UNIVERSIDAD TECMILENIO

Proyecto Final

Estructuras de datos

Adalberto Emmanuel
Rojas Perea

Leonardo Pereda Chánez
AL03087353

Documentación del código

ColaPrioridad

Este paquete implementa una **Cola con Prioridad** a través de una lista enlazada ordenada. Su función principal es gestionar las tareas complejas del sistema, asegurando que las más urgentes se atiendan primero.

- **Clase Tarea:** Contiene atributos como descripción, un nivel de prioridad (donde 1 es lo más alto), una fecha de entrega y las horas estimadas para su finalización. Cada tarea recibe un ID único y que se autoincrementa al momento de ser creada.
- **Clase ColaConPrioridad:** Utiliza un método **push** que inserta cada nueva tarea en la posición correcta de la lista, manteniendo siempre el orden. El criterio de ordenación es primero por nivel de prioridad y, en caso de que sea igual, se ordena por la fecha de entrega más cercana. Esto asegura que el método **pop** siempre devuelva la tarea más crítica del momento.
- **Clase Nodo:** Es el nodo genérico que usa la lista enlazada para almacenar cada **Tarea** y su apuntador al siguiente elemento.

ArbolBinario

Se encarga de toda la gestión de personal de la empresa, utilizando un Árbol Binario para un almacenamiento y acceso eficiente.

- **Clase Empleado:** Modela la información de un empleado, incluyendo un **id** único que funciona como la llave en el árbol, nombre y departamento.
- **Clase ArbolBinario:** Los empleados se insertan y ordenan basándose en su **id**. Incluye métodos para **insertar**, **buscar** y **eliminar** empleados. La eliminación maneja los diferentes casos posibles para mantener la

integridad del árbol. El método **mostrar** realiza un recorrido **in-orden** para listar a todos los empleados de forma ordenada.

- **Clase NodoArbol:** Es el nodo específico del árbol, conteniendo un objeto **Empleado** y las referencias a sus hijos izquierdo y derecho.

HashMap

Este paquete implementa un sistema de registros centrales utilizando **HashMap** para garantizar el acceso instantáneo a cualquier tarea o empleado si se conoce su ID.

- **Clase Registros:** Funciona como la "base de datos". Contiene dos **HashMap** principales: uno para **Empleados** y otro para **Tareas**. Proporciona métodos simples como **agregar**, **buscar** y **eliminar**, haciendo que el sistema pueda acceder a la información de manera rápida por medio de las llaves. También incluye los métodos **obtenerTodasLasTareas** y **mostrarEmpleados** para proporcionar reportes.

Algoritmos

Contiene una colección de algoritmos de ordenamiento y búsqueda.

- **Clase Algoritmos:**
  - **bubbleSort**: Implementa el método de ordenamiento de burbuja. Es utilizado en el sistema para generar reportes de tareas ordenadas por diferentes criterios como lo es la fecha, la prioridad y el ID.
  - **busquedaBinaria**: Implementa la búsqueda binaria. Se utiliza para buscar tareas por ID de forma eficiente.

Recursividad y DivideYVenceras

Estos paquetes contienen algoritmos avanzados para la planificación estratégica y el análisis de datos.

- **Recursividad/CalcStats:** Proporciona un método recursivo, **calcularTiempoTotal**, que toma una lista de tareas y calcula la suma total de sus **horasEstimadas**. El caso base es una lista vacía, y el paso recursivo consiste en sumar la hora de la primera tarea con el cálculo del resto de la lista.
- **DivideYVenceras:** Implementa la distribución equitativa de tareas de un proyecto.
    - **Proyecto y Asignacion:** Clases que estructuran un proyecto y la asignación de tareas a un empleado.
    - **DistribuidorTareas:** Sigue la estrategia de "Divide y Vencerás". Toma una lista grande de tareas y la "divide" recursivamente, asignando una tarea a la vez a un empleado y rotando la lista de empleados para asegurar un reparto justo. La conquista es la asignación misma, y la combinación es el resultado final de todas las asignaciones recursivas.

LinkedList

Contiene las clases que fueron implementadas en el avance de proyecto, son utilizadas por los roles operativos para tareas simples y gestión de incidentes.

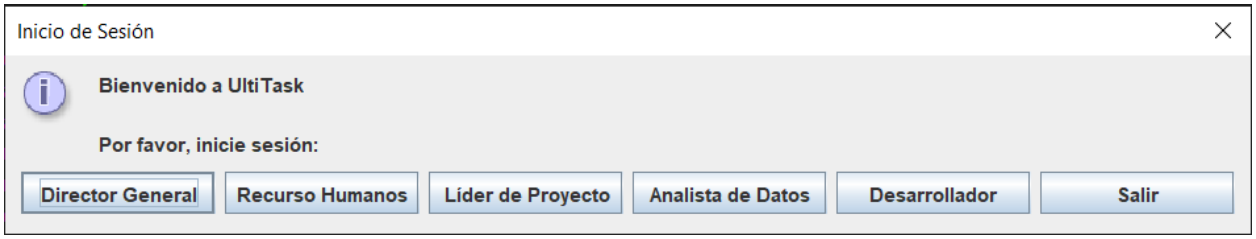
- **Stack:** Una Pila (LIFO) usada para gestionar incidentes críticos.
- **Queue:** Una Cola (FIFO) para tareas programables.

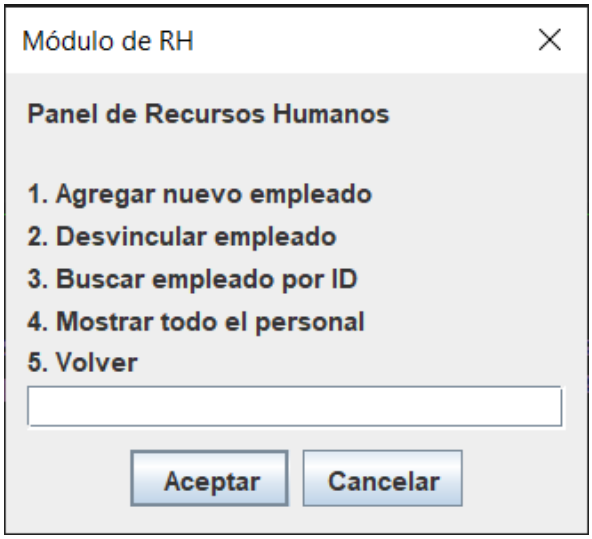
- **LinkedList:** Una Lista Enlazada simple para almacenar tareas departamentales. El método **findByDepartment** permite buscar tareas específicas por departamento.
- **NodeLL:** El nodo base para la **LinkedList**.

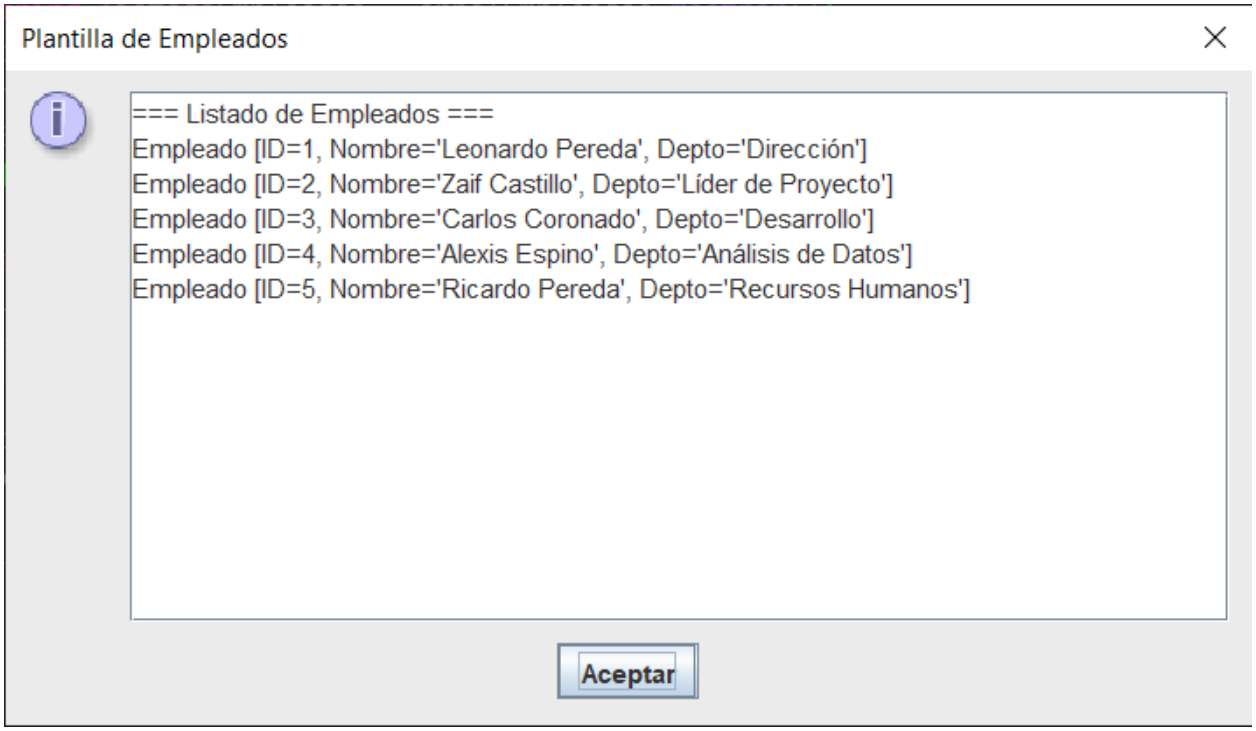
Main.java

1. **Inicio del sistema:** Al ejecutarse, la clase **Main** crea instancias globales y estáticas de todas las estructuras de datos principales: el **ArbolBinario** para empleados, los **Registros** para acceso rápido, la **ColaConPrioridad** para tareas con prioridades, **Stack**, **Queue** y **LinkedList** para tareas urgentes, programadas o por departamento.
2. **Datos de ejemplo:** El método datosBase() se llena con datos precargados para evitar que el usuario tenga que ingresar datos manualmente y que acceda rápidamente a las funciones que ofrece el sistema.
3. **Control de Acceso:** El main establece un inicio de sesión con los diferentes roles de acceso, y cada uno cuenta con funciones específicas de acuerdo al puesto dentro de la empresa/negocio.
4. **Interfaz de Usuario (UI):** Toda la interacción con el usuario (menús, solicitudes de datos, mensajes de error y resultados) se gestiona a través de la librería de **JOptionPane**. Se implementaron métodos de ayuda como **solicitarNumero**, **solicitarFecha** y **mostrador** para estandarizar la entrada y salida de datos, manejar errores y presentar información extensa de manera legible en ventanas.

Evidencias de funcionamiento







Conclusión

La planeación e implementación del proyecto ha sido compleja, sobre todo hacer que todas las clases tengan funciones prácticas y que trabajen entre sí, y pues es importante aprender sobre los diferentes tipos de estructuras y cómo se pueden implementar en ámbitos de la vida real.

Es un proyecto con un alcance un poco grande, supongo que podría colaborar con algún compañero más al trabajar con proyectos con alcances mayores.