ESTRUCTURA DE DATOS Y ALGORITMOS OBLIGATORIO

Ignacio Cabrera
(236296)

Santiago Manzoni
(230311)

Grupo: M3B
Docente: Francisco Bouza
Junio de 2019

Tabla de contenidos:

1. Interfaz del Sistema: Pre y Post condiciones

Nota:
****Para todos los métodos que no son crearSistemaMensajes()***

***//Pre*:** *-El objeto Sistema debe estar inicializado(crearSistemaMensajes()).*

_____

Retorno crearSistemaMensajes();

//Pre: *-*
//Post: *Incializa la variable texto de tipo Lista<ILista<String>> que posee el objeto Sistema, también incializa la variable diccionario de tipo ListaOrd<String>.*

_____

Retorno destruirSistemaMensajes();

//Pre: *-*
//Post: *En caso de tener información almacenada elimina la misma, luego procede a 'nullear' las referencias de los componentes de Sistema.*

_____

Retorno InsertarLinea();

//Pre: *-*
//Post: *Inserta una linea al final del texto (si el texto no posee lineas inserta la primera).*

_____

Retorno InsertarLineaEnPosicion(**int** posicionLinea);

//Pre: *-posicionLinea > 0 && posicionLinea < _cant (propiedad interna de Lista, cantidad de lineas en texto).*
//Post: *Inserta una linea en la posición del texto indicada por parámetro.*

_____

Retorno BorrarLinea(**int** posicionLinea);

//Pre: *-posicionLinea > 0 && posicionLinea < _cant .*
//Post: *Borra la linea de la posición indicada y mueve todas las líneas siguientes una posición hacia atrás.*

_____

Retorno BorrarTodo();

//Pre: *El texto debe tener al menos una linea y una palabra ingresada.*
//Post: *Elimina linea por linea todas las palabras ingresadas, luego elimina las lineas (deja el texto vacío).*

_____

Retorno BorrarOcurrenciasPalabraEnTexto(**String** palabraABorrar);

//Pre: *-El texto debe tener palabras ingresadas.*
-La palabra debe existir entre las ingresadas.
//Post: *Borra todas las ocurrencias en el texto de la palabra indicada, desplazando hacia atras en cada línea las otras palabras para no generar espacios vacios en medio.*

Retorno ImprimirTexto();

//Pre: *- El texto debe tener palabras ingresadas. (Sino imprime "Texto vacío").*
//Post: *Imprime en pantalla todas las palabras del texto línea por línea con un espacio en blanco entre ellas.*

Retorno InsertarPalabraEnLinea(**int** *posicionLinea*, **int** *posicionPalabra*, **String** *palabraAIngresar*);

//Pre: *-La posición posicionLinea debe ser > 0 y <= a la cantidad total de lineas del texto.*
-La posición posicionPalabra debe ser > 0 y <= cantidad de palabras en la línea.
//Post: *Inserta la palabra en la línea posicionLinea, y en la posición posicionPalabra de dicha línea.*

Retorno InsertarPalabraYDesplazar(**int** *posicionLinea*, **int** *posicionPalabra*, **String** *palabraAIngresar*);

//Pre: *-La posición posicionLinea debe ser > 0 y <= a la cantidad total de líneas del texto.*
-La posición posicionPalabra debe ser > 0 y <= cantidad de palabras en la línea.
//Post: *Inserta la palabra en la línea posicionLinea, y en la posición posicionPalabra de dicha línea, en caso que la línea estuviese llena, inserta igual y corre las demás hacia adelante.*

Retorno BorrarPalabra(**int** *posicionLinea*, **int** *posicionPalabra*);

//Pre: *-El texto debe tener líneas insertadas.*
-La posicionLinea debe ser >0 y < _cant.
-La línea indicada por parámetro debe contener palabras ingresadas.
-La posicionPalabra debe ser > 0 y <= a la cantidad de palabras que tiene la línea donde se encuentra (debe ser una posición válida en la línea).
//Post:*Se borra la palabra ubicada en la línea y posición indicada.*

Retorno ImprimirLinea(**int** *posicionLinea*);

//Pre: *-Debe existir al menos una línea ingresada al texto.*
-posicionLinea > 0 && posicionLinea <= cantidad de palabras de la línea.
//Post: *Imprime todas las palabras contenidas en la línea de la posición posicionLinea.*

Retorno IngresarPalabraDiccionario(**String** *palabraAIngresar*);

