EVALUACION	Obligatorio	GRUPO	TODOS	FECHA	15/05/2019
MATERIA	**Algoritmos y Estructuras de Datos 1**				
CARRERA	Analista Programador / Analista en Tecnologías de la información				
CONDICIONES	**- Puntos:** Máximo: 45 Mínimo: 0 **- Fecha máxima de entrega**: 20/06/2019 **IMPORTANTE** - Los grupos deben estar conformados por dos personas. - Inscribirse (sacar la "boleta de entrega"). - La entrega será únicamente por gestión. - Se debe entregar un conjunto de pruebas en JUnit. Estas deben cubrir al menos todos los métodos, tanto los casos con resultado OK, como los con resultado ERROR.				

Obligatorio: Red Social

Contenido

1. Introducción

Se desea implementar un simulador de un editor de textos para una app estilo WhatsApp que permita el manejo de mensajes, líneas, palabras y su vinculación a un diccionario ortográfico de palabras. El sistema actuará a nivel de memoria no persistente y no contempla el desarrollo de interfaces gráficas.

El texto del mensaje puede estar compuesto por más de una línea no habiendo limite en cuanto a la cantidad de líneas a manejar.

Cada línea está compuesta por palabras y en este caso si tenemos un límite máximo de palabras, (MAX_CANT_PALABRAS_X_LINEA), que estará definido por el sistema.

Algunas consideraciones:

Las palabras no deben tener espacios en blanco.
En una línea, las palabras existentes deben ocupar posiciones consecutivas, desde la posición 1 en adelante (no hay huecos posibles entre palabras).
Al iniciar el sistema el texto será vacío (0 líneas) y el diccionario ortográfico no contendrá palabras.
Definimos primero las operaciones relativas al Sistema crear sistema y destruir sistema, luego operaciones sobre texto (líneas), luego las operaciones sobre una línea y finalmente las funcionalidades relativas al diccionario ortográfico de palabras. En cada caso describimos una funcionalidad del sistema, desarrollamos un ejemplo de la operación partiendo de un texto vacío, e indicamos los posibles retornos.

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

Sistema	public class Sistema{
	/*Aquí introduzca la información que estime conveniente*/
	}

Retorno	public class Retorno{
	enum TipoRet{OK,ERROR, NO_IMPLEMENTADA};
	/*Aquí introduzca la información que estime conveniente*/
	
	boolean valorBooleano
	int valorEntero;
	String valorString;
	Resultado resultado;
	
	}

Pueden definirse tipos de datos (clases) auxiliares.

2. Funcionalidades

2.1. Crear Sistema de Mensajes

Firma: `Retorno crearSistemaMensajes();`

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

Retornos posibles	
OK	Siempre retorna OK.
ERROR	No hay errores
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.

2.2. Destruir Sistema de Mensajes

Firma: `Retorno destruirSistemaMensajes();`

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.

OPERACIONES RELATIVAS A LAS LÍNEAS (MENSAJE):

2.3. Inserta una nueva línea vacía al final del texto.

Retorno InsertarLinea();

Retornos posibles	
OK	Esta operación siempre debe insertar una nueva línea.
ERROR	No hay errores (lógicos) posibles.
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

Ejemplo:

InsertarLinea();
InsertarLinea();
ImprimirTexto();
Salida:

1:
2:

Nota: ImprimirTexto() se desarrolla más adelante como operación del sistema

2.4. Inserta una nueva línea vacía en la posición indicada.

Retorno InsertarLineaEnPosicion(Posicion posicionLinea);

Inserta una línea vacía en la posición indicada y mueve todas las líneas que se encuentran a partir de la posición indicada, una posición más adelante.
La posición es válida solamente si (posicionLinea >= 1) y (posicionLinea <= cantidad de líneas + 1)

Retornos posibles	
OK	Si se pudo insertar la línea con éxito.
ERROR	Si la posición no es valida
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

Ejemplo:

```
InsertarLineaEnPosicion(1);
InsertarPalabra(1, 1, "Palabra1");
ImprimirTexto();
```

Salida:

```
1: Palabra1
InsertarLineaEnPosicion(1);
ImprimirTexto();
```

Salida:

```
1:
2: Palabra1
```

Nota: InsertarPalabra se desarrolla como operación 8 del sistema

2.5. Borra la línea en la posición indicada.

Retorno BorrarLinea(Posicion posicionLinea);

Borra la línea en la posición indicada y mueve todas las líneas que se encuentran a partir de la posición indicada, una posición más hacia arriba.
La posición es válida solamente si posicionLinea existe en el texto, esto es, si posicionLinea >= 1 y posicionLinea <= cantidad de líneas.

