



Problema resuelto usando listas enlazadas en Java

Un estudiante universitario necesita un sistema simple para gestionar sus tareas diarias. Desea agregar tareas a una lista, consultarlas en orden y poder eliminarlas cuando estén completadas.

Para realizar esta tarea se usará una lista enlazada para representar la lista de tareas. Se usará la clase LinkedList que internamente java trabaja como una lista enlazada, donde cada nodo está enlazado con una referencia al anterior y al siguiente, y a través de esta clase se hace más sencillo el manejo de los datos.

Historias de usuario

Historia de usuario	1°	Título: Agregar tareas
Descripción	Como:	Estudiante universitario.
	Quiero:	Registrar mis tareas en un programa.
	Para:	Poder llevar un control de las tareas que me falta por completar.
Criterios de aceptación	<ul style="list-style-type: none">• El programa debe aceptar varias tareas de tipo string que almacenará en una LinkedList.• El registro de tareas finaliza cuando se escribe en la consola "FIN".• Después debe imprimir la lista de tareas que se han escrito.	

Historia de usuario	2°	Título: Eliminar tareas.
Descripción	Como:	Estudiante universitario.

	Quiero:	Eliminar mis tareas y ver las faltantes cada vez que lo hago.
	Para:	Visualizar las tareas faltantes y administrar mi tiempo.
Criterios de aceptación	<ul style="list-style-type: none"> • Si no hay tareas pendientes el programa debe imprimir “Sin tareas pendientes”. • Al eliminar una tarea se debe eliminar de la lista enlazada y después imprimir las listas faltantes. • Si una tarea no se encuentra en la lista, se debe imprimir “Tarea no encontrada” • Si se escribe en la consola “FIN” el programa debe finalizar. 	

Formato de la entrada y salida:

Entrada	Salida
Al ingresar tareas, cada vez que se ingresa texto se almacena como una string, y en caso de escribir “FIN” entonces se pasa a la etapa de visualización y eliminación de tareas. Desde aquí, cada entrada que se de, el programa buscará ese mismo string en la lista, y la eliminará en caso de encontrarla. Si la entrada es “FIN” el programa finalizará.	La salida debe ser primero la lista ordenada en el mismo orden en el que se ingresó, si no hay tareas pendientes imprimirá “Sin tareas pendientes”. Después, cada vez que se ingrese alguna tarea, en caso de que no esté en la lista imprimirá “Tarea no encontrada”, si no es el caso la eliminará de la lista. Cada vez que se ingrese un nuevo texto volverá a hacer este proceso.

Instrucciones para la calificación automática

- El nombre de la clase debe ser: Main
- El método principal debe ser public static void main(String[] args)
- Usar Scanner para leer las entradas desde consola.
- No se debe imprimir texto adicional fuera de lo indicado.

Ejemplos de entrada y de salida esperados.

Entrada	Salida esperada
----------------	------------------------

Sacar al perro Estudiar Lavar la ropa FIN Estudiar FIN	Sacar al perro Estudiar Lavar la ropa Sacar al perro Lavar la ropa
Estudiar Leer Barrer FIN Limpiar Leer Estudiar Barrer FIN	Estudiar Leer Barrer Tarea no encontrada Estudiar Leer Barrer Estudiar Barrer Barrer Sin tareas pendientes

Código en Java

```

import java.util.LinkedList;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        LinkedList<String> tareas = new LinkedList<>();

        // Leer tareas
        while (true) {
            String entrada = scanner.nextLine();
            if (entrada.equals("FIN")) break;
            tareas.add(entrada);
        }

        while (true) {
            if (tareas.isEmpty()) {
                System.out.println("Sin tareas pendientes");
            } else {
                for (String tarea : tareas) {
                    System.out.println(tarea);
                }
            }
            String tareaEliminar = scanner.nextLine();
            if (tareaEliminar.equals("FIN")) break;
        }
    }
}

```

```

        if (tareas.contains(tareaEliminar)) {
            tareas.remove(tareaEliminar);
        } else {
            System.out.println("Tarea no encontrada");
        }
    }
}
}
}

```

1. Problema resuelto usando listas enlazadas

Un estudiante universitario necesita un sistema simple para gestionar sus tareas diarias. Desea agregar tareas a una lista, consultarlas en orden y poder eliminarlas cuando estén completadas.

Para realizar esta tarea se usará una lista enlazada para representar la lista de tareas. Se usará la clase LinkedList que internamente java trabaja como una lista enlazada, donde cada nodo está enlazado con una referencia al anterior y al siguiente, y a través de esta clase se hace más sencillo el manejo de los datos.

