UNIVERSIDAD ORT
Uruguay

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel 902 15 05 Fax 908 13 70

EVALUACION	**Obligatorio – v1.1**	**GRUPO**	**Todos**	**FECHA**	**13/09/2018**
MATERIA	**Algoritmos y Estructuras de Datos 2**				
CARRERA	**Analista Programador – ATI**				
CONDICIONES	- Puntos: Máximo: 40 Mínimo: 1 - Fecha máxima de entrega: 9/11/2018 LA ENTREGA SE REALIZA EN FORMA ONLINE EN ARCHIVO NO MAYOR A 40MB EN FORMATO ZIP, RAR O PDF. **IMPORTANTE:** - Inscribirse - Formar grupos de hasta dos personas. - Subir el trabajo a Gestión antes de la hora indicada, ver hoja al final del documento: "RECORDATORIO"				

Obligatorio: Monitorización de la red

Introducción

Una compañía de cable local desea monitorizar su infraestructura y tendido de red a lo largo de la ciudad de Montevideo, pudiendo identificar rápidamente áreas afectadas por cortes del servicio o problemas en la señal.

La red de la empresa se constituye por tres tipos de emisores/receptores:

- Un servidor central que emite la señal de cable hacia toda la red.
- Un conjunto de nodos, a partir de los cuales se distribuye la señal hacia las canaleras u otros nodos.
- Las canaleras que están en el domicilio de cada afiliado al servicio, las cuales reciben la señal de cable de los nodos que están esparcidos por la ciudad.

Cabe considerar que el servidor central no puede conectarse directamente con una canalera, sino que siempre debe pasar por al menos un nodo.

Para lograr la monitorización, las canaleras envían información del estado del servicio al servidor central, y a su vez el servidor central se comunica con los nodos de distribución para consultar su estado.

Cada nodo podrá tener cableado hacia otro nodo, una canalera, o al servidor central. Ese cableado, según su calidad, deberá tener un índice de pérdida de calidad, que indicará cuál es la pérdida de calidad que sufre la señal cuando pasa por ese cable. Idealmente, la señal que llega a una canalera deberá ser de la mejor calidad posible, siempre que se pueda elegir entre diferentes caminos posibles.

El obligatorio plantea entonces una serie de operaciones que permitirán al sistema obtener información de la red y brindarla a un operador para que pueda tomar acciones, como por ejemplo saber por donde se debe rutear la señal para mejorar la calidad, o enviar una cuadrilla de técnicos a reparar un nodo o área afectada por una caída de algún punto de la red.

Generalidades

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 proveen los siguientes tipos de datos que deberán ser respetados.

Sistema	`public class Sistema{`
	
	`/* Aquí van las operaciones del sistema */`
	
	`}`

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

Pueden definirse tipos de datos (clases) auxiliares.

La clase sistema NO PODRÁ SER UN SINGLETON. Debe ser una clase **instanciable**.

Funcionalidades

1. Operaciones globales

1.1. Inicializar Sistema

Firma: `Retorno inicializarSistema (int maxPuntos, coordX, coordY);`

Descripción: Inicializa las estructuras necesarias para representar el sistema especificado, capaz de albergar como máximo *`maxPuntos`* puntos diferentes en el mapa. Como punto se entiende que puede ser tanto un nodo, una canalera, o el servidor central (que será siempre 1).

CoordX y *coordY* son las coordenadas de la ubicación geográfica del servidor central en el mapa.

Restricción de eficiencia: no tiene.

Retornos posibles	
OK	• Si el sistema pudo ser inicializado exitosamente.
ERROR	• 1. Si *`maxPuntos`* es menor o igual a 0.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

1.2. Destruir Sistema

Firma: `Retorno destruirSistema();`

Descripción: Destruye el sistema de todos sus elementos y estructuras, liberando la memoria utilizada.

Restricción de eficiencia: no tiene.

