Universidad de las Américas

FACULTAD: Ingeniería y Ciencias Aplicadas **PROFESOR:** Bernarda Sandoval
CARRERA: Ciberseguridad **ESTUDIANTE:** Bryan Quevedo, José Zambrano

CURSO: ISWZ2102-2049 **DESCRIPCIÓN:** Taller semana 6

ASIGNATURA: Programación 3

TEMA:

Informe de Implementación de Métodos de Búsqueda en Java

1. Introducción

Este informe presenta el desarrollo e implementación de un método de búsqueda dentro de un sistema de gestión de paquetes programado en Java. El trabajo se realizó siguiendo las indicaciones de la actividad, utilizando el código base presentado en clase, aplicando conceptos fundamentales de programación orientada a objetos y realizando la implementación final dentro del IDE IntelliJ IDEA.

El enfoque principal del proyecto fue la implementación del método de búsqueda binaria, aprovechando su eficiencia en comparación con la búsqueda lineal. Para garantizar un diseño flexible y escalable, se emplearon interfaces que permiten intercambiar fácilmente diferentes algoritmos de búsqueda dentro del sistema.

2. Objetivo del Proyecto

- Implementar un método de búsqueda eficiente para localizar paquetes por número de tracking.
- Utilizar interfaces para desacoplar la lógica de búsqueda de la estructura de datos.
- Aplicar principios correctos de programación orientada a objetos.
- Facilitar futuras extensiones, como el uso de búsqueda secuencial, interpolada o por cadenas.

3. Diseño General del Sistema

El sistema está compuesto por tres elementos principales:

3.1. Clase Paquete

Representa cada paquete registrado. Contiene información como:

- Número de tracking (ntrack)
- Peso
- Ciudad de recepción
- Cédula del receptor

Además, implementa la interfaz Comparable para permitir ordenar los paquetes por peso.

3.2. Clase Lista

Es la estructura encargada de almacenar y gestionar los objetos Paquete. Sus funciones principales son:

- Registrar nuevos paquetes.
- Buscar paquetes por número de tracking.
- Eliminar paquetes.
- Ordenar por peso.

Utiliza un atributo de tipo BusquedaPaquete (interfaz), lo que permite cambiar el método de búsqueda sin modificar la estructura interna.

3.3. Interfaz Gráfica VentanaPaquete

Esta interfaz permite al usuario:

- Ingresar datos de paquetes.
- Guardarlos en la lista.
- Buscar por número de tracking.
- Eliminar paquetes.
- Ordenar por peso.
- Visualizar el resultado de cada operación.

Esta ventana no depende del algoritmo de búsqueda utilizado gracias al uso de interfaces.

4. Método Implementado: Búsqueda Binaria

Para este proyecto se implementó el algoritmo de búsqueda binaria, uno de los métodos más eficientes cuando se trata de encontrar elementos en una lista ordenada.

4.1. Interfaz de búsqueda

Se creó la interfaz:

```
public interface BusquedaPaquete {
   Paquete buscar(List<Paquete> datos, int ntrack);
}
```

Esta interface define el comportamiento de cualquier algoritmo de búsqueda que busque un paquete en una lista.

4.2. Implementación de búsqueda binaria

La clase BusquedaBinariaPaquete implementa el método definido en la interfaz. Este algoritmo:

1. Ordena una copia de la lista por el número de tracking.
2. Define índices izquierdo y derecho.
3. Calcula la posición media.
4. Compara el valor buscado con el elemento en la posición media.
5. Decide si continúa buscando en la mitad izquierda o la derecha.
6. Finaliza cuando encuentra el paquete o cuando los índices se cruzan.

Este proceso reduce drásticamente el número de comparaciones necesarias.

Complejidad temporal:

- O(log n), lo cual es significativamente superior a la búsqueda lineal que es O(n).

4.3. Integración con la Clase Lista

La clase Lista usa esta implementación por defecto:

```
this.buscador = new BusquedaBinariaPaquete();
```

Cuando la interfaz gráfica solicita una búsqueda con:

```
listaPaquetes.buscarPorNtrack(valor)
```

La llamada es redirigida al algoritmo binario automáticamente.

5. Funcionamiento del Sistema

El flujo general de la aplicación es el siguiente:

1. El usuario ingresa los datos del paquete en la ventana.
2. Al presionar Guardar:
    - Se crea un objeto Paquete.
    - Se guarda en la lista.
    - Se limpian los campos de texto.
3. Al presionar Buscar:
    - El sistema aplica búsqueda binaria.
    - Muestra toda la información del paquete si existe.
4. Al presionar Eliminar:
    - Se busca el paquete por ntrack.
    - Si existe, se elimina.
5. Al presionar Ordenar por peso:
    - La lista se ordena utilizando Comparable.
    - Se muestra el listado ordenado.

6. Enfoque Utilizado

6.1. Programación Orientada a Objetos

Se aplicaron principios de:

- Encapsulamiento
- Abstracción
- Modularidad
- Bajo acoplamiento / alta cohesión

La lógica de búsqueda no está mezclada con la interfaz gráfica ni con la estructura de datos. Cada clase tiene una única responsabilidad.

6.2. Uso de Interfaces para Flexibilidad

La gran ventaja del diseño es que:

- Se puede cambiar el algoritmo de búsqueda (lineal, binaria) sin modificar la GUI.
- Para agregar un nuevo método solo se requiere implementar la interfaz.

Esto es un ejemplo práctico del principio Open/Closed (OCP):

7. Resultados Obtenidos

Los resultados del trabajo son los siguientes:

Búsqueda binaria implementada exitosamente

La aplicación localiza paquetes de forma rápida y eficiente, incluso cuando la lista crece.

Interfaz gráfica funcional

Permite guardar, eliminar, buscar y ordenar sin errores.

Los campos se limpian automáticamente al guardar un paquete, mejorando la experiencia del usuario.

Arquitectura clara y modular

El uso de la interfaz BusquedaPaquete permitió separar completamente:

- Lógica de búsqueda,
- Administración de datos,
- Interfaz gráfica.

Facilidad para futuras mejoras

Pueden implementarse nuevas búsquedas (interpolada, cadenas, secuencial, etc.) sin reescribir el programa.

8. Conclusión

El proyecto logró implementar de manera correcta el método de búsqueda binaria dentro del sistema de gestión de paquetes. El uso de interfaces y la programación orientada a objetos permitieron desarrollar una solución escalable, organizada y eficiente. Además, la interfaz gráfica ofrece una interacción intuitiva para el usuario final.

El enfoque adoptado garantiza que el sistema pueda ser extendido fácilmente, reemplazando o agregando nuevas estrategias de búsqueda sin alterar la estructura ya existente. Esto demuestra una buena práctica de diseño de software y facilita el trabajo colaborativo.

Link de github:

https://github.com/baquevedog/Taller-semana-6.git