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**	**GRUPO**	**Todos**	**FECHA**	**26/09/2016**
MATERIA	**Algoritmos y Estructuras de Datos 2**				
CARRERA	**Analista Programador - ATI**				
CONDICIONES	**- Puntos:** Máximo: 50 Mínimo: 1 **- Fecha máxima de entrega:** 23/11/2016 **- Horario de Coordinación adjunta para entregas:** de lunes a jueves de 8:00 a 12:30 y de 17:30 a 19:00 hs. **IMPORTANTE** - Los grupos deben estar conformados por **dos personas**. - Inscribirse (sacar la "boleta de entrega"). - Entregar en carpeta con elástico. - Entregar 2 copias en cd, donde se incluya la documentación en pdf con foto de los integrantes. - No entregue el código impreso. - Etiquetar documentación y cd con nombre, nº estudiante, carrera, grupo, materia y docente				

Obligatorio :: Gestión de Data-Centers

Introducción

Una empresa de tecnología que tiene varios data-centers (DC) en américa está interesada en contar con un sistema que registre sus DC y los de la competencia, así como las conexiones entre éstos.

Un DC tiene una empresa propietaria, una ubicación, y una capacidad de análisis de información.

Si un DC necesita procesar información (o simplemente almacenarla) y no tiene la capacidad necesaria, puede enviarla a otro DC para que lo procese. El DC al que se envíe la información para procesar será elegido dependiendo de:

- El costo de transmisión de la información.
- Que el DC de destino tenga esa capacidad
- El costo de procesamiento del DC donde irá. Un DC de la misma empresa no tendrá costos, pero sí tendrá costo adicional si el procesamiento lo hace un DC de la competencia.

El sistema deberá contar con un mapa en donde se podrán ver las principales ciudades del continente y los diferentes DC. Además, será el encargado de decidir a qué DC serán enviados los pedidos de procesamiento siguiendo las reglas especificadas.

Finalmente, cada cierto tiempo será necesario hacer mantenimiento sobre los DC y las conexiones entre estos, por lo que el sistema deberá dar un reporte de cuáles son los DC y las conexiones mínimas que deben permanecer funcionando para poder mantener la conexión entre todos los puntos con un costo total mínimo.

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

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{`
	
	`enum TipoPunto {CIUDAD,DATACENTER}; /* USO OPCIONAL */`
	
	`/* 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.

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

Funcionalidades

1. Operaciones globales

1.1. Inicializar Sistema

Firma: `Retorno inicializarSistema (int cantPuntos);`

Descripción: Inicializa las estructuras necesarias para representar el sistema especificado, capaz de albergar como máximo *cantPuntos* en el mapa. Los puntos del mapa serán las ciudades y data-centers.

Retornos posibles	
OK	• Si el sistema pudo ser inicializado exitosamente.
ERROR	• 1. Si *cantPuntos* 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.

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 data-centers

2.1. Registrar Empresa

Firma: `Retorno registrarEmpresa(String nombre, String direccion, String pais, String email_contacto, String color);`

Descripción: Registra a la empresa dueña de data-centers de nombre “nombre” en el sistema. La operación deberá controlar que el email tenga un formato correcto. El nombre identifica a la empresa. El color recibido será el que represente a la empresa en el mapa del estado del sistema (operación 3.1).

Esta operación deberá realizarse en orden (log n) promedio.

Retornos posibles	
OK	• Si la empresa pudo ser registrado exitosamente.
ERROR	• 1. Si el email *email* no cumple el formato de direcciones de e-mail • 2. Si la empresa de nombre *nombre* ya está registrado en el sistema.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

Se recomienda el uso de **expresiones regulares** para lograr fácilmente el control de los formatos. Ver Anexo con links de interés.

2.2. Registrar ciudad

Firma: `Retorno registrarCiudad(String nombre, Double coordX, Double coordY);`

Descripción: Registra la ciudad de nombre *nombre* y coordenadas *coordX, coordY* en el sistema.

Retornos posibles	
OK	• Si la ciudad 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.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

2.3. Registrar data-center

Firma: `Retorno registrarDC(String nombre, Double coordX, Double coordY, String empresa, int capacidadCPUenHoras, int costoCPUporHora);`

Descripción: Registra un DC de nombre *nombre* en el sistema, el cual está a cargo de la empresa *empresa* y tiene una capacidad de procesamiento dada, a un precio por hora `costoCPUporHora`.

