UNIVERSIDAD DE ALMERÍA

Estructura de Datos y Algoritmos I

Estructuras de datos lineales en JCF

Práctica: 01

Curso: 2º Grado en Ingeniería Informática

Grupo docente: B1 **Grupo**: GTA2

Nombre y Apellidos:

Jefferson Max Tomalá Villarreal

PRÁCTICA 1
(sesiones 01 y 02)

1. **Ejercicio con colecciones lineales**.

1) **De las principales colecciones lineales implementados en la JCF de Java (`ArrayList`, `LinkedList`, `Vector`, `Stack`, etc.), razone detalladamente cuál es la que cree más conveniente para resolver este ejercicio.**

Para la implementación de este ejercicio 1, consideramos que es mejor hacer uso de una LinkedList en lugar de un ArrayList. La razón reside principalmente en que, como bien sabemos, una LinkedList ofrece mejor eficiencia en operaciones de inserción y eliminación, mientras que un ArrayList ofrece mejores tiempos en operaciones de búsqueda indexada. Basándonos en este hecho, podemos considerar que un editor de texto como el que hemos implementado realizará un mayor número de operaciones de inserción y eliminación en lugar de búsquedas indexadas. Si implementásemos el ejercicio con ArrayList, la eficiencia del programa podría verse afectada enormemente debido a que operaciones de inserción y eliminación pueden llevar tiempos de orden O(n).

2) **Para la resolución de este ejercicio se ha propuesto utilizar el `LinkedList`, ¿sería muy complicado realizar la misma implementación con `ArrayList`?, ¿qué habría que hacer?**

Hacer el cambio con un ArrayList no supondría mayor complicación que sustituir la estructura LinkedList por una estructura ArrayList:

```
//Declaración
private ArrayList<String> text;

//Inicializacion
text = new ArrayList<String>();
```

Podriamos seguir haciendo uso del mismo iterador y no sería necesario realizar ningún cambio más.

De hecho, se propone como alternativa, declarar la variable text como un tipo List y que sea el constructor el que determine si se hace uso de una LinkedList o un ArrayList.

```
// Variables
    private List<String> text;
    private ListIterator<String> current;
    private boolean inserting;

    //Constructor por defecto
    public Editor() {
        this(true);
```

```
}

//Constructor que permite decidir el tipo de ED
public Editor(boolean useLinkedList) {
	if(useLinkedList){
		text = new LinkedList<String>();
	}else{
		text=new ArrayList<String>();
	}
   current = text.listIterator();
   inserting = false;
}
```

2. Ejercicio con colecciones lineales un poco más complejas.

1) Para finalizar, como ya hemos podido comprobar, para la resolución de este ejercicio se ha propuesto utilizar el ArrayList. ¿Sería apropiado realizar la misma implementación con LinkedList?, ¿qué habría que hacer?

El uso de una estructura LinkedList en lugar de un ArrayList consideramos que sería una solución que podría mejorar la eficiencia única y exclusivamente en la operación de carga de datos en la estructura, que es donde tiene lugar la operación de inserción de elementos en la estructura de datos.

Sin embargo, resto de métodos en la implementación suponen realizar un recorrido por todos y cada uno de los elementos contenidos en la estructura, por lo que ambas estructuras (LinkedList y ArrayList) tendrían un coste O(n). Esto es debido a que hay que comprobar cada uno de los elementos con el fin de determinar si cumple las condiciones de búsqueda indicadas.

Al igual que en el ejercicio anterior, tan sólo sería necesario modificar el tipo de estructura en la declaración e instanciación.

```
private LinkedList<EmpresaProyectos> listaEmpresas = new
LinkedList<EmpresaProyectos>();

private LinkedList<ProyectoCiudades> proyectosCiudades;

private LinkedList<String> ciudades;
```