1. Obligatorio: Empresa de Cruceros

EVALUACION	Obligatorio	GRUPO	TODOS	FECHA	17-18/Oct/2017
MATERIA	**Algoritmos y Estructuras de Datos 1**				
CARRERA	Analista Programador / Analista en Tecnologías de la información				
CONDICIONES	**- Puntos:** Máximo: 50 Mínimo: 1 **- Duración:** Fecha máxima de entrega: 23/11/2017 **IMPORTANTE** - Los grupos deben estar conformados por dos personas. - Inscribirse (sacar la "boleta de entrega"). - Es obligatoria la entrega tanto en bedelía como en Aulas, en una tarea que será habilitada oportunamente por el cuerpo docente. La no entrega de cualquiera de ellas implica la pérdida del curso. - Se debe entregar un conjunto de pruebas en **JUnit.** Estas deben cubrir todos los métodos, tanto los casos con resultado OK, como los con resultado ERROR.				

Introducción

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

Los Cruceros serán registrados en el sistema con la ciudad a la que pertenecen, su nombre (que podrá ser único dentro de la ciudad), una cierta cantidad de habitaciones (capacidad), la categoría (cantidad de estrellas), y una lista de servicios provistos.

Para simplificar la realidad consideraremos que las habitaciones de los Cruceros serán todas unitarias, o sea que cada habitación podrá ser ocupada por un único cliente.

El sistema deberá proveer funcionalidades para que los clientes puedan gestionar sus reservas para los Cruceros, realizar búsquedas por ciudad o por ranking, consultar los servicios de los mismos, etc.

Se define una clase Retorno, la cual se utilizará como tipo de retorno para todas las operaciones del sistema. 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 valor entero, para las operaciones que retornen un número entero.
- Un valor String, para las operaciones que retornen un String, o un valor más complejo (por ejemplo una lista o clase), la cuál será formateada según lo indicado en el Anexo I de este documento.

Se provee: una interfaz llamada ISistema, la cual no podrá ser modificada en ningún sentido, y una clase sistema que la implementa, donde el estudiante deberá completar la

implementación de las operaciones solicitadas. Además, se provee el tipo de dato Retorno, el cual debe ser respetado y se despliega a continuación:

```
public class Retorno {
        enum Resultado {
                OK, ERROR_1, ERROR_2, ERROR_3, ERROR_4, ERROR_5, NO_IMPLEMENTADA
        };
        int valorEntero;
        string valorString;
        Resultado resultado;
}
```

Índice

2. Funcionalidades

1.1.Crear Sistema de Reservas

Firma: Retorno crearSistemaReservas(int cantCiudades);

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

Retornos posibles	
OK	Siempre retorna OK.
ERROR	1. En caso que "cantCiudades" sea menor que cero.
NO_IMPLEMENTADA Cuando aún no se implementó. Es el tipo de retorno por defecto.	

Nota: el sistema debe recibir como parámetro la cantidad de ciudades que va a gestionar, si el parámetro es 0 no se limita esta cantidad, de lo contrario se valida la misma.

1.2.Destruir Sistema de Reservas

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.	

1.3.Registrar Ciudad

Firma: Retorno registrarCiudad(String ciudad);

Descripción: Registra la Ciudad "Ciudad"
Nota: esto se utilizará para la parte 2 del obligatorio para definir la matriz de ciudades (mapa de ciudades)

Retornos posibles	
OK	En caso que la ciudad "Ciudad" haya sido ingresado correctamente en el sistema.
ERROR	1. En caso que la ciudad "Ciudad" ya existe en el sistema.
NO_IMPLEMENTADA Cuando aún no se implementó. Es el tipo de retorno por defecto.	

1.4.Registrar Crucero

Firma: Retorno registrarCrucero(String ciudad, String Nombre, int Estrellas,int Capacidad);

Descripción: Registra el Crucero de nombre “Nombre”, en la ciudad “Ciudad”, con cantidad de estrellas “Estrellas”, capacidad “Capacidad” y ranking 0 en el sistema de reservas.

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

Retornos posibles	
OK	En caso que el Crucero haya sido ingresado correctamente en el sistema.
ERROR	1. En caso que “Estrellas” sea menor a 1 o mayor a 5.
	2. En caso que “Capacidad” sea menor a 0.
	3. En caso que ya exista un Crucero de nombre “Nombre” dentro de la ciudad “Ciudad”.
	4. En caso que no exista la ciudad “Ciudad”
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

1.5.Ingresar Servicio

Firma: Retorno ingresarServicio(String ciudad, String crucero, String Servicio);

Descripción: Ingresa el servicio “Servicio” al Crucero “Crucero”.

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 que el servicio “Servicio” haya sido ingresado correctamente en el Crucero “Crucero”.
ERROR	1. En caso que no exista un Crucero “Crucero” registrado dentro de la ciudad “Ciudad”.
	2. En caso que no exista la ciudad “Ciudad”.
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

1.6.Borrar Servicio

Firma: Retorno borrarServicio(String ciudad, String crucero, String Servicio);

Descripción: Borra la primera ocurrencia del servicio “Servicio” del Crucero “Crucero”.

Retornos posibles	
OK	En caso de que el servicio “Servicio” haya sido borrado correctamente del Crucero “Crucero”.
ERROR	1. En caso de que no exista un Crucero “Crucero” registrado dentro de la ciudad “Ciudad”.
	2. En caso de que no exista el servicio “Servicio” registrado en el Crucero “Crucero”.
	3. En caso que no exista la ciudad “Ciudad”.
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

1.7.Realizar Reserva

Firma: Retorno realizarReserva(int cliente, String ciudad, String crucero);

Descripción: Realiza la reserva de una habitación en el Crucero “Crucero” dentro de la ciudad “Ciudad”. En caso de que no haya cupos de habitaciones disponibles, el cliente “cliente” quedará en lista de espera.

Retornos posibles	
OK	En caso de que la reserva haya sido efectuada correctamente en el Crucero “Crucero”.
	En caso de que la reserva haya quedado en lista de espera.
ERROR	1. En caso de que no exista un Crucero de nombre “Crucero” registrado dentro de la ciudad Ciudad”.
	2. En caso que no exista la ciudad “Ciudad”.
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

Nota: La lista de espera debe ser implementada de tal forma que no sea necesario recorrer la estructura para obtener el primer cliente en lista de espera para cierto Crucero en cierta ciudad.

1.8.Cancelar Reserva

Firma: Retorno cancelarReserva(int cliente, String ciudad, String crucero);

Descripción: Cancela la reserva en el Crucero “Crucero” para el cliente “cliente”. 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 habitaciones y luego en la lista de espera. En caso de que el cliente “cliente” tenga más de una reserva para el Crucero “Crucero” se cancelará solo la primera reserva encontrada en el orden establecido.

Retornos posibles	
OK	En caso que la cancelación de la habitación se lleve a cabo correctamente en el Crucero “Crucero”.
ERROR	1. En caso de que no exista un Crucero de nombre “Crucero” registrado en la ciudad “Ciudad”.
	2. En caso de que el cliente “cliente” no tenga ninguna reserva en el Crucero “Crucero” registrado en la ciudad “Ciudad”.
	3. En caso que no exista la ciudad “Ciudad”.
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

1.9.Ingresar Comentario

Firma: Retorno ingresarComentario(String ciudad, String crucero, String Comentario, int Ranking);

Descripción: Ingresa el comentario “Comentario” al Crucero “Crucero”. Cada comentario llevará una calificación para el Crucero. El ranking general del Crucero 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 de que el comentario “Comentario” haya sido ingresado correctamente en el Crucero “Crucero”.
ERROR	1. En caso que el ranking sea menor a 0 o mayor a 5.
	2. En caso que no exista un Crucero de nombre “Crucero” registrado dentro de la ciudad “Ciudad”.
	3. En caso que no exista la ciudad “Ciudad”.
NO_IMPLEMENTADA Cuando aún no se implementó. Es el tipo de retorno por defecto.	

1.10. Listado de Servicios

Firma: Retorno listarServicios(String ciudad, String crucero);
Descripción: Lista todos los servicios prestados por el Crucero “Crucero” de la ciudad “Ciudad” con el siguiente

Formato.
Servicios del Crucero <Crucero> <Ciudad>
1 - <Servicio1>
…
N - <ServicioN>

En caso que no haya servicios registrados en el Crucero “Crucero”, el sistema deberá imprimir:
No existen servicios registrados en el Crucero <Crucero> <Ciudad>

Retornos posibles	
OK	En caso que se imprima el listado correctamente.
ERROR	1. En caso que no exista el Crucero “Crucero” registrado en la ciudad “Ciudad”.
	2. En caso que no exista la ciudad “Ciudad”.
NO_IMPLEMENTADA Cuando aún no se implementó. Es el tipo de retorno por defecto.	

1.11. Listado de Cruceros por Ciudad

Firma: Retorno listarCrucerosCiudad(String ciudad);

Descripción: Lista todos los Cruceros registrados en la ciudad “Ciudad” ordenados por Nombre con el siguiente formato.

Cruceros en <Ciudad>
<NombreCrucero1> <Estrellas1> <Ranking1>
…
<NombreCruceroN> <EstrellasN> <RankingN>

En caso que no haya ningún Crucero registrado en la ciudad “Ciudad” el sistema deberá imprimir:
No existen Cruceros registrados en <Ciudad>

Nota: La impresión de este listado, por ser muy utilizado, deberá realizarse en el menor tiempo posible.