Retornos posibles	
OK	• Si el DC pudo ser registrado exitosamente.
ERROR	• 1. Si en el sistema ya hay registrados *cantPuntos* puntos. • 2. Si *capacidad* es menor o igual a 0. • 3. Si el punto de coordenadas *coordX, coordY* ya está registrado en el sistema. • 4. Si la empresa no existe en el sistema.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

2.4. Registrar tramo

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

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

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*.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

2.5. Eliminar tramo

Firma: `Retorno eliminarTramo(Double coordXi, Double coordYi, Double coordXf, Double coordYf);`

Descripción: Elimina el tramo desde la coordenada inicio (coordXi, coordYi) hasta la coordenada destino (coordXf, coordYf).

Retornos posibles	
OK	• Si el tramo pudo ser eliminado exitosamente.
ERROR	• 1. Si no existen algunos de los puntos *coordi* o *coordf* • 2. Si no existe un tramo registrado desde *coordi a coordf*.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

Nota: Como los tramos son no dirigidos, al eliminar el tramo desde el punto A al punto B, también deberá eliminarse el tramo del punto B al punto A.

2.6. Eliminar punto del mapa

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

Descripción: Elimina el punto (coordXi, coordYi) del mapa. El punto puede ser un dc o una ciudad. Si existen tramos conectados con el punto, deberán ser eliminados también.

Retornos posibles	
OK	• Si el punto pudo ser eliminado exitosamente.
ERROR	• 1. Si no existe el punto de coordenadas *coordX, coordY* en el mapa
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

