EVALUACION	**Obligatorio – v1.2**	**GRUPO**	**Todos**	**FECHA**	**09-12 / Set / 2019**
MATERIA	**Algoritmos y Estructuras de Datos 2**				
CARRERA	**Analista Programador – ATI**				
CONDICIONES	Puntos: Máximo: 40 Mínimo: 1 LA ENTREGA SE REALIZA EN FORMA ONLINE EN GESTIÓN, EN UN ARCHIVO NO MAYOR A 40MB EN FORMATO ZIP O RAR. **Fechas importantes:** Fecha lectura letra: semana del 9/9/2019 Fecha límite configuración repositorio github: 19/9/2019 Fecha máxima de entrega: 07/11/2019 Fecha de defensas: semana del 11/11/2019 **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” - Utilizar la plataforma github para el versionado del código. Deberán crear su repositorio en la primera semana luego de la lectura del obligatorio. Ver Anexo III.				

Obligatorio: Gestión de monopatines

Introducción

Una compañía de monopatines eléctricos necesita contar con ciertas funcionalidades en su aplicación para operar en la ciudad de Montevideo.

La app será utilizada por los usuarios que deseen hacer uso de los monopatines y también por los técnicos que recolectan los mismos en la noche.

Se busca que los usuarios puedan ubicar rápidamente qué monopatines tienen cerca visualizando su ubicación en el mapa.

Los monopatines tendrán un status, el cual siempre será uno de los siguientes:

Averiado, Descargado, Activo.

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

Pueden definirse tipos de datos (clases) auxiliares.

Aclaración: 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);`

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 una esquina o un monopatín.

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 usuarios

2.1. Registrar usuario

Firma: `Retorno registrarUsuario(String email, String nombre);`

Descripción: Registra el usuario con sus datos. El email identifica al usuario.

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 dirección de email no es una dirección válida. • 2. Si ya existe un afiliado con ese mail 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 el formato de email. Ver Anexo con links de interés.

2.2. Buscar usuario

Firma: `Retorno buscarUsuario(String email);`

Descripción: Retorna en valorString los datos del usuario con el formato “Email; Nombre”. Además, en el campo valorEntero de la clase Retorno, deberá retornar 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 usuario se encontró. • Retorna en valorString los datos del usuario. • Retorna en valorEntero la cantidad de elementos recorridos durante la búsqueda.
ERROR	• 1. Si el email no tiene un formato válido. • 2. Si no existe un usuario registrado con ese email en el sistema.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

Formato de retorno del valor String:

ana@mail.com;Ana

2.3. Listar todos los usuarios

Firma: `Retorno listarUsuarios();`

Descripción: Retorna en valorString los datos de todos los usuarios registrados, ordenados en forma alfabética ascendente por email.

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

Retornos posibles	
OK	• Si se pudo listar los usuarios 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:

ana@mail.com;Ana|omarejo@adinet.com.uy;Omar

3. Operaciones relativas a la red

3.1. Registrar monopatín

Firma: `Retorno registrarMonopatin(String chipId, double coordX, double coordY);`

Descripción: Registra el monopatín en las coordenadas *coordX, coordY* en el sistema y lo deja en estado Activo.

Retornos posibles	
OK	• Si el monopatín fue registrado exitosamente.
ERROR	• 1. Si en el sistema ya hay registrados el máximo de 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.

3.2. Registrar esquina

Firma: `Retorno registrarEsquina(double coordX, double coordY);`

Descripción: Registra la esquina de coordenadas *coordX, coordY* en el sistema.

Retornos posibles	
OK	• Si la esquina fue registrada exitosamente.
ERROR	• 1. Si en el sistema ya hay registrados el máximo de 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 metros);`

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

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.

Esta operación no tiene restricciones de eficiencia.

3.5. Monopatín más cercano

Firma: `Retorno monopatinMasCercano(double coordX, double coordY);`

Descripción: Calcula el camino desde el punto donde se encuentra el usuario (dado por las coordenadas) hasta el monopatín más cercano. Además de devolver el camino mínimo se deberá devolver el acumulado en metros de dicho camino.