Retornos posibles	
OK	• Siempre retorna OK.
ERROR	• No hay errores posibles.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

2. Operaciones relativas a los afiliados

2.1. Registrar afiliado

Firma: `Retorno registrarAfiliado(String cedula, String nombre, String email);`

Descripción: Registra el afiliado con sus datos. La CI identifica al afiliado, y debe tener el formato N.NNN.NNN-N.

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

Retornos posibles	
OK	• Si el afiliado pudo ser registrado exitosamente.
ERROR	• 1. Si la CI no es una cédula con formato válido. • 2. Si la dirección de email no es una dirección válida. • 3. Si ya existe un afiliado con esa CI registrado.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

Se recomienda el uso de **expresiones regulares** para lograr validar los formatos de CI y email. Ver Anexo con links de interés.

2.2. Buscar afiliado

Firma: `Retorno buscarAfiliado(String CI);`

Descripción: Retorna en valorString los datos del afiliado con el formato “CI; nombre; email”. Además, en el campo valorEntero de la clase Retorno, deberá la cantidad de elementos que recorrió durante la búsqueda en sus estructuras.

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

Retornos posibles	
OK	• Si el afiliado se encontró. • Retorna en valorString los datos del afiliado. • Retorna en valorEntero la cantidad de elementos recorridos durante la búsq.
ERROR	• 1. Si la CI no es una cédula válida. • 2. Si no existe un afiliado registrado con esa CI en el sistema.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel 902 15 05 Fax 908 13 70

Formato de retorno del valor String:
3.701.515-1;Ana;ana@mail.com

2.3. Listar todos los afiliados

Firma: `Retorno listarAfiliados();`

Descripción: Retorna en valorString los datos de todos los afiliados registrados, ordenados en forma creciente por CI.

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

Retornos posibles	
OK	• Si se pudo listar los afiliados correctamente.
ERROR	• No hay errores posibles.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

Formato de retorno del valor String:
3.701.515-1;Ana;ana@mail.com|3.702.829-5;Omar;omarejo@adinet.com.uy

3. Operaciones relativas a la red

3.1. Registrar canalera

Firma: `Retorno registrarCanalera(String chipid, String CIafiliado, Double coordX, Double coordY);`

Descripción: Registra la canalera asociada al afiliado, en las coordenadas *coordX, coordY* en el sistema.

Retornos posibles	
OK	• Si la canalera fue registrada exitosamente.
ERROR	• 1. Si en el sistema ya hay registrados *cantPuntos* puntos. • 2. Si ya existe un punto en las coordenadas *coordX*, *coordY* del sistema. • 3. Si el afiliado no existe.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

Esta operación no tiene restricciones de eficiencia.

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel 902 15 05 Fax 908 13 70

3.2. Registrar nodo

Firma: `Retorno registrarNodo(String nodoid, Double coordX, Double coordY);`

Descripción: Registra el nodo en las coordenadas *coordX, coordY* en el sistema.

Retornos posibles	
OK	• Si el nodo fue registrado exitosamente.
ERROR	• 1. Si en el sistema ya hay registrados *cantPuntos* puntos. • 2. Si ya existe un punto en las coordenadas *coordX*, *coordY* del sistema.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

Esta operación no tiene restricciones de eficiencia.

3.3. Registrar tramo

Firma: `Retorno registrarTramo(Double coordXi, Double coordYi, Double coordXf, Double coordYf, int perdidaCalidad);`

Descripción: Registra un tramo en el sistema desde la coordenada inicio (*coordXi, coordYi*) hasta la coordenada destino (*coordXf, coordYf),* con un peso *perdidaCalidad.*

Nota: Se considerará que los tramos son navegables en ambos sentidos. O sea que si agregamos el tramo para ir del punto A al punto B, también se podrá navegar del punto B al punto A.

Retornos posibles	
OK	• Si el tramo pudo ser registrado exitosamente.
ERROR	• 1. Si *peso* es menor o igual a 0. • 2. Si no existe *coordi* o *coordf*. • 3. Si ya existe un tramo registrado desde *coordi a coordf*. • 4. Si se está intentando conectar una canalera con el servidor central.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

Esta operación no tiene restricciones de eficiencia.

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel 902 15 05 Fax 908 13 70

3.4. Modificar calidad tramo

Firma: `Retorno modificarTramo(Double coordXi, Double coordYi, Double coordXf, Double coordYf, int nuevoValorPerdidaCalidad);`

Descripción: Modifica el peso *perdidaCalidad* del tramo dado por las coordenadas de inicio y fin del tramo.

Nota: Se considerará que los tramos son navegables en ambos sentidos. O sea que si agregamos el tramo para ir del punto A al punto B, también se podrá navegar del punto B al punto A.

Retornos posibles	
OK	• Si el tramo pudo ser modificado exitosamente.
ERROR	• 1. Si *peso* es menor o igual a 0. • 2. Si no existe un tramo entre las coordenadas dadas.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

Esta operación no tiene restricciones de eficiencia.

3.5. Consultar la calidad señal que llega a una canalera

Firma: `Retorno calidadCanalera(Double coordX, Double coordY);`

Descripción: Calcula el mínimo índice acumulado de pérdida de calidad desde el servidor central hasta una canalera dada por sus coordenadas X e Y. De todos los caminos posibles, se deberá elegir el que tiene la menor pérdida de calidad acumulada por todo el cableado recorrido, y retornar dicho valor.

Retornos posibles	
OK	• Si el costo pudo ser calculado exitosamente. • Retorna en valorEntero la pérdida acumulada de todo el camino.
ERROR	• 1. Si la canalera de coordenadas X e Y no existe. • 2. Si no se encuentra un camino desde el servidor central a la canalera.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

Esta operación no tiene restricciones de eficiencia.

3.6. Calcular nodos críticos

Firma: `Retorno nodosCriticos();`

Descripción: Dada la red, se deberán identificar cuales nodos son críticos. Un nodo crítico es uno que si deja de funcionar, hay otros nodos que perderían su conectividad con el servidor central. Para este punto se deberá tomar en cuenta solamente la red inducida por los nodos (y el servidor central), no tomando en cuenta las canaleras, ya que las canaleras conectadas a cualquier nodo, obviamente perderán la señal si éste deja de funcionar.
En valorString se deberá devolver una lista de los nodos que son críticos.

Retornos posibles	
OK	• Si se pudo calcular correctamente.
ERROR	• Nunca retorna error.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

Formato de retorno del valor String:
nodoId1|nodoId2

Esta operación no tiene restricciones de eficiencia.

3.7. Dibujar mapa

Firma: `Retorno dibujarMapa();`

Descripción: Esta operación deberá mostrar en un mapa de Google Maps la red formada por los nodos y el servidor central.

Retornos posibles	
OK	• Si se pudo mostrar el mapa correctamente.
ERROR	• Nunca retorna error.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

Esta operación no tiene restricciones de eficiencia.

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel 902 15 05 Fax 908 13 70

Información importante

- Se deberán **respetar los formatos de retorno** dados para las operaciones que devuelven datos.
- **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 ISistema 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 Eclipse.
- No se contestarán dudas sobre el obligatorio en las 48 horas previas a la entrega.
- No se contestarán dudas a través del mail del docente. Las preguntas se deberán hacer en el foro de consultas de aulas.

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel 902 15 05 Fax 908 13 70

Anexo I: Formatos de retorno

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

Anexo II: Información útil

Expresiones regulares:
http://www.mkyong.com/regular-expressions/how-to-validate-email-address-with-regular-expression/
http://regexpal.com/

Crear mapas de Google Maps con marcadores:
https://developers.google.com/maps/documentation/staticmaps/?csw=1
Ejemplo:
http://maps.googleapis.com/maps/api/staticmap?center=Montevideo,Uruguay&zoom=13&size=1200x600&maptype=roadmap&markers=color:blue%7Clabel:1%7C-34.90,-56.16&markers=color:red%7Clabel:2%7C-34.91,-56.17&markers=color:green%7Clabel:3%7C-34.905,-56.19&sensor=false

Parsear un string:
http://stackoverflow.com/questions/3481828/how-to-split-a-string-in-java