categoria.ESTANDAR);
```

Resultado esperado: ERROR\_2

```
registrarPasajero("1.3.45.345-4", Guillermo, "555-054888", Categoria.FRECUENTE);
```

```
registrarPasajero("3-45.345-4","Guillermo", "555-054888", Categoria.PLATINO);
```

```
registrarPasajero("0.232.232","Romina", "232-054888", Categoria.FRECUENTE);
```

### 03. Buscar Pasajero

Retorno **buscarPasajero**(String cedula);

Descripción: Retorna en *valorString* los datos del pasajero con el formato “cedula;nombre;telefono;categoria”. Además, en el campo *valorEntero* de la clase Retorno, deberá devolver la cantidad de elementos que recorrió durante la búsqueda en la estructura utilizada.

Restricción de eficiencia: Esta operación deberá realizarse en orden  **$O(\log n)$**  promedio.

Retornos posibles	
OK	Si el usuario se encontró. Retorna en <i>valorString</i> los datos del pasajero. Retorna en <i>valorEntero</i> la cantidad de elementos recorridos durante la búsqueda.
ERROR	<ol style="list-style-type: none"> <li>1. Si la cédula es vacía o null.</li> <li>2. Si la cédula no tiene formato válido.</li> <li>3. Si no existe un pasajero registrado con esa cédula.</li> </ol>
NO_IMPLEMENTADA	Cuando aún no se implementó.

Formato de retorno de *valorString*: cedula;nombre;teléfono;categoria.

Por ejemplo, *valorString* del retorno en una consulta válida:

1.345.345-4;Guillermo;555-054888;Estandar

## 04. Listar pasajeros por cédula ascendente

Retorno **listarPasajerosAscendente()**;

Descripción: Retorna en *valorString* los datos de todos los pasajeros registrados, ordenados por cédula en forma creciente (numéricamente).

Restricción de eficiencia: Esta operación deberá realizarse en orden **O(n)**.

Retornos posibles	
OK	Retornando el listado de pasajeros en <i>valorString</i> .
ERROR	No hay errores posibles.
NO_IMPLEMENTADA	Cuando aún no se implementó.

Formato de retorno del *valorString*:

*cédula1;nombre1;teléfono1;categoría1|cédula2;nombre2;telefono2;categoría2*

Por ejemplo:

800.232-2;Roberto;422-232;Estandar|1.345.345-4;Guillermo;555-054888;Estandar|4.985.345-4;Ana;555-044488;Frecuente

## 05. Listar pasajeros por categoría

Retorno **listarPasajerosPorCategoría**(Categoria unaCategoria);

Descripción: Retorna en valorString los datos de todos los pasajeros registrados con esa categoría. Ordenados numéricamente por cédula de forma creciente.

Restricción de eficiencia: Esta operación deberá realizarse en orden **O(k)**, siendo k la cantidad de pasajeros con dicha categoría.

Retornos posibles	
OK	Si se pudo listar los pasajeros que pertenecen a esa categoría correctamente.
ERROR	No hay errores posibles.
NO_IMPLEMENTADA	Cuando aún no se implementó.

Formato de retorno de *valorString*:

`cédula1;nombre1;teléfono1;categoría1|cédula2;nombre2;telefono2;categoría1`

Por ejemplo:

`1.985.345-4;Alberto;555-044488;Frecuente|3.345.345-4;Guillermo;555-054888;Frecuente`

## 06. Registrar aerolínea

Retorno **registrarAerolinea**(String codigo, String nombre);

Descripción: Registra la aerolínea en el sistema indicando el código (identificador único) y el nombre.

Restricción de eficiencia: Esta operación deberá realizarse en orden  **$O(\log n)$**  promedio.

Retornos posibles	
OK	Si la aerolínea fue registrada exitosamente.
ERROR	<ol style="list-style-type: none"><li>1. Si en el sistema ya hay registrados <i>maxAerolineas</i>.</li><li>2. Si código o nombre, son vacíos o null</li><li>3. Si ya existe una aerolínea con ese código.</li></ol>
NO_IMPLEMENTADA	Cuando aún no se implementó.



## 07. Listar aerolíneas por código descendente

Retorno **listarAerolineasDescendente()**;

Descripción: Retorna en `valorString` los datos de todas las aerolíneas registradas, ordenados por código en forma decreciente.

Restricción de eficiencia: Esta operación deberá realizarse en orden **O(n)**.

Retornos posibles	
OK	Retornando el listado de aerolíneas en <i>valorString</i> .
ERROR	No hay errores posibles.
NO_IMPLEMENTADA	Cuando aún no se implementó.

Formato de retorno del *valorString*:  
`codigo1;nombre1|codigo2;nombre2;`

Por ejemplo:  
`LA14;Latam Airlines|IB01;Iberia|CA10;Copa Airlines`

## 08. Registrar aeropuerto

Retorno **registrarAeropuerto**(String codigo, String nombre);

Descripción: Registra el aeropuerto en el sistema con el código y nombre indicado. El código es el identificador único, el código y nombre no pueden ser vacíos

Retornos posibles	
OK	Si el aeropuerto fue registrado exitosamente.
ERROR	<ol style="list-style-type: none"> <li>1. Si en el sistema ya hay registrados <i>maxAeropuertos</i>.</li> <li>2. Si código o nombre son vacíos o null</li> <li>3. Si ya existe un aeropuerto con ese código.</li> </ol>
NO_IMPLEMENTADA	Cuando aún no se implementó.

Esta operación no tiene restricciones de eficiencia.

## 09. Registrar Conexión

Retorno **registrarConexion**(String codigoAeropuertoOrigen, String codigoAeropuertoDestino, double kilometros);

Descripción: Registra una conexión en el sistema desde el aeropuerto *codigoAeropuertoOrigen* al aeropuerto *codigoAeropuertoDestino*.

Retornos posibles	
OK	Si la conexión fue registrada exitosamente.
ERROR	<ol style="list-style-type: none"> <li>1. Si kilómetros es menor o igual a 0.</li> <li>2. Si alguno de los parámetros String es vacío o null</li> <li>3. Si no existe el aeropuerto de origen</li> <li>4. Si no existe el aeropuerto de destino</li> <li>5. Si ya existe una conexión entre el origen y el destino</li> </ol>
NO_IMPLEMENTADA	Cuando aún no se implementó.

**Nota:** Se considera que las conexiones no son navegables en ambos sentidos. O sea que, si existe una conexión entre los aeropuertos A al B, puede no existir de B al A.

## 10. Registrar vuelo

Retorno **registrarVuelo**(String codigoAeropuertoOrigen, String codigoAeropuertoDestino ,String codigoDeVuelo, double combustible, double minutos, double costoEnDolares, String codigoAerolinea);

Descripción: Registra un nuevo vuelo en el sistema, debe existir la conexión entre el aeropuerto de origen y el destino. Puede haber varios vuelos entre el aeropuerto de origen y el destino. El identificador de un vuelo es el *aeropuertoOrigen*, el *aeropuertoDestino* y el *codigoDeVuelo*. En cada conexión no puede haber dos vuelos con el mismo código.

Retornos posibles	
OK	Si el vuelo se registró exitosamente.
ERROR	<ol style="list-style-type: none"> <li>1. Si alguno de los parámetros double es menor o igual a 0.</li> <li>2. Si alguno de los parámetros String es vacío o null.</li> <li>3. Si no existe el aeropuerto de origen.</li> <li>4. Si no existe el aeropuerto de destino.</li> <li>5. Si no existe la aerolínea indicada.</li> <li>6. Si no existe una conexión entre origen y destino</li> <li>7. Si ya existe un vuelo con ese código en esa conexión.</li> </ol>
NO_IMPLEMENTADA	Cuando aún no se implementó.