Retornos posibles	
OK	• Si el camino pudo ser calculado exitosamente. • Retorna en valorEntero la cantidad de metros del camino.
ERROR	• 1. Si la esquina de coordenadas X e Y no existe. • 2. Si no se encuentra un camino desde el punto del usuario a un monopatín.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

Esta operación no tiene restricciones de eficiencia.

3.6. Zonas de recolección de monopatines

Firma: `Retorno monopatinesEnZona(double coordX, double coordY);`

Descripción: Al terminar el horario de uso de los monopatines, ciertos vehículos recolectores comienzan a recoger los monopatines. Cada vehículo tiene un punto de partida asignado desde el cual deberá asegurarse que todos los monopatines en su zona sean recogidos. La zona asignada a cada vehículo es de 1000 metros.

El operador del vehículo querrá obtener un listado de todos los monopatines que debe recoger y la ubicación de cada uno.

Nota: La distancia a considerar es la distancia transitable mediante los tramos presentes en el mapa, no es la distancia matemática.

Retornos posibles	
OK	• Si se pudo ejecutar 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:

coordx1;coordy1|coordx2;coordy2

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 todos los monopatines presentes en el mapa. Se desea distinguir con colores los monopatines según su estado: activos en verde, descargados en amarillo y averiados en rojo.

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.

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. Las pruebas se realizarán solamente utilizando la librería **junit**.
- 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). Respetar los nombres de los paquetes y clases (**no modificarla**).
- Se proporcionará un juego de pruebas junit de ejemplo, el cual **no deberá ser modificada** por el estudiante.
- 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.
- **Luego de la lectura de la letra, dispondrá de una semana para crear y configurar el repositorio github, agregando al docente como colaborador del proyecto.**
- Podrán encontrar en aulas guías para el uso de junit y de github.

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

Ver ejemplos en aulas

Parsear un string:

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

Anexo III: Uso de github

El uso de github como repositorio de código es de caracter OBLIGATORIO.
Términos:

- Cada estudiante deberá tener una cuenta (una existente o una nueva) en la plataforma github.com. Las cuentas son gratuitas.
- Cada grupo deberá tener su repositorio **privado** para el obligatorio (creado por uno de los estudiantes del grupo). Deberá agregar como colaboradores a su compañero de grupo y al docente.
- El nombre del repositorio deberá ser “AED2_OB_<nro_estudiante>_<nro_estudiante>”.
- El buen uso de la plataforma durante el obligatorio también será evaluado por el docente al momento de la corrección (grupal o también individual).

RECORDATORIO: IMPORTANTE PARA LA ENTREGA

Ø **Obligatorios** (Cap.IV.1, Doc. 220)

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 a destacar sobre la **entrega online de obligatorios** son:

1. La entrega se realizará desde **gestion.ort.edu.uy**
2. Previo a la conformación de grupos cada estudiante deberá estar inscripto a la evaluación. S**ugerimos realizarlo con anticipación.**
3. **Uno de los integrantes del grupo de obligatorio será el administrador del mismo** y es quien formará el equipo y subirá la entrega
4. Cada equipo (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)
5. El archivo a subir debe tener **un tamaño máximo de 40mb**
6. Les sugerimos **realicen una 'prueba de subida' al menos un día antes**, donde conformarán el **'grupo de obligatorio'.**
7. La **hora tope para subir el archivo será las 21:00** del día fijado para la entrega.
8. La entrega se podrá realizar desde cualquier lugar (ej. hogar del estudiante, laboratorios de la Universidad, etc)
9. Aquellos de ustedes que presenten alguna dificultad con su inscripción o tengan inconvenientes técnicos, por favor pasar por la oficina del Coordinador o por Coordinación adjunta **antes de las 20:00hs.** del día de la entrega

Si tuvieras una situación particular de fuerza mayor, debes dirigirte con suficiente antelación al plazo de entrega, al Coordinador de Cursos o Secretario Docente.