//Pre: *-*
//Post: *Ingresa la palabra palabraAIngresar de forma ordenada a la ListaOrd diccionario.*

_____

Retorno BorrarPalabraDiccionario(**String** *palabraABorrar*);

//Pre: *palabraABorrar debe existir contenida en diccionario.*
//Post: *Elimina la palabraABorrar del diccionario.*

_____

Retorno ImprimirDiccionario();

//Pre: *-La ListaOrd diccionario debe contener al menos una palabra ingresada.*
//Post: *Imprime todas las palabras que estén dentro de la ListaOrd diccionario.*

_____

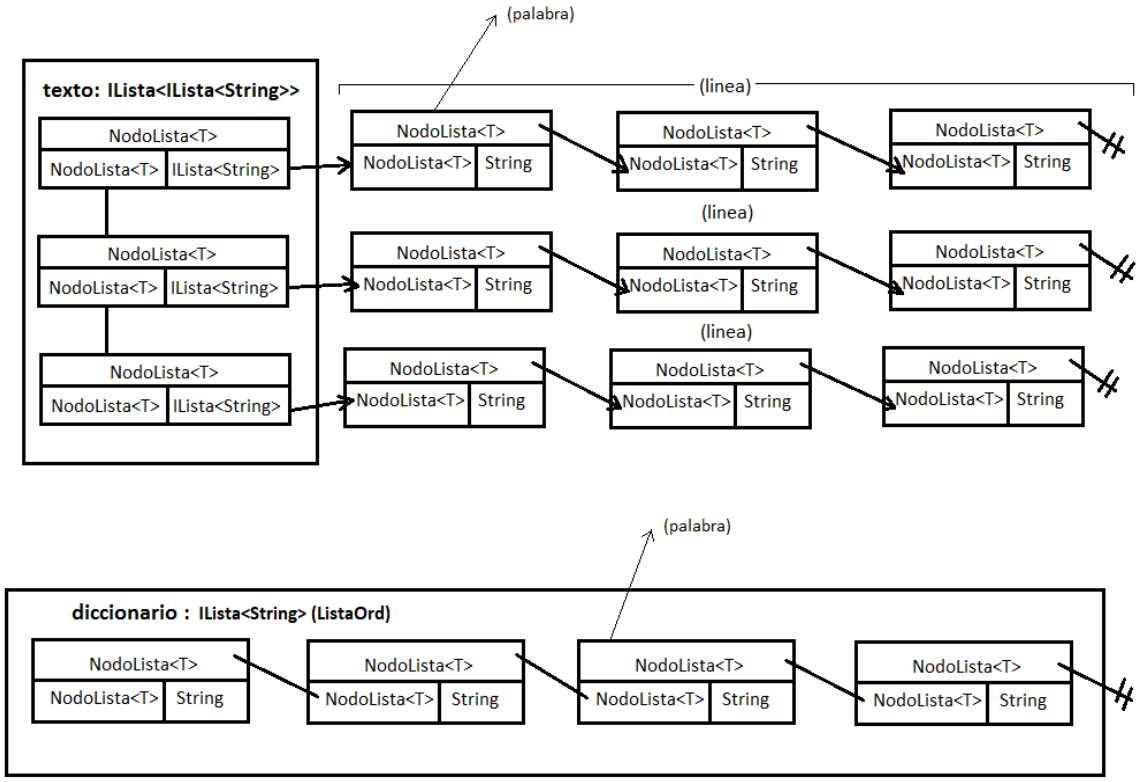
Retorno ImprimirTextoIncorrecto();

//Pre: *-texto debe contener al menos una línea y una palabra ingresada.*
- diccionario debe contener al menos una palabra ingresada.
//Post: *Imprime todas las palabras contenidas en texto que no estén en diccionario.*

_____

2. Solución escogida

2.1 Diagrama de la estructura de datos



2.2 Justificación

Nombre Entidad: texto	
Estructura	ILista<IListaString>> (Lista)
Justificación	La estructura mencionada nos permite movernos entre los NodoLista con facilidad ya que la clase Lista tiene definidos varios métodos de utilidad para trabajar con la misma, a diferencia de lo que podría haber sido implementar el ejercicio con nodos sueltos que sería mucho más engorroso.

Nombre Entidad: diccionario	
Estructura	IListaString> (ListaOrd)
Justificación	El implementar IListaString> como ListaOrd nos da el beneficio de que ya la misma se ordena alfabéticamente (aparte de implementar todos los métodos de Lista genéricos) por automatismo, facilitándonos las tareas de inserción y eliminación.

