Exercise-4

The Employee Management System is a Java-based application that allows users to manage employee records. It provides functionalities to add, search, display, and delete employee information, as well as maintain a simple database of employee IDs and salaries.

Classes

1. Employee

This class represents an individual employee with attributes such as ID, name, position, and salary.

Methods:

Getters for all attributes

toString()`: Returns a string representation of the employee

2. EmployeeDatabase

This class handles the persistence of employee data to a text file.

Attributes:

`DATABASE\_FILE`: String (constant)

Methods:

- `updateDatabase(List<Employee> employees)`: Writes employee data to the file

- `displayDatabase()`: Reads and displays the contents of the database file

3. EmployeeManagementSystem

This class manages the core functionalities of the system, including adding, searching, displaying, and deleting employees.

Attributes:

- `employees`: Employee[] (array to store employees)

- `size`: int (current number of employees)

- `database`: EmployeeDatabase (instance to handle database operations)

Methods:

- `addEmployee(Employee employee)`: Adds a new employee to the system

- `searchEmployee(int employeeId)`: Searches for an employee by ID

- `traverseEmployees()`: Displays all employees in the system

- `deleteEmployee(int employeeId)`: Deletes an employee from the system

- `displayDatabase()`: Displays the contents of the database

4. Main

This class contains the main method and handles user interaction through a console menu.

Methods:

- `main(String[] args)`: Entry point of the application, manages the menu loop

- `addEmployee(EmployeeManagementSystem ems, Scanner scanner)`: Handles user input for adding an employee

- `searchEmployee(EmployeeManagementSystem ems, Scanner scanner)`: Handles user input for searching an employee

- `deleteEmployee(EmployeeManagementSystem ems, Scanner scanner)`: Handles user input for deleting an employee

Functionality

1. \*\*Add Employee\*\*: Users can add a new employee by entering their ID, name, position, and salary.

2. \*\*Search Employee\*\*: Users can search for an employee by their ID.

3. \*\*Display All Employees\*\*: The system can display all employees currently in the system.

4. \*\*Delete Employee\*\*: Users can delete an employee by their ID.

5. \*\*Display Database\*\*: The system can display the contents of the database file, showing employee IDs and salaries.

Time Complexity Analysis

- Add Employee: O(1) amortized (O(n) when array resizing is needed)

- Search Employee: O(n) in the worst case

- Display All Employees: O(n)

- Delete Employee: O(n) in the worst case

- Database Update: O(n) as it writes all employees to the file each time

Where n is the number of employees in the system.

Limitations and Potential Improvements

1. The current implementation uses an array, which has limitations in terms of dynamic sizing and efficiency for large datasets.

2. The database is a simple text file, which may not be suitable for large-scale applications or concurrent access.

3. The search operation is linear, which can be slow for large datasets.

Potential improvements could include:

- Using more efficient data structures like ArrayList or HashMap

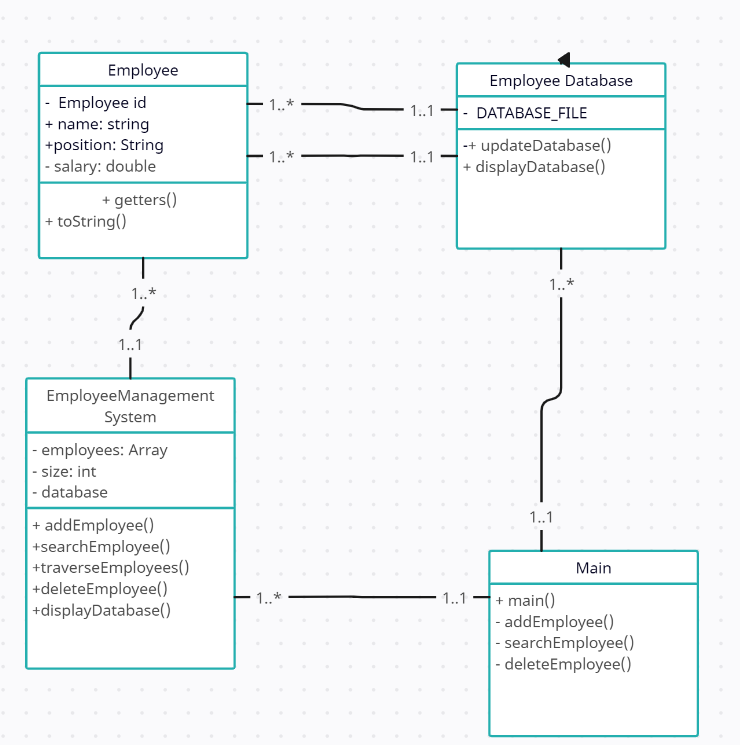
- Implementing a proper database system for better data management

- Adding more advanced search and sort functionalities

- Implementing error handling and input validation

- Adding a graphical user interface for better user experience

**Class Diagram**



Implementation

Link: [code is here](https://github.com/Akashmondal55/Akash_5016855/tree/main/Week-1/DSA/Exercise-4)