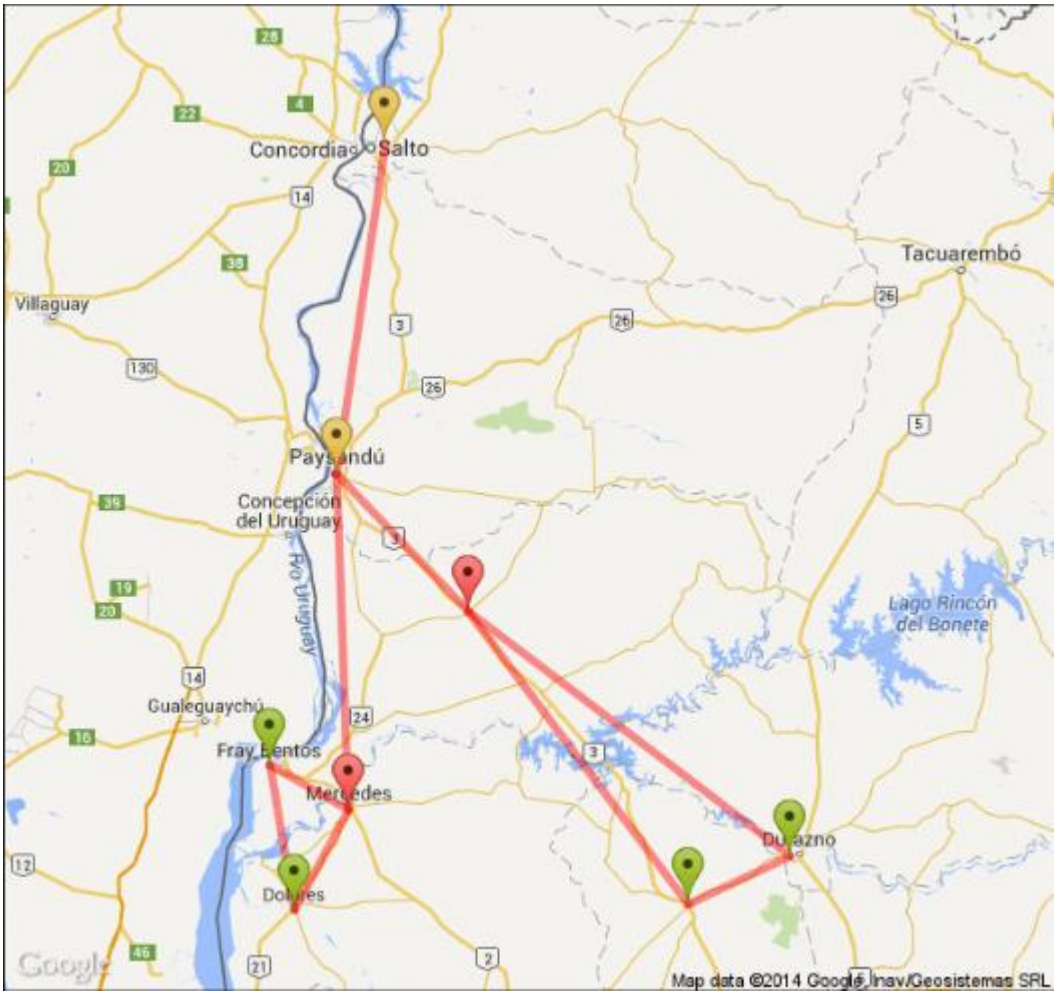
3. Reportes y Rutas

3.1. Mapa de estado

Firma: `Retorno mapaEstado();`

Descripción: Abre una ventana del navegador y muestra en un mapa de Google Maps todos los puntos registrados en el sistema: data-centers y ciudades. Se deberá utilizar la siguiente referencia de colores para diferenciarlos: ciudades en amarillo, y los data-centers en otros colores (uno para cada empresa).

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.



3.2. Solicitud de procesamiento (10 puntos)

Firma: `Retorno procesarInformación (Double coordX, Double coordY, int esfuerzoCPUrequeridoEnHoras);`

Descripción: Cuando llega un pedido de procesamiento de información a un DC, éste debe decidir si lo procesa él, o si debe enviarlo a otro DC, de manera tal de minimizar el costo (sumando el costo de transferencia entre los tramos al costo de procesamiento en el DC si es de otra empresa).

El método deberá retornar el nombre del DC que es elegido para procesar la información, el costo que tendrá, y deberá "ocupar" o marcar como utilizado de alguna manera el esfuerzo de CPU requerido en el DC de tal manera que no pueda ser utilizado por otro proceso.

El costo del proceso será calculado como la suma entre el costo de enviar la información hasta el DC destino, más el costo de procesamiento (0 si es de la misma empresa, o si es de otra empresa será *esfuerzoCPUrequeridoEnHoras* * *costoCPUporHora*).

Retornos posibles	
OK	• Si la solicitud pudo ser generada exitosamente.
ERROR	• 1. Si el punto de coordenadas *coordX, coordY* no existe en el sistema. • 2. Si no encuentra ningún DC que pueda satisfacer la necesidad de procesamiento.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

3.3. Listado de red mínima requerida (6 puntos)

Firma: `Retorno listadoRedMinima();`

Descripción: Se deberá retornar en el campo valorString un listado de los tramos (sin importar el orden) mínimos requeridos para mantener la conexión entre todos los DC y todas las ciudades con un costo total de conexión mínimo.

Además deberá retornar el costo total de conexión en el campo valorEntero.

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

Formato: Ejemplo con tres tramos que unen 4 puntos

valorString: DC1;Ciudad1|DC2;Ciudad1|DC3;DC2

valorEntero: 19

3.4. Listado de empresas

Firma: `Retorno listadoEmpresas();`

Descripción: Retorna la lista de todos las empresas ordenadas por nombre en manera ascendente. La lista deberá ser retornada en el campo valorString de tipo retorno siguiendo el formato de retorno establecido.

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

Formato: Ejemplo con dos empresas:

nombre1;email1|nombre2;email2

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.
- Deberá entregar dos copias en CD: conteniendo el código fuente y la documentación (en pdf con las fotos de los integrantes) que entregó impresa.
- Puntos acerca de la documentación impresa:
    - **Incluir las pre y post condiciones de los métodos solicitados**
    - **Incluir un diagrama de la estructura de datos que se implementó para representar el sistema junto con una breve explicación indicando por qué eligió dichas estructuras**
    - **Comentar las pruebas realizadas y los resultados obtenidos**
    - **La documentación tiene un valor de 7 puntos del total del obligatorio, que podrán restarse en caso de estar incompleta o mal presentada.**
    - **No incluir el código fuente en la documentación**
- 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 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