Historias de usuario

Historia de usuario	1°	Título: Agregar tareas
Descripción	Como:	Estudiante universitario.
	Quiero:	Registrar mis tareas en un programa.
	Para:	Poder llevar un control de las tareas que me falta por completar.
Criterios de aceptación	<ul style="list-style-type: none"> • El programa debe aceptar varias tareas de tipo string que almacenará en una LinkedList. • El registro de tareas finaliza cuando se escribe en la consola "FIN". • Después debe imprimir la lista de tareas que se han escrito. 	

Historia de usuario	2°	Título: Eliminar tareas.
Descripción	Como:	Estudiante universitario.
	Quiero:	Eliminar mis tareas y ver las faltantes cada vez que lo hago.
	Para:	Visualizar las tareas faltantes y administrar mi tiempo.
Criterios de aceptación	<ul style="list-style-type: none"> ● Si no hay tareas pendientes el programa debe imprimir “Sin tareas pendientes”. ● Al eliminar una tarea se debe eliminar de la lista enlazada y después imprimir las listas faltantes. ● Si una tarea no se encuentra en la lista, se debe imprimir “Tarea no encontrada” ● Si se escribe en la consola “FIN” el programa debe finalizar. 	

Formato de la entrada y salida:

Entrada	Salida
Al ingresar tareas, cada vez que se ingresa texto se almacena como una string, y en caso de escribir “FIN” entonces se pasa a la etapa de visualización y eliminación de tareas. Desde aquí, cada entrada que se de, el programa buscará ese mismo string en la lista, y la eliminará en caso de encontrarla. Si la entrada es “FIN” el programa finalizará.	La salida debe ser primero la lista ordenada en el mismo orden en el que se ingresó, si no hay tareas pendientes imprimirá “Sin tareas pendientes”. Después, cada vez que se ingrese alguna tarea, en caso de que no esté en la lista imprimirá “Tarea no encontrada”, si no es el caso la eliminará de la lista. Cada vez que se ingrese un nuevo texto volverá a hacer este proceso.

Instrucciones para la calificación automática

- El nombre de la clase debe ser: Main
- El método principal debe ser public static void main(String[] args)

- Usar Scanner para leer las entradas desde consola.
- No se debe imprimir texto adicional fuera de lo indicado.

Ejemplos de entrada y de salida esperados.

Entrada	Salida esperada
Sacar al perro Estudiar Lavar la ropa FIN Estudiar FIN	Sacar al perro Estudiar Lavar la ropa Sacar al perro Lavar la ropa
Estudiar Leer Barrer FIN Limpiar Leer Estudiar Barrer FIN	Estudiar Leer Barrer Tarea no encontrada Estudiar Leer Barrer Estudiar Barrer Barrer Sin tareas pendientes