3. Pruebas Ejecutadas

```
@Test
public void testCrearSistemaMensajes() {

        Retorno ret = sis.crearSistemaMensajes();
        assertEquals(Resultado.OK, ret.resultado);
}


@Test
public void testDestruirSistemaMensajes() {
        Retorno ret = sis.crearSistemaMensajes();
        ret = sis.destruirSistemaMensajes();
        assertEquals(Resultado.OK, ret.resultado);
}


@Test
public void testInsertarLinea() {
        Retorno ret = sis.crearSistemaMensajes();
        ret = sis.InsertarLinea();
        assertEquals(Resultado.OK, ret.resultado);
}


@Test
public void testInsertarLineaEnPosicion() {
        Retorno ret = sis.crearSistemaMensajes();
        ret = sis.InsertarLineaEnPosicion(1);
        assertEquals(Resultado.OK, ret.resultado);
}


@Test
public void testInsertarLineaEnPosicionFueraDeRango() {
        Retorno ret = sis.crearSistemaMensajes();
        ret = sis.InsertarLineaEnPosicion(486);
        assertEquals(Resultado.ERROR_1, ret.resultado);
}


@Test
public void testInsertarLineaEnPosicionIgualA0() {
        Retorno ret = sis.crearSistemaMensajes();
        ret = sis.InsertarLineaEnPosicion(0);
        assertEquals(Resultado.ERROR_1, ret.resultado);
}


@Test
public void testBorrarLinea() {
        Retorno ret = sis.crearSistemaMensajes();
        ret = sis.InsertarLinea();
        ret = sis.BorrarLinea(1);
        assertEquals(Resultado.OK, ret.resultado);
}
```

```
@Test
public void testBorrarLineaIgualA0() {
	Retorno ret = sis.crearSistemaMensajes();
	ret = sis.InsertarLinea();
	ret = sis.BorrarLinea(0);
	assertEquals(Resultado.ERROR_1, ret.resultado);
}


@Test
public void testBorrarLineaMayorRango() {
	Retorno ret = sis.crearSistemaMensajes();
	ret = sis.InsertarLinea();
	ret = sis.BorrarLinea(6541);
	assertEquals(Resultado.ERROR_1, ret.resultado);
}


@Test
public void testBorrarTodo() {
	Retorno ret = sis.crearSistemaMensajes();
	ret = sis.InsertarLinea();
	ret = sis.BorrarTodo();
	assertEquals(Resultado.OK, ret.resultado);
}


@Test
public void testBorrarOcurrenciasPalabraEnTexto() {
	Retorno ret = sis.crearSistemaMensajes();
	ret = sis.InsertarLinea();
	ret = sis.InsertarLinea();
	ret = sis.InsertarPalabraEnLinea(1, 1, "PEPE");
	ret = sis.InsertarPalabraEnLinea(2, 1, "CACA");
	ret = sis.InsertarPalabraEnLinea(2, 2, "Sape");
	ret = sis.BorrarOcurrenciasPalabraEnTexto("CACA");

	assertEquals(Resultado.OK, ret.resultado);
}


@Test
public void testBorrarOcurrenciasPalabraEnTextoSinOcurrencia() {
	Retorno ret = sis.crearSistemaMensajes();
	ret = sis.InsertarLinea();
	ret = sis.InsertarLinea();
	ret = sis.InsertarPalabraEnLinea(1, 1, "PEPE");
	ret = sis.InsertarPalabraEnLinea(2, 1, "SOSO");
	ret = sis.InsertarPalabraEnLinea(2, 2, "Sape");
	ret = sis.BorrarOcurrenciasPalabraEnTexto("ASASDASDSAD");

	assertEquals(Resultado.ERROR_1, ret.resultado);
}
```

```
@Test
public void testImprimirTexto() {
	Retorno ret = sis.crearSistemaMensajes();
	ret = sis.InsertarLinea();
	ret = sis.InsertarPalabraEnLinea(1, 1, "Palabra1");
	ret = sis.InsertarLinea();
	ret = sis.InsertarPalabraEnLinea(2, 1, "Palabra2");
	ret = sis.InsertarLinea();
	ret = sis.ImprimirTexto();

	assertEquals(Resultado.OK, ret.resultado);
}


@Test
public void testImprimirTextoVacio() {
	Retorno ret = sis.crearSistemaMensajes();
	ret = sis.InsertarLinea();
	ret = sis.InsertarPalabraEnLinea(1, 1, "Palabra1");
	ret = sis.InsertarLinea();
	ret = sis.InsertarPalabraEnLinea(2, 1, "Palabra2");