Retornos posibles	
OK	Si se pudo borrar la línea con éxito.
ERROR	Si la posición no es valida
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

Ejemplo:

```
InsertarLinea();
InsertarLinea();
InsertarPalabra(2, 1, "Palabra1");
ImprimirTexto();
```

Salida:

```
1:
2: Palabra1
BorrarLinea(1);
ImprimirTexto();
```

Salida:

```
1: Palabra1
```

2.6. Borra todas las líneas del texto.

Retorno BorrarTodo();
Borra todas las líneas del texto.

Retornos posibles	
OK	Si se pudo borrar el texto con éxito.
ERROR	No hay errores posibles
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

Ejemplo:

```
InsertarLinea();
InsertarLinea();
InsertarPalabra(2, 1, "Palabra1");
ImprimirTexto();
```

Salida:

```
1:
2: Palabra1
BorrarTodo();
ImprimirTexto();
```

Salida:

```
Texto vacio
```

2.7. Borra todas las ocurrencias de una palabra en el texto.

Retorno BorrarOcurrenciasPalabraEnTexto(Cadena palabraABorrar);

Borra todas las ocurrencias en el texto de la palabra indicada, desplazando hacia adelante en cada línea las palabras que eventualmente se encuentren en posiciones posteriores a la eliminada.

Retornos posibles	
OK	Si se pudieron borrar las ocurrencias de la palabra con éxito.
ERROR	Si la palabra a borrar no existe
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

Ejemplo:

```
InsertarLinea();
InsertarLinea();
InsertarPalabra(1, 1, "Palabra1");
InsertarPalabra(1, 2, "Palabra2");
InsertarPalabra(2, 1, "Palabra1");
InsertarPalabra(2, 1, "Palabra2");
InsertarLineaEnPosicion(2);
InsertarPalabra(2, 1, "Palabra2");
ImprimirTexto();
```

Salida:

```
1: Palabra1 Palabra2
2: Palabra2
3: Palabra2 Palabra1
BorrarOcurrenciasPalabraEnTexto("Palabra2");
ImprimirTexto();
```

Salida:

```
1: Palabra1
2:
3: Palabra1
```

2.8. Imprime el texto por pantalla.

Retorno ImprimirTexto();

Muestra todas las líneas del texto con sus respectivas palabras. Se debe imprimir el número de línea, para cada línea, según muestra el ejemplo. Cuando el texto no tiene líneas se debe mostrar el mensaje "Texto vacio".

Retornos posibles	
OK	Se muestra texto con éxito.
ERROR	
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

Ejemplo:

```
InsertarLinea();
InsertarPalabra(1, 1, "Palabra1");
InsertarLinea();
InsertarPalabra(2, 1, "Palabra2");
InsertarLinea();
ImprimirTexto();
```

Salida:

```
1: Palabra1
2: Palabra2
3:
BorrarTodo();
ImprimirTexto();
```

Salida:

```
Texto vacio
```

OPERACIONES RELATIVAS A LAS PALABRAS:

2.9. Inserta una palabra en una línea sólo si la línea no está llena.

Retorno InsertarPalabraEnLinea(Posicion posicionLinea, Posicion posicionPalabra, Cadena palabraAIngresar);

Inserta la palabra palabraAIngresar en la línea posicionLinea y dentro de la línea en la posición posicionPalabra. Si al ingresar la palabra se superara la cantidad máxima de palabras por línea (MAX_CANT_PALABRAS_X_LINEA), la inserción no ser haría y devolvería un mensaje de error.
La posición posicionLinea es válida solamente si existe en el texto.
La posición posicionPalabra es válida solamente si (posicionPalabra >= 1) y (posicionPalabra <=cantidad de palabras en la línea + 1).

Retornos posibles	
OK	Se pudo insertar palabra con éxito.
ERROR	Si la posición de la línea no es válida. Si la posición de la palabra no es válida. Si se superara la cantidad máxima de palabras por línea.
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

Ejemplo (suponiendo MAX_CANT_PALABRAS_X_LINEA igual a 3):

```
InsertarLinea();
InsertarLinea();
InsertarPalabra(2, 1, "Palabra1");
InsertarPalabra(2, 2, "Palabra2");
ImprimirTexto();
```

Salida:

```
1:
2: Palabra1 Palabra2
InsertarPalabra(2, 1, "Palabra3");
ImprimirTexto();
```

Salida:

```
1:
2: Palabra3 Palabra1 Palabra2
InsertarPalabra(2, 2, "Palabra4");
ImprimirTexto();
```

Salida:

```
1:
2: Palabra3 Palabra1 Palabra2
```

2.10. Inserta una palabra en una línea y desplaza a la siguiente si es necesario.

Retorno InsertarPalabraYDesplazar(Posicion posicionLinea, Posicion posicionPalabra, Cadena palabraAIngresar);