Retornos posibles	
OK	En caso que se imprima el listado correctamente.
ERROR	1. En caso que no exista la ciudad “Ciudad”.
NO_IMPLEMENTADA Cuando aún no se implementó. Es el tipo de retorno por defecto.	

1.12. Listado de Cruceros por Ranking para una ciudad

Firma: Retorno listarCrucerosRanking(String ciudad);

Descripción: Lista todos los Cruceros registrados en el sistema ordenados por ranking descendente para una ciudad en particular.

El formato de impresión deberá ser el siguiente:

Cruceros ordenados por ranking
<Ciudad>
<Crucero1> - <Ranking1>
…
<CruceroN> - <RankingN>

En caso que no exista ningún Crucero registrado en el sistema para esa ciudad, se deberá imprimir: No hay registros de Cruceros en el sistema.

Retornos posibles	
OK	En caso que se imprima el listado correctamente.
ERROR	1. En caso que no exista la ciudad “Ciudad”.
NO_IMPLEMENTADA Cuando aún no se implementó. Es el tipo de retorno por defecto.	

1.13. Listado de Cruceros por Ranking

Firma: Retorno listarCrucerosRanking();

Descripción: Lista todos los Cruceros registrados en el sistema ordenados por ranking descendente.

El formato de impresión deberá ser el siguiente:

Cruceros ordenados por ranking
<Ciudad1> - <Crucero1> - <Ranking1>
…
<CiudadN> - <CruceroN> - <RankingN>

En caso que no exista ningún Crucero registrado en el sistema, se deberá imprimir: No hay registros de Cruceros en el sistema.

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.	

1.14. Listado de Comentarios

Firma: Retorno listarComentarios(String ciudad, String crucero);

Descripción: Lista todos comentarios ingresados para el Crucero “Crucero” dentro de la ciudad “Ciudad”. Los comentarios más recientes deberán aparecer primeros en el listado.

El formato deberá ser el siguiente:

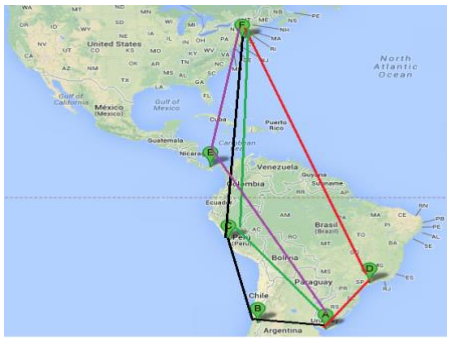
N - <Comentario N> - <RankingN>
…
1 - <Comentario 1> - <Ranking1>

En caso que no existan comentarios para el Crucero “Crucero” el sistema deberá imprimir: No se han agregado comentarios al Crucero <Crucero> <Ciudad>

Retornos posibles	
OK	En caso que se imprima el listado correctamente.
ERROR	1. En caso que no exista el Crucero “Crucero” registrado en la ciudad “Ciudad”.
	2. En caso que no exista la ciudad “Ciudad”.
NO_IMPLEMENTADA Cuando aún no se implementó. Es el tipo de retorno por defecto.	

2. Ejercicio Complementario

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



	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:

2.1.Cargar Matriz de Distancias

Firma: Retorno CargarDistancias(int[][] Ciudades)

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.

2.2.Camino más corto

Firma: Retorno BuscarCamino (int [][] M, 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. La 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. No se aceptará Tests con nombres Test1, Test2, etc.

Información importante

Puntaje mínimo/máximo: 1/50
Los obligatorios se realizan por equipos de **2 estudiantes.**
Plazo máximo de segunda entrega (con boleta): **23/11/2017 – 23:00.**
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 se publicará en el sitio de la
materia en aulas.ort.edu.uy (El uso de esta interfaz es obligatorio). Las pruebas deben codificarse en el framework de pruebas unitarias **JUnit.**

3. Anexo 1 – Retorno de tipos complejos (listas, objetos)

Para las operaciones en las que se debe retornar un tipo complejo (varios valores, o una colección de valores), se define el siguiente formato de manera de serializar el valor y encapsularlo en un único String.

Si el valor a retornar es una colección de datos, se separarán cada ítem de la colección por un carácter "|".

Si el valor a retornar es un tipo complejo y tiene más de un atributo (por ejemplo la CI y nombre), se separarán ambos valores por un carácter ";"

Ejemplos:

Retornar la CI y el nombre de una persona:

32551567;Fernando

Retornar una colección de nombres:

Fernando|Esteban|Fabián

Retornar una colección de personas, con sus CI y nombres:

32551567;Fernando|1234567;Esteban|98765432;Fabián