	ret = sis.InsertarLinea();
	ret = sis.BorrarTodo();
	ret = sis.ImprimirTexto();

	assertEquals(Resultado.OK, ret.resultado);
}


@Test
public void testInsertarPalabraEnLinea() {
	Retorno ret = sis.crearSistemaMensajes();
	ret = sis.InsertarLinea();
	ret = sis.InsertarPalabraEnLinea(1, 1, "Palabra1");
	ret = sis.InsertarLinea();
	ret = sis.InsertarPalabraEnLinea(1, 2, "Palabra2");

	assertEquals(Resultado.OK, ret.resultado);
}


@Test
public void testInsertarPalabraEnLineaFueraDeRango() {
	Retorno ret = sis.crearSistemaMensajes();
	ret = sis.InsertarLinea();
	ret = sis.InsertarPalabraEnLinea(1, 1, "Palabra1");
	ret = sis.InsertarLinea();
	ret = sis.InsertarPalabraEnLinea(5, 3, "Palabra2");

	assertEquals(Resultado.ERROR_1, ret.resultado);
}
```

```
@Test
public void testInsertarPalabraEnLineaSinPalabraEnPosMayorA1() {
        Retorno ret = sis.crearSistemaMensajes();
        ret = sis.InsertarLinea();
        ret = sis.InsertarPalabraEnLinea(1, 1, "Palabra1");
        ret = sis.InsertarLinea();
        ret = sis.InsertarPalabraEnLinea(1, 3, "Palabra2");

        assertEquals(Resultado.ERROR_2, ret.resultado);
}


@Test
public void testInsertarPalabraEnLineaPalabraFueraDeRango() {
        Retorno ret = sis.crearSistemaMensajes();
        ret = sis.InsertarLinea();
        ret = sis.InsertarPalabraEnLinea(1, 1, "Palabra1");
        ret = sis.InsertarLinea();
        ret = sis.InsertarPalabraEnLinea(1, 2, "Palabra2");
        ret = sis.InsertarPalabraEnLinea(1, 3, "Palabra2");

        assertEquals(Resultado.ERROR_3, ret.resultado);
}


@Test
public void testInsertarPalabraYDesplazar() {
        sis = new Sistema(3);
        Retorno ret = sis.crearSistemaMensajes();
        ret = sis.InsertarLinea();
        ret = sis.InsertarPalabraYDesplazar(1, 1, "Palabra1");
        ret = sis.InsertarPalabraYDesplazar(1, 2, "Palabra3");
        ret = sis.InsertarPalabraYDesplazar(1, 2, "Palabra2");

        assertEquals(Resultado.OK, ret.resultado);
}


@Test
public void testInsertarPalabraYDesplazarLineaFueraDeRango() {
        Retorno ret = sis.crearSistemaMensajes();
        ret = sis.InsertarLinea();
        ret = sis.InsertarPalabraYDesplazar(15, 1, "Palabra1");

        assertEquals(Resultado.ERROR_1, ret.resultado);
}


@Test
public void testInsertarPalabraYDesplazarPalabraFueraDeRango() {
        Retorno ret = sis.crearSistemaMensajes();
        ret = sis.InsertarLinea();
        ret = sis.InsertarPalabraYDesplazar(1, 56, "Palabra1");

        assertEquals(Resultado.ERROR_2, ret.resultado);
}
```

```
@Test
public void testBorrarPalabra() {
Retorno ret = sis.crearSistemaMensajes(), aux_ret = sis.crearSistemaMensajes();
        ret = sis.InsertarLinea();
        ret = sis.InsertarLinea();
        ret = sis.InsertarLinea();
        ret = sis.InsertarPalabraEnLinea(1, 1, "Palabra1");
        ret = sis.InsertarPalabraEnLinea(1, 2, "Palabra2");
        ret = sis.InsertarPalabraEnLinea(2, 1, "Palabra3");
        ret = sis.InsertarPalabraEnLinea(3, 1, "Palabra4");
        ret = sis.InsertarPalabraEnLinea(3, 2, "Palabra5");
        ret = sis.BorrarPalabra(2, 1);
        aux_ret = sis.ImprimirTexto();

        assertEquals(Resultado.OK, ret.resultado);
}


@Test
public void testBorrarPalabraLineaFueraDeRango() {
        Retorno ret = sis.crearSistemaMensajes();
        ret = sis.InsertarLinea();
        ret = sis.InsertarPalabraYDesplazar(1, 1, "Palabra1");
