

<b>EVALUACION</b>	Obligatorio	<b>GRUPO</b>	Todos	<b>FECHA</b>	Marzo 2023
<b>MATERIA</b>	Algoritmos y Estructuras de Datos 2				
<b>CARRERA</b>	Analista Programador – Analista en TI				
<b>CONDICIONES</b>	<ul style="list-style-type: none"> <li>• Puntos: <b>Máximo: 40</b>      Mínimo: 1</li> <li>• <b>Fecha máxima de entrega: 01/06/2023</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 <b>hasta 3 personas</b>.</li> <li>○ Subir el trabajo a Gestión antes de la hora indicada, ver hoja al final del documento: "RECORDATORIO"</li> </ul> </li> </ul>				

## Obligatorio: Trenes

### Introducción

Se desea implementar un software para la gestión de trenes de toda Europa, al gestionar una gran cantidad de líneas y pasajeros debemos ser cuidadosos a la hora de diseñar las estructuras. Para llevar a cabo tal tarea se nos pide mantener un registro de pasajeros, estaciones y conexiones.



Luego de un breve análisis, se creó una interfaz para separar las reglas de negocio de la parte del manejo de servidor donde 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};  
    public int valorEntero;  
    public String valorString;  
    public Resultado resultado;  
}
```

Además, 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 maxEstaciones);

Descripción: Inicializa las estructuras necesarias para representar el sistema especificado, capaz de registrar como máximo la cantidad *maxEstaciones* de estaciones de trenes.

Restricción de eficiencia: no tiene.

Retornos posibles	
OK	Si el sistema pudo ser inicializado exitosamente.
ERROR	1. Si <i>maxEstaciones</i> es menor o igual a 5.
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

### 02. Registrar pasajero

Retorno **registrarPasajero** (String identificadorPasajero, String nombre, int edad);

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

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

Retornos posibles	
OK	El pasajero fue registrado exitosamente.
ERROR	1. Si alguno de los parámetros es vacío o <i>null</i> . 2. Si el identificador no tiene el formato válido. 3. Si ya existe un pasajero registrado con ese identificador.
NO_IMPLEMENTADA	Cuando aún no se implementó.

Formato del identificador: Se requiere el uso de expresiones regulares (a investigar por los estudiantes) para lograr validar el formato del id de pasajero. A partir del identificador podemos saber la nacionalidad del pasajero la cual usaremos más adelante.

Las posibles Nacionalidades son:

- FR: Francia
- DE: Alemania
- UK: Reino Unido
- ES: España
- OT: Otro

Los formatos válidos de código son:

- (CodigoNacionalidad)P.NNN.NNN#N
- (CodigoNacionalidad)PNN.NNN#N

Ejemplos válidos: FR123.456#2, DE1.233.222#5, ES222.333#9, OT3.212.322#2, OT1.232.322#0

Ejemplos inválidos:

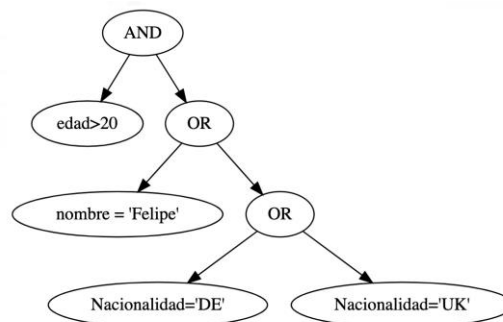
- UY322.222#3: El código de nacionalidad no está dentro de las opciones válidas.
- DE2...322212: Le falta el numeral, tiene varios puntos seguidos.
- DE0.234.223#2: El primero número no puede ser cero.
- US3212: Nacionalidad inválida, formato invalido
- ES029.232#3: Empieza en 0

### 03. Filtrar pasajeros

Retorno **filtrarPasajeros**(Consulta c);

Descripción: Nos interesa poder consultar pasajeros de acuerdo con varios criterios, para dar flexibilidad se le permite al administrador ingresar estas consultas en forma de texto, por ejemplo “[edad>20] AND [nacionalidad=’DE’ OR nacionalidad=’UK’ OR nombre=’Felipe’]”.

A partir de ese texto se crea un árbol binario como el siguiente:



Los nodos pueden ser de 5 tipos, AND/OR (tienen 2 hijos) y el resultado se computa a partir del resultado de sus hijos haciendo la operación booleana correspondiente. Los otros tipos son EDAD\_MAYOR, NOMBRE\_IGUAL, NACIONALIDAD\_IGUAL que solo dan true si se cumple la condición del tipo.

Restricción de eficiencia: Esta operación deberá realizarse en  $O(n*q)$ , donde n es la cantidad de pasajeros y q es la cantidad de nodos de la consulta.

Retornos posibles	
OK	Si la consulta no es vacía. Retorna en <i>valorString</i> el listado de los identificadores ordenados numéricamente en forma creciente de los pasajeros que cumplen las condiciones. En caso de no haber pasajeros que cumplan las condiciones, <i>valorString</i> será un string vacío.
ERROR	1. Si la consulta es vacía
NO_IMPLEMENTADA	Cuando aún no se implementó.

Por ejemplo, valorString del retorno en una consulta válida:

DE234.233#2 | FR1.234.233#2 | ES9.234.233#2

## 04. Buscar Pasajero

Retorno **buscarPasajero**(String identificador);

Descripción: Retorna en valorString los datos del pasajero según el formato indicado. 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 pasajero 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	1. Si el identificador no tiene formato válido. 2. Si no existe un pasajero registrado con ese identificador.
NO_IMPLEMENTADA	Cuando aún no se implementó.

Formato de retorno del valorString:

identificador;nombre;edad;nacionalidad;

Por ejemplo, valorString del retorno en una consulta válida:

DE234.233#2;Ana;32;DE;

## 05. Listar pasajeros por identificador descendente

Retorno **listarPasajerosDescendente()** ;

**Descripción:** Retorna en valorString los datos de todos los pasajeros registrados, ordenados **numéricamente** por identificador en forma descendente.

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

Retornos posibles	
OK	Retornando el listado de pasajeros en <i>valorString</i> . En caso de no haber pasajeros, <i>valorString</i> será un String vacío.
ERROR	No hay errores posibles.
NO_IMPLEMENTADA	Cuando aún no se implementó.

Formato de retorno del valorString:

```
identificador1;nombre1;edad1;nacionalidad1|identifcador2;nombr  
e2;edad2;nacionalidad2
```

Por ejemplo:

```
FR1.234.233#2;Alberto;20;FR|DE234.233#2;Ana;32;DE
```

## 06. Listar pasajeros por identificador ascendente

Retorno `listarPasajerosAscendente()` ;

**Descripción:** Retorna en `valorString` los datos de todos los pasajeros registrados, ordenados **numéricamente** por identificador en forma ascendente.

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

Retornos posibles	
OK	Retornando el listado de pasajeros en <i>valorString</i> . En caso de no haber pasajeros, <i>valorString</i> será un <i>String</i> vacío.
ERROR	No hay errores posibles.
NO_IMPLEMENTADA	Cuando aún no se implementó.

Formato de retorno del `valorString`:

```
identificador1;nombrel;edad1;nacionalidad1|identificador2;nomb  
re2;edad2;nacionalidad2
```

Por ejemplo:

```
DE234.233#2;Ana;32;DE|FR1.234.233#2;Alberto;20;FR|DE5.232.222#  
2;Jorgen;45;DE
```



## 07. Listar pasajeros por nacionalidad

Retorno **listarPasajerosPorNacionalidad**(Nacionalidad tipo);

Descripción: Retorna en valorString los datos de todos los pasajeros registrados con ese tipo. No se requiere que estén ordenados.

Restricción de eficiencia: Esta operación deberá realizarse en orden (k), siendo k la cantidad de pasajeros con dicho tipo.

Retornos posibles	
OK	Devuelve en valorString el listado de pasajeros de ese tipo o un String vacío si no hay pasajeros de ese tipo.
ERROR	1. Si la nacionalidad es null
NO_IMPLEMENTADA	Cuando aún no se implementó.

Formato de retorno del valorString:

identificador1;nombrel1;edad1;nacionalidad1|identificador2;nomb  
re2;edad2;nacionalidad2

Por ejemplo:

FR1.234.233#2;Alberto;32;FR|FR234.233#2;Benjamin;22;FR|  
FR6.234.233#2;Roberto;15;FR

## 08. Registrar estación de tren

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

Descripción: Registra la estación de tren en el sistema con el código y nombre indicado. El código es el identificador único y tiene el formato (LLLNNN), ningún dato puede ser vacío.

Restricción de eficiencia: no tiene.

Retornos posibles	
OK	Si la estación fue registrada exitosamente.
ERROR	1. Si en el sistema ya hay registrados <i>maxEstaciones</i> . 2. Si algún dato es vacío o nulo. 3. Si el código es inválido. 4. Si ya existe una estación con ese código.
NO_IMPLEMENTADA	Cuando aún no se implementó.

Formato de código LLLNNN con las siguientes restricciones:

- L = cualquier letra de la "A" a la "Z" en mayúscula.
- N = cualquier dígito del 0 al 9

Ejemplos válidos:

- LLL000
- BAB010
- AAA123

Ejemplos inválidos:

- LLL00: Largo incorrecto
- AB010: Largo incorrecto
- AAA1234: Largo incorrecto
- aaa123: Letras en minúscula
- A!B456: Carácter no soportado.

## 09. Registrar Conexión

Retorno **registrarConexion**(String codigoEstacionOrigen, String codigoEstacionDestino, int identificadorConexion, double costo, double tiempo, double kilometros, Estado estadoDeLaConexion);

**Descripción:** Registra una conexión en el sistema desde la estación codigoEstacionOrigen a la estación codigoEstacionDestino. Para poder ir de una estación a otra se debe pagar un costo, y tiene un tiempo que es lo que lleva el viaje de una estación a otra.

**Restricción de eficiencia:** no tiene.

Retornos posibles	
OK	Si el camino fue registrado 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 o enum es vacío o null.</li> <li>3. Si alguno de los códigos de estación no es válido.</li> <li>4. Si no existe la estación de origen.</li> <li>5. Si no existe la estación de destino.</li> <li>6. Si ya existe un camino 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 un camino de la estación A al B, puede no existir de B al A. Además, se considera que entre dos estaciones puede haber más de una conexión que te lleva de A a B (se asume que estas se distinguen por su identificador que es un número cualquiera).

Los posibles estados del camino son:

- A: Excelente
- B: Bueno
- C: Malo

## 10. Actualizar camino

Retorno **actualizarCamino**(String codigoEstacionOrigen, String codigoEstacionDestino, int identificadorConexion, double costo, double tiempo, double kilometros, Estado estadoDelCamino);

Descripción: Actualiza una conexión existente en el sistema, debe existir una conexión entre la estación de origen y destino.

Restricción de eficiencia: no tiene.

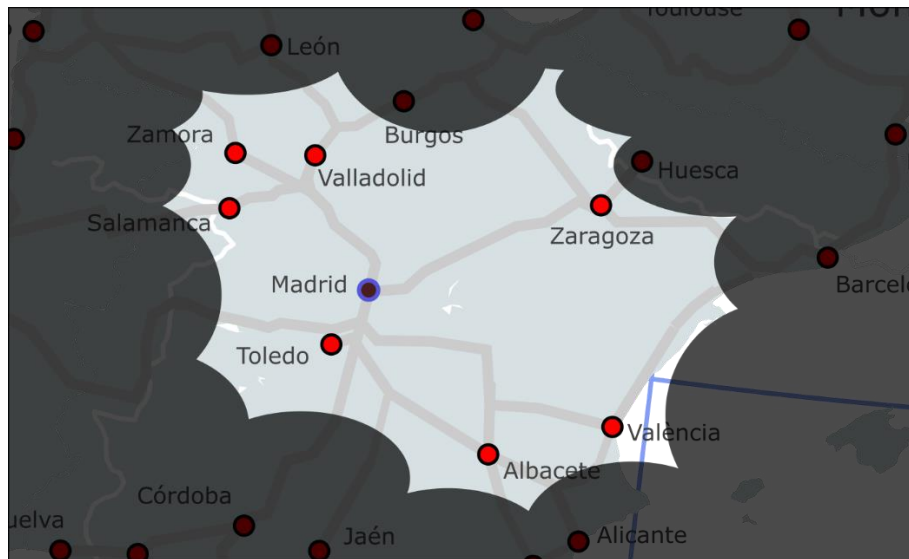
Retornos posibles	
OK	Si el camino se actualizó exitosamente.
ERROR	<ol style="list-style-type: none"> <li>1. Si alguno de los parámetros double es menor o igual a 0.</li> <li>2. Si alguno de los parámetros String o enum es vacío o null.</li> <li>3. Si alguno de los códigos de estacion no es válido.</li> <li>4. Si no existe la estación de origen.</li> <li>5. Si no existe la estación de destino.</li> <li>6. Si no existe la conexión con identificador entre origen y destino.</li> </ol>
NO_IMPLEMENTADA	Cuando aún no se implementó.

## 11. Estaciones por cantidad de trasbordos

Retorno **listadoEstacionesCantTrasbordos** (String codigo, int cantidad);

Descripción: Dado una Estación de origen se debe retornar en el valorString los datos de las Estaciones (ordenados por código creciente) a los que se pueda llegar realizando hasta la cantidad de trasbordos indicada por parámetro.

Restricción de eficiencia: no tiene.



Retornos posibles	
OK	Retorna en <i>valorString</i> los datos de las estaciones a los que se pueda llegar con hasta "cantidad" de transbordos. Si es cero la cantidad de transbordos debe devolver solo la estación de origen.
ERROR	<ol style="list-style-type: none"> <li>1. Si la cantidad es menor que cero.</li> <li>2. Si el código es nulo.</li> <li>3. Si el código de la estación no es válido.</li> <li>4. Si la estación no está registrad en el sistema.</li> </ol>
NO_IMPLEMENTADA	Cuando aún no se implementó.

### Formato de retorno del valor String:

codigoEstacion1;nombreEstacion1|codigoEstacion2;nombreEstacion  
2

## 12. Viaje de costo mínimo en kilómetros

Retorno **viajeCostoMinimoKilometros** (String  
codigoEstacionOrigen, String codigoEstacionDestino);

**Descripción:** Retorna el camino más corto en kilómetros que podría realizar un pasajero para ir de la estación de origen a la de destino. No se pueden tomar en cuenta los caminos con estado “malo”.

**Restricción de eficiencia:** no tiene.

Retornos posibles	
OK	Si el camino pudo ser calculado exitosamente. Retorna en <i>valorEntero</i> la cantidad de kilometros total del camino. Retorna en <i>valorString</i> el camino desde la estación de origen a la de destino incluidos.
ERROR	1. Si alguno de los códigos es vacío o <i>null</i> . 2. Si alguno de los códigos de estacion no es válido. 3. Si no hay camino entre el origen y el destino. 4. Si no existe origen. 5. Si no existe destino.
NO_IMPLEMENTADA	Cuando aún no se implementó.

### Formato de retorno del valor String:

```
codigoEstacionOrigen;nombreEstacionOrigen
codigoEstacion1;nombreEstacion1|
codigoEstacion2;nombreEstacion2
|codigoEstacionDestino;nombreEstacionDestino
```

### 13. Viaje de costo mínimo en costo

Retorno **viajeCostoMinimoDolares**(String codigoEstacionOrigen,  
String codigoEstacionDestino);

Descripción: Retorna el camino menos costoso que podría realizar un pasajero para ir de la estación de origen a la de destino. No se pueden tomar en cuenta los caminos con estado "malo".

Restricción de eficiencia: no tiene.

Retornos posibles	
OK	Si el camino pudo ser calculado exitosamente. Retorna en <i>valorEntero</i> el costo total del camino. Retorna en <i>valorString</i> el camino desde la estación de origen al de destino incluidos.
ERROR	1. Si alguno de los códigos es vacío o <i>null</i> . 2. Si alguno de los códigos de estacion no es válido. 3. Si no hay camino entre el origen y el destino. 4. Si no existe origen. 5. Si no existe destino.
NO_IMPLEMENTADA	Cuando aún no se implementó.

**Formato de retorno del valor String:**

```
codigoEstacionOrigen;nombreEstacionOrigen
codigoEstacion4;nombreEstacion4|
codigoEstacion7;nombreEstacion7
|codigoEstacionDestino;nombreEstacionDestino
```

### 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**.

Los objetos deben se deben comparar utilizando **equals**.

Se debe aplicar el uso de buenas prácticas en los **métodos recursivos**.

**Ninguna** de las operaciones 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 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](http://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.




## 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 la misma.

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

1. Ingresá al sistema de Gestión.
2. En el menú, seleccioná el ítem “Evaluaciones” y la instancia de evaluación correspondiente, que figura bajo el título “Inscripto”.
3. Para iniciar la entrega hacé clic en el ícono: 
4. Ingresá el número de estudiante de cada uno de los integrantes y hacé 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, hacé clic en “Crear equipo”.

**Cualquier integrante podrá:**

- **Modificar la integración del equipo.**
- **Subir el archivo de la entrega.**

5. Seleccioná el archivo que deseás entregar. Verificá el nombre del archivo que aparecerá en la pantalla y hacé clic en “Subir” para iniciar la entrega. Cada equipo (hasta 2 estudiantes) debe entregar **un único archivo en formato zip o rar** (los documentos de texto deben ser pdf, y deben ir dentro del zip o rar). El archivo a subir debe tener **un tamaño máximo de 40mb**

Quando 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. Podés 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, al Coordinador de Cursos ([gervaz@ort.edu.uy](mailto:gervaz@ort.edu.uy)) o Secretario Docente ([paulos@ort.edu.uy](mailto:paulos@ort.edu.uy)).