## 11. Aeropuertos por Cantidad de escalas

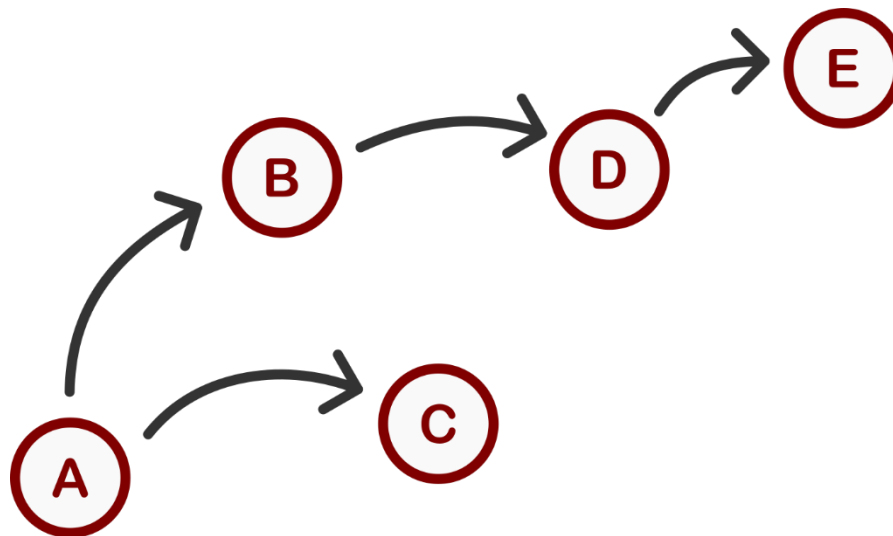
Retorno **listadoAeropuertosCantDeEscalas**(String codigoAeropuertoDeOrigen, int cantidad, String codigoAerolinea);

Descripción: Dado un aeropuerto de origen se debe retornar en el valorString los datos de los aeropuertos (ordenados por código creciente) a los que se pueda llegar realizando hasta la cantidad de escalas indicada por parámetro y mediante vuelos de la aerolínea indicada.

Retornos posibles	
OK	Retorna en <i>valorString</i> los datos de aeropuertos a los que se pueda llegar con hasta "cantidad" de escalas.
ERROR	<ol style="list-style-type: none"> <li>1. Si la cantidad es menor que cero.</li> <li>2. Si el aeropuerto no está registrado en el sistema.</li> <li>3. Si la aerolínea no está registrada en el sistema.</li> </ol>
NO_IMPLEMENTADA	Cuando aún no se implementó.

Formato de retorno del *valorString*:

codigoAeropuerto1;nombreAeropuerto1|codigoAeropuerto2;nombreAeropuerto2



## 12. Viaje de costo mínimo en kilómetros

Retorno **viajeCostoMinimoKilometros**(String codigoAeropuertoOrigen, String codigoAeropuertoDestino);

Descripción: Retorna el camino más corto en kilómetros que podría realizar un pasajero para ir desde el aeropuerto de origen al de destino.

Retornos posibles	
OK	Si el camino pudo ser calculado exitosamente. Retorna en <i>valorEntero</i> la cantidad de kilómetros total del camino. Retorna en <i>valorString</i> el camino desde el aeropuerto de origen al de destino incluidos.
ERROR	<ol style="list-style-type: none"> <li>1. Si alguno de los códigos es vacío o null.</li> <li>2. Si no hay camino entre el origen y el destino.</li> </ol>
NO_IMPLEMENTADA	Cuando aún no se implementó.

Formato de retorno del *valorString*:

codigoAeropuertoOrigen;nombreAeropuertoOrigen|codigoAeropuerto1;nombreAeropuerto1|codigoAeropuerto2;nombreAeropuerto2|codigoAeropuertoDestino;nombreAeropuertoDestino

### 13. Viaje de costo mínimo en minutos

Retorno **viajeCostoMinimoEnMinutos**(String codigoAeropuertoOrigen, String codigoAeropuertoDestino);

Descripción: Retorna el camino menos costoso en tiempo que podría realizar un pasajero para ir del aeropuerto de origen al de destino.

Retornos posibles	
OK	Si el camino pudo ser calculado exitosamente. Retorna en <i>valorEntero</i> la cantidad de dólares total del camino. Retorna en <i>valorString</i> el camino desde el aeropuerto de origen al de destino incluidos.
ERROR	1. Si alguno de los códigos es vacío o <i>null</i> . 2. Si no hay camino entre el origen y el destino.
NO_IMPLEMENTADA	Cuando aún no se implementó.

Formato de retorno de *valorString*:

codigoAeropuertoOrigen;nombreAeropuertoOrigen|codigoAeropuerto1;nombreAeropuerto1|codigoAeropuerto2;nombreAeropuerto2|codigoAeropuertoDestino;nombreAeropuertoDestino

Ejemplo:

A001;Aeropuerto de Carrasco|A002;Aeropuerto de Ezeiza|A003;GRUAirport

### Información importante

- Se deberán **respetar los formatos de retorno** dados para las operaciones que devuelven datos.