        ret = sis.InsertarPalabraYDesplazar(1, 2, "Palabra3");
        ret = sis.InsertarPalabraYDesplazar(1, 2, "Palabra2");
        ret = sis.BorrarPalabra(0, 1);

        assertEquals(Resultado.ERROR_1, ret.resultado);
}


@Test
public void testBorrarPalabraFueraDeRango() {
        Retorno ret = sis.crearSistemaMensajes();
        ret = sis.InsertarLinea();
        ret = sis.InsertarPalabraYDesplazar(1, 1, "Palabra1");
        ret = sis.InsertarPalabraYDesplazar(1, 2, "Palabra3");
        ret = sis.InsertarPalabraYDesplazar(1, 2, "Palabra2");
        ret = sis.BorrarPalabra(1, 3);

        assertEquals(Resultado.ERROR_2, ret.resultado);
}


@Test
public void testImprimirLinea() {
        Retorno ret = sis.crearSistemaMensajes();
        ret = sis.InsertarLinea();
        ret = sis.InsertarPalabraYDesplazar(1, 1, "Palabra1");
        ret = sis.InsertarPalabraYDesplazar(1, 2, "Palabra3");
        ret = sis.InsertarPalabraYDesplazar(1, 2, "Palabra2");
        ret = sis.ImprimirLinea(1);

        assertEquals(Resultado.OK, ret.resultado);
}
```

```
@Test
public void testImprimirLineaFueraDeRango() {
        Retorno ret = sis.crearSistemaMensajes();
        ret = sis.InsertarLinea();
        ret = sis.InsertarPalabraYDesplazar(1, 1, "Palabra1");
        ret = sis.InsertarPalabraYDesplazar(1, 2, "Palabra3");
        ret = sis.InsertarPalabraYDesplazar(1, 2, "Palabra2");
        ret = sis.ImprimirLinea(7);

        assertEquals(Resultado.ERROR_1, ret.resultado);
}


@Test
public void testIngresarPalabraDiccionario() {
        Retorno ret = sis.crearSistemaMensajes();
        ret = sis.IngresarPalabraDiccionario("Palabra1");

        assertEquals(Resultado.OK, ret.resultado);
}


@Test
public void testIngresarPalabraDiccionarioPalabraYaExistente() {
        Retorno ret = sis.crearSistemaMensajes();
        ret = sis.IngresarPalabraDiccionario("Palabra1");
        ret = sis.IngresarPalabraDiccionario("Palabra1");

        assertEquals(Resultado.ERROR_1, ret.resultado);
}


@Test
public void testBorrarPalabraDiccionario() {
        Retorno ret = sis.crearSistemaMensajes();
        ret = sis.IngresarPalabraDiccionario("Palabra1");
        ret = sis.BorrarPalabraDiccionario("Palabra1");

        assertEquals(Resultado.OK, ret.resultado);
}


@Test
public void testBorrarPalabraDiccionarioPalabranoExistente() {
        Retorno ret = sis.crearSistemaMensajes();
        ret = sis.IngresarPalabraDiccionario("Palabra1");
        ret = sis.BorrarPalabraDiccionario("Palabra 1");

        assertEquals(Resultado.ERROR_1, ret.resultado);
}
```

```
@Test
public void testImprimirDiccionario() {
        Retorno ret = sis.crearSistemaMensajes();
        ret = sis.IngresarPalabraDiccionario("Palabra1");
        ret = sis.IngresarPalabraDiccionario("Palabra2");
        ret = sis.IngresarPalabraDiccionario("Palabra3");
        ret = sis.ImprimirDiccionario();

        assertEquals(Resultado.OK, ret.resultado);
}


@Test
public void testImprimirTextoIncorrecto() {
        Retorno ret = sis.crearSistemaMensajes();
        ret = sis.InsertarLinea();
        ret = sis.InsertarPalabraEnLinea(1, 1, "Palabra1");
        ret = sis.InsertarLinea();
        ret = sis.InsertarPalabraEnLinea(2, 1, "Palabra2");
        ret = sis.InsertarLinea();

        ret = sis.IngresarPalabraDiccionario("Palabra1");
        ret = sis.ImprimirTextoIncorrecto();

        assertEquals(Resultado.OK, ret.resultado);
}
```

·Resultados JUnit: