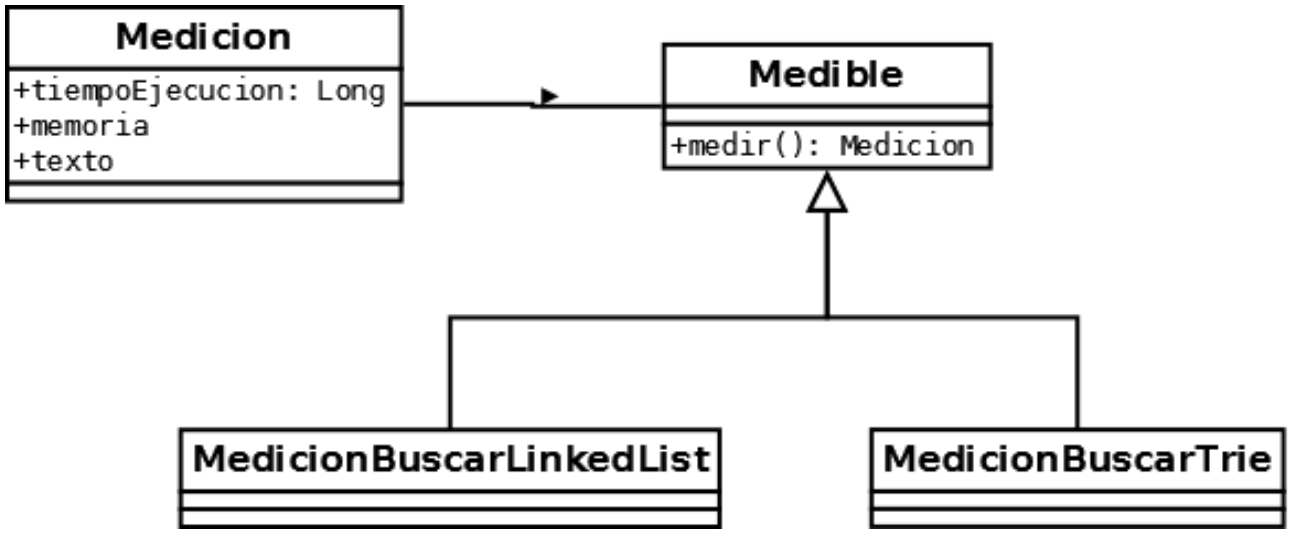
UNIDAD TEMÁTICA 6 – Diccionarios, Mapas, Hashing e implementaciones JAVA

Trabajo de Aplicación 3

Utilización de Medibles



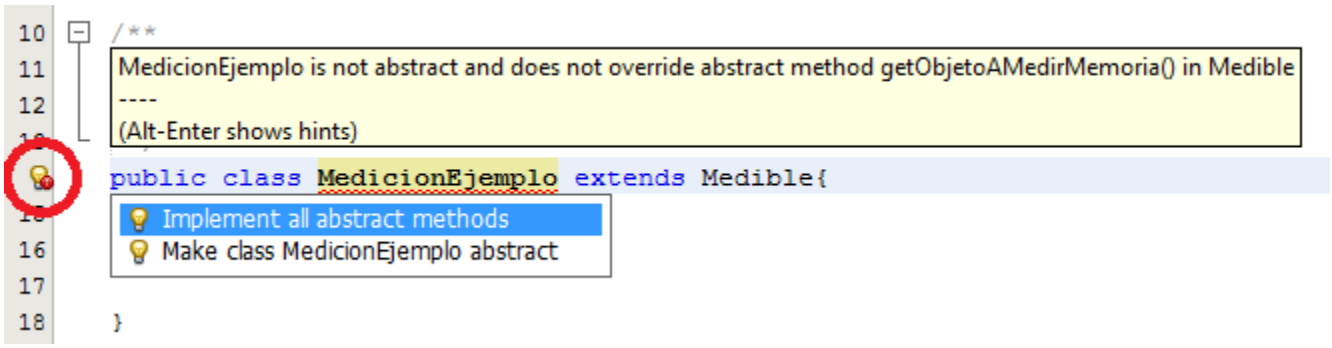
Introducción:

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.

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

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

Ejemplo completo

```java
public class MedicionLinkedList extends Medible{
    private LinkedList linkedList;
    public MedicionLinkedList(LinkedList linkedList) {
        this.linkedList = linkedList;
    }

    @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);
            }
        }
    }

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

Método principal del programa

```
DECLARACION DE PAQUETE
import java.util.*;

public class Main {

    private static final int REPETICIONES = 100;

    public static void main(String[] args){
        TArbolTrie trie = new TArbolTrie();
        LinkedList linkedList = new LinkedList();
        ArrayList arrayList = new ArrayList();
        HashMap hashMap = new HashMap();
        TreeMap treeMap = new TreeMap();
        String[] palabrasclave = ManejadorArchivosGenerico.leerArchivo(RUTA_PALABRAS_DESORDENADAS);
        String[] palabrasBuscar = ManejadorArchivosGenerico.leerArchivo(RUTA_PALABRAS_BUSCAR);
        for (String p : palabrasclave) {
                trie.insertar(p);
                linkedList.add(p);
                arrayList.add(p);
                hashMap.put(p, p);
                treeMap.put(p,p);
        }

        Medible[] medibles = {new MedicionBuscarLinkedList(linkedList)};
        //TODO: Cargar el resto de los medibles

        Object[] params = {REPETICIONES, palabrasBuscar};
        for (Medible m: medibles){
            m.medir(params).print();
        }
    }
}
```