- Está **terminantemente prohibido** el uso de clases de Java tales como **ArrayList**.
- **Ninguna** de las operaciones debe 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 valorará la selección adecuada de las estructuras para modelar el problema y la eficiencia en cada una de las operaciones.** Deberá aplicar la metodología vista en el curso.
- El proyecto será implementado en lenguaje JAVA sobre una interfaz Sistema que se publicará en el sitio de la materia en aulas.ort.edu.uy (El uso de esta interfaz es obligatorio).
- El proyecto entregado debe compilar y ejecutar correctamente en IntelliJ IDEA.
- No se contestarán dudas sobre el obligatorio en las 48 horas previas a la entrega.
- **Defensa:** la defensa del obligatorio será en fecha y formato a coordinar, consultar con el docente. Se publicará en Aulas la fecha y horario específico con antelación. La no asistencia a la defensa implica la pérdida de todos los puntos.
- **Entrega (Se debe subir un único zip o rar con):**
  - Proyecto con las operaciones implementadas respetando las buenas prácticas y estándares vistos en el curso. **La implementación debe respetar los órdenes solicitados.** Incluir en las clases los nombres y números es estudiante como comentario.
  - **Documentaciones:** Entregar un documento PDF con la información de: nombres, números de estudiante, grupo y la justificación del cumplimiento de los órdenes solicitados.




## RECORDATORIO: IMPORTANTE PARA LA ENTREGA

- **Obligatorios**

La entrega de los obligatorios será en formato digital online, a excepción de algunas materias que se entregarán en Bedelía y en ese caso recibirá información específica en el dictado de esta.

Los principales aspectos para destacar sobre la **entrega online de obligatorios** son:

1. Ingresa al sistema de Gestión.
2. En el menú, selecciona el ítem “Evaluaciones” y la instancia de evaluación correspondiente, que figura bajo el título “Inscripto”.
3. Para iniciar la entrega hacer clic en el ícono: 
4. Ingresa el número de estudiante de cada uno de los integrantes y hace clic en “Agregar”. El sistema confirmará que los integrantes estén inscriptos al obligatorio y, de ser así, mostrará el nombre y la fotografía de cada uno de ellos. Una vez agregados todos los integrantes, hace clic en “Crear equipo”.

**Cualquier integrante podrá:**

- **Modificar la integración del equipo.**
  - **Subir el archivo de la entrega.**
5. Seleccione el archivo que desees entregar. Verificar el nombre del archivo que aparecerá en la pantalla y hacer clic en “Subir” para iniciar la entrega. Cada equipo (hasta 2 estudiantes del mismo dictado) debe entregar **un único archivo en formato zip o rar** (los documentos de texto deben ser formato pdf, y deben ir dentro del zip o rar). El archivo a subir debe tener **un tamaño máximo de 40mb**
- Cuando el archivo quede subido, se mostrará el nombre generado por el sistema (1), el tamaño y la fecha en que fue subido.
6. El sistema enviará un e-mail a todos los integrantes del equipo informando los detalles del archivo entregado y confirmando que la entrega fue realizada correctamente.
  7. Podes cerrar la pestaña de entrega y continuar utilizando Gestión o salir del sistema.
  8. La **hora tope para subir el archivo será las 21:00** del día fijado para la entrega.
  9. La entrega se podrá realizar desde cualquier lugar (ej. hogar del estudiante, laboratorios de la Universidad, etc).
  10. Aquellos de ustedes que presenten alguna dificultad con su inscripción o tengan inconvenientes técnicos, por favor contactarse con el Coordinador o Coordinación adjunta **antes de las 20:00hs.** del día de la entrega, a través de los mails [gervaz@ort.edu.uy](mailto:gervaz@ort.edu.uy) , [alamon@ort.edu.uy](mailto:alamon@ort.edu.uy), y [fernandez\\_ma@ort.edu.uy](mailto:fernandez_ma@ort.edu.uy), o telefónicamente al 29021505 - int 1156 (de 8:00 a 14:00 hs) y 1436 (de 17:30 a 20:00 hs).

Si tuvieras una situación particular de fuerza mayor, debes contactarte con suficiente antelación al plazo de entrega, con la secretaría docente a ([gervaz@ort.edu.uy](mailto:gervaz@ort.edu.uy)) o ([paulos@ort.edu.uy](mailto:paulos@ort.edu.uy)).