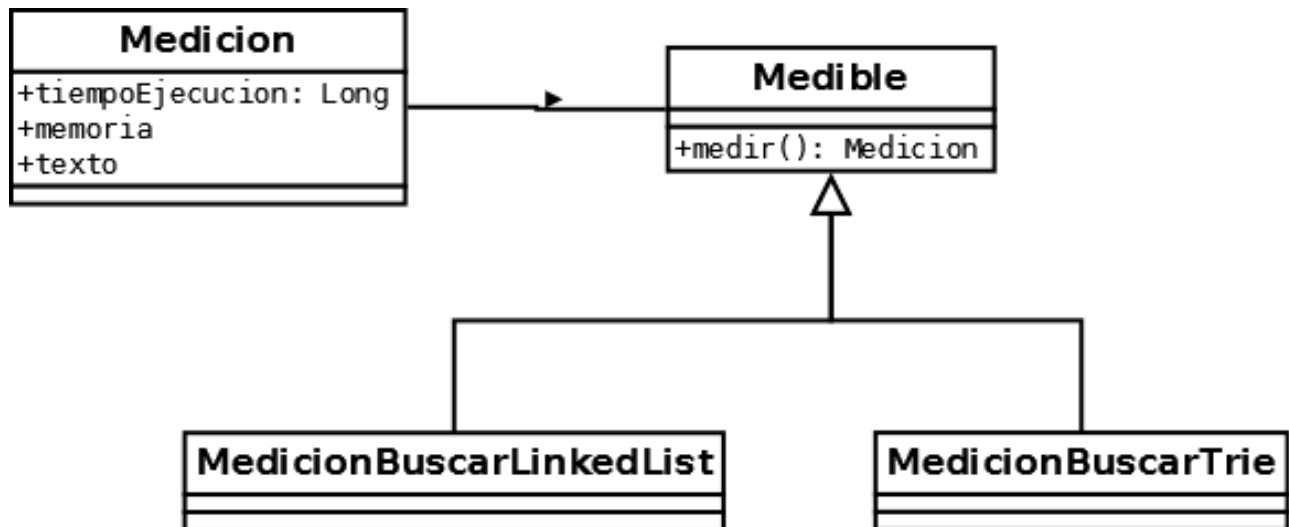


UNIDAD TEMÁTICA 3 – Diccionarios, Mapas, Hashing e implementaciones JAVA

Trabajo de Aplicación 3

Utilización de Medibles



Introducción:

La clase abstracta **Medible** contiene la lógica necesaria para calcular el tiempo de ejecución de un algoritmo y obtener el consumo de memoria de un objeto en particular.

Para implementar una clase **Medible**, por ejemplo, queremos medir el tiempo de ejecución de 100 búsquedas sobre una linkedlist, y también nos interesa saber el tamaño de tal colección en Bytes.

Heredando de esta clase y solo implementando el método que queremos medir, ya tenemos resuelta toda esa lógica.

Pasos:

1. Crear nueva clase
2. Agregar sentencia de herencia sobre la clase Medible: “extends Medible”.
3. Implementar métodos abstractos:

4. Completar el método **ejecutar**. En este método escribiremos todo el código que queremos analizar su tiempo de ejecución, por lo que hacemos la búsqueda:

```
@Override
public void ejecutar(Object... params) {
    int repeticion = (int) params[0];
    String[] palabras = (String[]) params[1];
    for(int i = 0; i < repeticion; i++){
        for(String palabra : palabras){
            linkedList.contains(palabra);
        }
    }
}
```

El método **ejecutar** recibe un array de parametros. En la primer posicion se encuentra la cantidad de iteraciones y en la segunda la lista de palabras para realizar la búsqueda.

- Completamos el método **getObjetoAMedirMemoria()**: Este método debe retornar el objeto que queremos medir la memoria.

```
@Override
public Object getObjetoAMedirMemoria() {
    return this.linkedList;
}
```

Ejemplo completo

```
1 public class MedicionLinkedList extends Medible{
2     private LinkedList linkedList;
3     public MedicionLinkedList(LinkedList linkedList) {
4         this.linkedList = linkedList;
5     }
6
7     @Override
8     public void ejecutar(Object... params) {
9         int repeticion = (int) params[0];
10        String[] palabras = (String[]) params[1];
11        for(int i = 0; i < repeticion; i++){
12            for(String palabra : palabras){
13                linkedList.contains(palabra);
14            }
15        }
16    }
17
18    @Override
19    public Object getObjetoAMedirMemoria() {
20        return this.linkedList;
21    }
22 }
```

Método principal del programa

```
1  DECLARACION DE PAQUETE
2  import java.util.*;
3
4  public class Main {
5
6      private static final int REPETICIONES = 100;
7
8      public static void main(String[] args){
9          TArbolTrie trie = new TArbolTrie();
10         LinkedList linkedList = new LinkedList();
11         ArrayList arrayList = new ArrayList();
12         HashMap hashMap = new HashMap();
13         TreeMap treeMap = new TreeMap();
14         String[] palabrasclave = ManejadorArchivosGenerico.leerArchivo(RUTA_PALABRAS_DESORDENADAS);
15         String[] palabrasBuscar = ManejadorArchivosGenerico.leerArchivo(RUTA_PALABRAS_BUSCAR);
16         for (String p : palabrasclave) {
17             trie.insertar(p);
18             linkedList.add(p);
19             arrayList.add(p);
20             hashMap.put(p, p);
21             treeMap.put(p,p);
22         }
23
24         Medible[] medibles = {new MedicionBuscarLinkedList(linkedList)};
25         //TODO: Cargar el resto de los medibles
26
27         Object[] params = {REPETICIONES, palabrasBuscar};
28         for (Medible m: medibles){
29             m.medir(params).print();
30         }
31     }
32 }
```
