Tutor: Rodrigo Bentancor

Algoritmos y Estructuras de Datos 2 – Obligatorio – CTC ORT Rosario– Octubre 2018

Reserva de Restaurantes

1. Introducción

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

Los restaurantes 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 mesas (capacidad), la categoría (puntaje), y una lista de servicios provistos.

Para que sea más simple el desarrollo consideraremos que las mesas de los restaurantes serán todas unitarias, o sea que cada mesa podrá ser ocupada por un único cliente.

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

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

Sistema	**public class Sistema{ enum TipoRet{OK,ERROR_1, ERROR_2, ERROR_3, NO_IMPLEMENTADA};** **/*Aquí introduzca la información que estime conveniente*/** **}**

Pueden definirse tipos de datos (clases) auxiliares.

2. Funcionalidades

2.1. Crear Sistema de Reservas

Firma: `TipoRet 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 Reservas

Firma: `TipoRet 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 Restaurante

Firma: `TipoRet registrarRestaurante(String Ciudad, String Nombre, int Puntaje, int Capacidad);`

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

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

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

2.4. Ingresar Servicio

Firma: `TipoRet ingresarServicio(String Ciudad, String Restaurante, String Servicio);`

Descripción: Ingresa el servicio "Servicio" al restaurante "Restaurante".
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 restaurante "Restaurante".
ERROR	• 1. En caso que no exista un restaurante "Restaurante" registrado dentro de la ciudad.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

2.5. Borrar Servicio

Firma: `TipoRet borrarServicio(String Ciudad, String Restaurante, String Servicio);`

Descripción: Borra la primera ocurrencia del servicio "Servicio" del Restaurante "Restaurante".

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

2.6. Ingresar Comentario

Firma: `TipoRet ingresarComentario(String Ciudad, String Restaurante, String Comentario, int Ranking);`

Descripción: Ingresa el comentario "Comentario" al restaurante "Restaurante". Cada comentario llevará una calificación para el restaurante. El ranking general del restaurante 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 restaurante "Restaurante".
ERROR	• 1. En caso que el ranking sea menor a 0 o mayor a 5. • 2. En caso que no exista un restaurante de nombre "Restaurante" registrado dentro de la ciudad "Ciudad".
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

2.7. Realizar Reserva

Firma: `TipoRet realizarReserva(int cliente, String Ciudad, String Restaurante);`

Descripción: Realiza la reserva de una mesa en el restaurante "Restaurante" dentro de la ciudad "Ciudad". En caso de que no haya cupos de mesas disponibles, el cliente "cliente" quedará en lista de espera.

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

2.8. Cancelar Reserva

Firma: `TipoRet cancelarReserva(int cliente, String Ciudad, String Restaurante);`

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

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

2.9. Listado de Servicios

Firma: `TipoRet listarServicios(String Ciudad, String Restaurante);`

Descripción: Lista todos los servicios prestados por el restaurante “Restaurante” de la ciudad “Ciudad” con el siguiente formato.

Servicios del Restaurante <Restaurante> <Ciudad>

1 - <Servicio1>
…
N - <ServicioN>

En caso que no haya servicios registrados en el restaurante “Restaurante”, el sistema deberá imprimir:

No existen servicios registrados en el restaurante <Restaurante> <Ciudad>

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

2.10. Listado de Restaurantes por Ciudad

Firma: `TipoRet listarRestaurantesCiudad(String Ciudad);`

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

Restaurantes en <Ciudad>

<NombreRestaurante1> <Puntaje1> <Ranking1>
…
<NombreRestauranteN> <PuntajeN> <RankingN>

En caso que no haya ningún restaurante registrado en la ciudad “Ciudad” el sistema

deberá imprimir: No existen restaurantes registrados en <Ciudad>

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

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.

2.11. Listado de Restaurantes por Ranking

Firma: `TipoRet listarRestaurantesRanking();`

Descripción: Lista todos los restaurantes registrados en el sistema ordenados por ranking descendente. El formato de impresión deberá ser el siguiente:

Restaurantes ordenados por ranking

<Ciudad1> - <Restaurante1> - <Ranking1>
…
<CiudadN> - <RestauranteN> - <RankingN>

En caso que no exista ningún restaurante registrado en el sistema, se deberá imprimir:

No hay registros de restaurantes 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.

2.12. Listado de Comentarios

Firma: `TipoRet listarComentarios(String Ciudad, String Restaurante);`

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

El formato deberá ser el siguiente:

Comentarios sobre el restaurante <Restaurante><Ciudad>

N - <Comentario N> - <RankingN>
…

1 - <Comentario 1> - <Ranking1>

En caso que no existan comentarios para el restaurante “Restaurante” el sistema deberá imprimir:

No se han agregado comentarios al restaurante <Restaurante> <Ciudad>

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

2.13. Listado de Espera

Firma: `TipoRet listarEspera(String Ciudad, String Restaurante);`

Descripción: Lista los clientes en lista de espera para el restaurante “restaurante” dentro de la ciudad “Ciudad”. Los clientes deberán imprimirse en el orden que serán considerados para tomar una posible mesa 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

restaurante <Restaurante>

<Ciudad>

1 - <CedulaCliente1>
…
N - <CedulaClienteN>

En caso que no existan clientes en la lista de espera se deberá imprimir en pantalla:

No existen reservas pendientes para el restaurante <Restaurante> <Ciudad>

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

Información importante

- Los obligatorios se realizan en equipos de **2 estudiantes a 3 estudiantes.**
- Se deberán respetar los formatos de impresión 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 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.
- Deberá entregar las copias en CD: conteniendo el código fuente y la documentación (en pdf) que entregó impresa.
- Deberá entregar impreso: **La las pre y post condiciones de los métodos solicitados, un 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 y las pruebas realizadas y el resultado obtenido.**
- El proyecto será implementado en lenguaje JAVA sobre una interfaz que se publicará en el sitio de la materia (El uso de esta interfaz es obligatorio).