Código en Java

```
import java.util.LinkedList;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        LinkedList<String> tareas = new LinkedList<>();

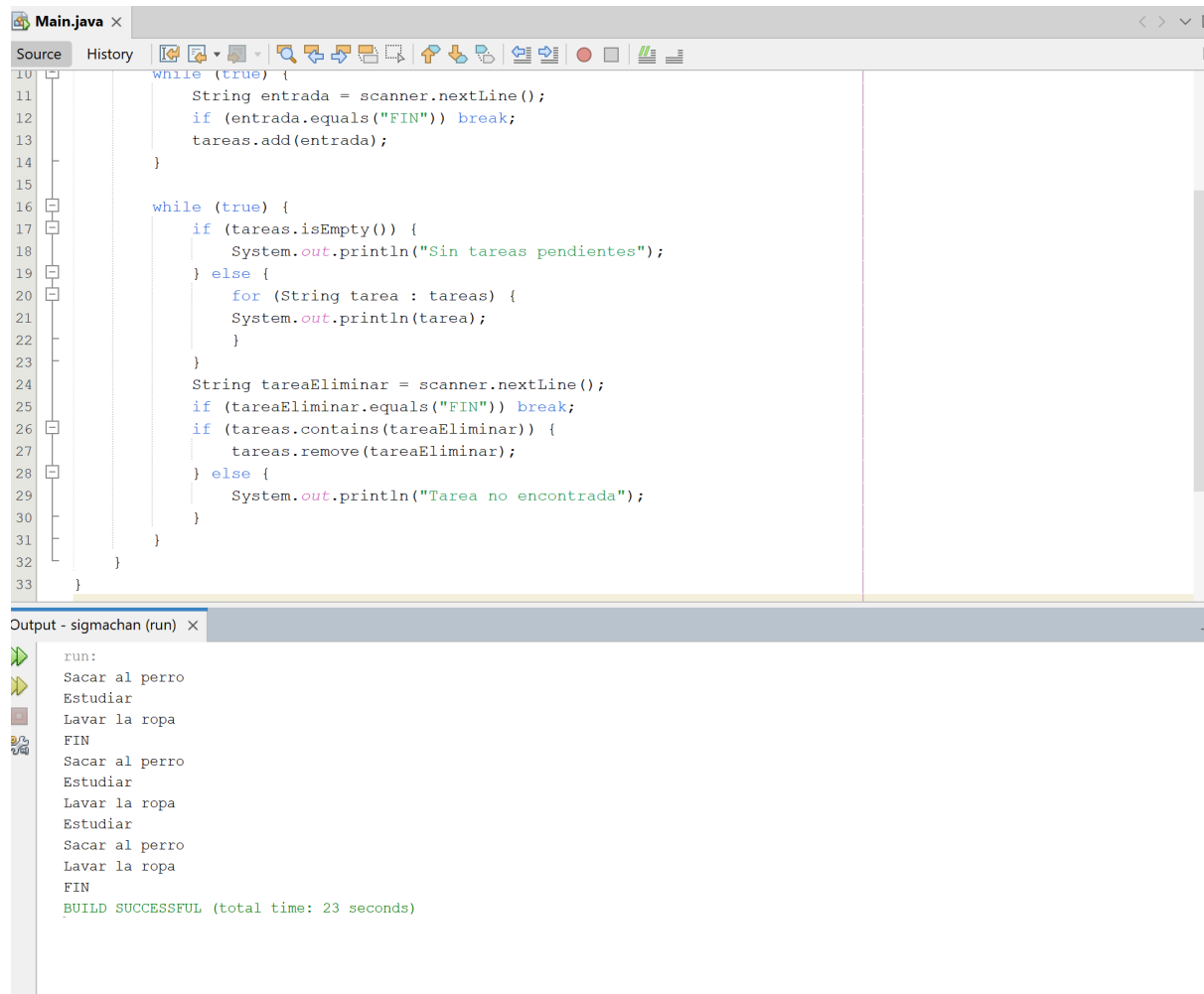
        // Leer tareas
        while (true) {
            String entrada = scanner.nextLine();
            if (entrada.equals("FIN")) break;
            tareas.add(entrada);
        }

        while (true) {
            if (tareas.isEmpty()) {
```

```
        System.out.println("Sin tareas pendientes");
    } else {
        for (String tarea : tareas) {
            System.out.println(tarea);
        }
    }
    String tareaEliminar = scanner.nextLine();
    if (tareaEliminar.equals("FIN")) break;
    if (tareas.contains(tareaEliminar)) {
        tareas.remove(tareaEliminar);
    } else {
        System.out.println("Tarea no encontrada");
    }
}
}
```

Evidencia de ejecución

Al ejecutar el programa y usando el primer caso de ejemplo sucede lo siguiente:



The screenshot shows an IDE window titled "Main.java" with a source code editor and an output console. The code is a Java program that manages a list of tasks. It uses a Scanner to read input from the user. The first while loop adds tasks to a list until the user enters "FIN". The second while loop prints the tasks if the list is not empty, or a message if it is. It also allows the user to remove a task by entering its name, or prints a message if the task is not found. The output console shows the execution of the program, displaying the tasks added and removed, and a final success message.

```
10 while (true) {
11     String entrada = scanner.nextLine();
12     if (entrada.equals("FIN")) break;
13     tareas.add(entrada);
14 }
15
16 while (true) {
17     if (tareas.isEmpty()) {
18         System.out.println("Sin tareas pendientes");
19     } else {
20         for (String tarea : tareas) {
21             System.out.println(tarea);
22         }
23     }
24     String tareaEliminar = scanner.nextLine();
25     if (tareaEliminar.equals("FIN")) break;
26     if (tareas.contains(tareaEliminar)) {
27         tareas.remove(tareaEliminar);
28     } else {
29         System.out.println("Tarea no encontrada");
30     }
31 }
32 }
33 }
```

Output - sigmachan (run) X

```
run:
Sacar al perro
Estudiar
Lavar la ropa
FIN
Sacar al perro
Estudiar
Lavar la ropa
Estudiar
Sacar al perro
Lavar la ropa
FIN
BUILD SUCCESSFUL (total time: 23 seconds)
```