Inserta la palabra palabraAIngresar en la línea posicionLinea y dentro de la línea en la posición posicionPalabra. Desplaza todas las palabras que se encuentran a partir de la posición posicionPalabra un lugar hacia adelante. Si al ingresar la palabra se superara la cantidad máxima de palabras por línea (MAX_CANT_PALABRAS_X_LINEA), el desplazamiento afectaría a la línea siguiente y eventualmente a las posteriores, llegando incluso a ser necesario en el caso extremo agregar una nueva línea al final del documento con una sola palabra (si todas las líneas posteriores estuvieran llenas).
La posición posicionLinea es válida solamente si existe en el texto.
La posición posicionPalabra es válida solamente si (posicionPalabra >= 1) y (posicionPalabra <=cantidad de palabras en la línea + 1).

Retornos posibles	
OK	Se pudo insertar palabra con éxito.
ERROR	Si la posición de la línea no es válida. Si la posición de la palabra no es válida.
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

Ejemplo (suponiendo MAX_CANT_PALABRAS_X_LINEA igual a 3):

```
InsertarLinea();
InsertarLinea();
InsertarPalabra(2, 1, "Palabra1");
InsertarPalabra(2, 2, "Palabra2");
ImprimirTexto();
```

Salida:

```
1:
2: Palabra1 Palabra2
```

```
InsertarPalabra(2, 1, "Palabra3");
ImprimirTexto();
```

Salida:

```
1:
2: Palabra3 Palabra1 Palabra2
```

```
InsertarPalabraYDesplazar(2, 2, "Palabra4");
ImprimirTexto();
```

Salida:

```
1:
2: Palabra3 Palabra4 Palabra1
3: Palabra2
```

```
InsertarPalabraYDesplazar (2, 3, "Palabra5");
ImprimirTexto();
```

Salida:

```
1:
2: Palabra3 Palabra4 Palabra5
3: Palabra1 Palabra2
```

2.11. Borra la palabra en la posición indicada.

Retorno BorrarPalabra(Posicion posicionLinea, Posicion posicionPalabra);

Borra la palabra en la posición posicionPalabra de la línea posicionLinea y mueve todas las palabras (si las hay) en las posiciones posteriores de dicha línea un lugar hacia adelante.La posicionLinea es válida solamente si posicionLinea existe en el texto.
La posicionPalabra es válida solamente si posicionPalabra existe en la línea.

Retornos posibles	
OK	Se pudo insertar palabra con éxito.
ERROR	Si la posición de la línea no es válida. Si la posición de la palabra no es válida.
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

Ejemplo:

```
InsertarLinea();
InsertarLinea();
InsertarPalabra(2, 1, "Palabra1");
InsertarPalabra(2, 2, "Palabra2");
InsertarPalabra(2, 1, "Palabra3");
ImprimirTexto();
```

Salida:

```
1:
2: Palabra3 Palabra1 Palabra2
```

```
BorrarPalabra(2, 1);
ImprimirTexto();
```

Salida:

```
1:
2: Palabra1 Palabra2
```

```
InsertarLinea();
InsertarPalabra(3, 1, "Palabra3");
BorrarPalabra(2, 1);
BorrarPalabra(2, 1);
ImprimirTexto();
```

Salida:

```
1:
2:
3: Palabra3
```

2.12. Borra todas las ocurrencias de una palabra en la línea indicada.

Retorno BorrarOcurrenciasPalabraEnLinea(Posicion posicionLinea,
Cadena palabraABorrar);

Borra todas las ocurrencias de la palabra palabraABorrar en la línea indicada. Cada vez que borra una palabra mueve todas las palabras de la línea que se encuentran a partir de la posición borrada una posición hacia delante.

Retornos posibles	
OK	Si se pudieron borrar las ocurrencias de la palabra con éxito.
ERROR	Si la posición de la línea no es válida
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

La posicionLinea es válida solamente si posicionLinea existe en el texto.

Ejemplo:

```
InsertarLinea();
InsertarLinea();
InsertarPalabra(2, 1, "Palabra1");
InsertarPalabra(2, 2, "Palabra2");
InsertarPalabra(2, 1, "Palabra2");
ImprimirTexto();
```

Salida:

```
1:
2: Palabra2 Palabra1 Palabra2
BorrarOcurrenciasPalabraEnLinea(2, "Palabra2");
ImprimirTexto();
```

Salida:

```
1:
2: Palabra1
```

2.13. Imprime la línea por pantalla.

Retorno ImprimirLinea(Posicion posicionLinea);

Imprime la línea con todas sus palabras. Se debe imprimir el número de línea según muestra el ejemplo.

La posicionLinea es válida solamente si posicionLinea existe en el texto.

Retornos posibles	
OK	Si se mostró la línea.
ERROR	Si la posición de la línea no es válida
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

Ejemplo:

```
InsertarLinea();
InsertarLinea();
ImprimirLinea(2);
Salida:
2:

InsertarPalabra(2, 1, "Palabra1");
InsertarPalabra(2, 2, "Palabra2");
ImprimirLinea(2);
Salida:
2: Palabra1 Palabra2
```

OPERACIONES RELATIVAS AL DICCIONARIO:

2.14. Agrega una palabra al diccionario.

Retorno IngresarPalabraDiccionario(Cadena palabraAIngresar);
Ingresa una palabra al diccionario si ésta no se encuentra en el mismo.

Retornos posibles	
OK	Si se ingresó correctamente.
ERROR	Si la palabra ya existe
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

Ejemplo:
IngresarPalabraDiccionario("Hoja");
IngresarPalabraDiccionario("Hojalata");
IngresarPalabraDiccionario("Bosque");
ImprimirDiccionario();
Salida:
Bosque
Hoja
Hojalata

Nota: ImprimirDiccionario se desarrolla como operación 14 del sistema.

2.15. Borra una palabra del diccionario.

Retorno BorrarPalabraDiccionario(Cadena palabraABorrar);
Borra una palabra del diccionario si se encuentra en el mismo.

Retornos posibles	
OK	Si se borró correctamente.
ERROR	Si la palabra no existe
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

Ejemplo:
IngresarPalabraDiccionario("Hoja");
IngresarPalabraDiccionario("Hojalata");
IngresarPalabraDiccionario("Bosque");
BorrarPalabraDiccionario("Hoja");
ImprimirDiccionario();
Salida:
Bosque
Hojalata

2.16. Muestra las palabras del diccionario alfabéticamente.

Retorno ImprimirDiccionario();

Muestra las palabras del diccionario ordenadas alfabéticamente, de menor a mayor. Debe imprimirse según muestra el ejemplo. Cuando el diccionario no tiene palabras debe mostrarse el mensaje "Diccionario vacio".

Retornos posibles	
OK	Se mostro diccionario correctamente
ERROR	No hay errores
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

Ejemplo:

```
ImprimirDiccionario();
```

Salida:

```
Diccionario vacio
IngresarPalabraDiccionario("Hoja");
IngresarPalabraDiccionario("Hojalata");
IngresarPalabraDiccionario("Bosque");
ImprimirDiccionario();
```

Salida:

```
Bosque
Hoja
Hojalata
```

2.17. Muestra las palabras del texto que no se encuentran en el diccionario.

Retorno ImprimirTextoIncorrecto();

Muestra todas las líneas del texto, pero exhibiendo solamente las palabras que no se encuentran en el diccionario. Se debe imprimir el número de línea según muestra el ejemplo. Cuando el texto no tiene líneas se debe mostrar el mensaje "Texto vacio".

Retornos posibles	
OK	Se mostro texto
ERROR	No hay errores
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

Ejemplo:

```
InsertarLinea();
InsertarLinea();
InsertarLinea();
InsertarPalabra(2, 1, "Palabra21");
InsertarPalabra(2, 2, "Palabra22");
InsertarPalabra(1, 1, "Palabra11");
InsertarPalabra(1, 2, "Palabra12");
InsertarPalabra(1, 3, "Palabra13");
InsertarPalabra(3, 1, "Palabra31");
ImprimirTexto();
```

Salida:

```
1: Palabra11 Palabra12 Palabra13
2: Palabra21 Palabra22
3: Palabra31
```

```
IngresarPalabraDiccionario("Palabra12");
IngresarPalabraDiccionario("Palabra21");
IngresarPalabraDiccionario("Palabra22");
ImprimirTextoIncorrecto();
```

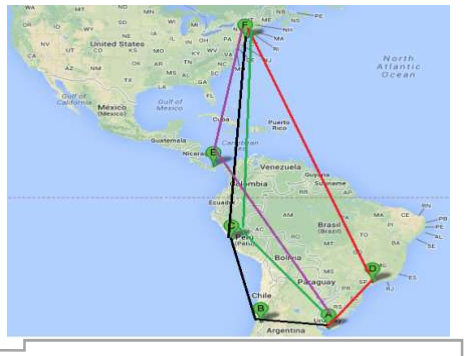
Salida:

```
1: Palabra11 Palabra13
2:
3: Palabra31
```

3. 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:

3.1. Cargar Matriz de Distancias

Firma: `TipoRet 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.

3.2. Camino más corto

Firma: `TipoRet 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.

4. 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: 0/45
- Los obligatorios se realizan por equipos de **2 estudiantes.**
- Plazo máximo de segunda entrega (con boleta): **20/06/2019 21:00.**
- 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).