Employee Management System Documentation

---

1. Project Overview

The Employee Management System is a Java-based console application that allows easy maintenance of employee records by an HR administrator, as well as upkeep of department-related information. It leverages the OOP principles, Java Collections, and exception handling for reliability and ease of use.

---

2. How to Run the Application

Pre-Requisites:

- **Java Development Kit (JDK)**: The system should contain JDK 8 or above. IDE: If so desired, this can be done in an IDE environment such as IntelliJ IDEA or Eclipse, or a code editor like VS Code for better handling and execution, or from the command line itself.

Steps to Execute:

1. **Clone or Download Project:**
    - Clone the project from version control, or directly download the project files.
2. **Open Project in IDE:**
    - Open any desired IDE and import the above project directory as a new project.
3. **Compile the Code:**
    - The project should be compiled automatically by an IDE for the most part.
4. **Run the Main Class:**
    - Inside the `main` package, there is a file named `EmployeeManagementSystem.java`.
    - Right click on the file and press "Run `EmployeeManagementSystem.main()`" or if using via terminal/command prompt:

      ```
      javac src/main/EmployeeManagementSystem.java
      java src/main/EmployeeManagementSystem
      ```

5. **Interaction:**
    - The application will start in the console and provide options through a menu. Operate them as per instructions to go through the system.

---

3. How Each Feature Works

1. **Adding an Employee:**
    - **Objective:** To include a new employee into the system.
    - **Procedure:**:
        - Select option `1`.
        - Enter the employee's ID, name, designation, salary, and department ID.

        - The system will check for any invalid entry and confirm the addition.

2. **Remove Employee:**

    - **Description**: This operation is used to delete an already existing employee from the system.
    - **Procedure:**:
        - Use menu option `2`.
            - Enter the ID of the employee
            - The system will attempt to remove the employee and will notify you if the employee has been removed, or that the ID doesn't exist.

3. **Update Employee:**

    - **Description**: To update details of an existing employee.
    - **Procedure:**
        - Select option `3` from the menu.
        - Enter ID of employee which you want to update.
        - Enter new info like name, designation, salary, department ID, etc.
        - It will validate and update the info of that employee.

4. **Add Department:**

    - **Purpose**: Add a new department into the system.
    - **Procedure:**
        - Select menu option `4`.
        - Provide input for the department details including ID, name and description.
        - The system performs validation on all the inputs, hence confirms whether the addition is done or not.

5. **Delete Department:**

    - **Description**: To delete an existing department from the system .
    - **Procedure**:
        - In the menu, select option `5`.
        - Enter the ID of the department.
        - The system will try to delete the department, letting you know if it was successfully deleted or if the ID does not exist.

6. **Assign Employee to Department:**

    - **Purpose**: To assign an employee to a specific department.
    - **Procedure:**
        - Then, menu option `6`.
        - Then enter the employee ID and the department ID.
        - The system will update the employee's department assignment.

7. **Update Employee Department:**

    - **Purpose**: In this module, a program is supposed to update the department assignment of an already existing employee.
    - **Procedure**:

- After showing the menu option `7`.
- Enter the employee ID and the new department ID.
- Update the department assignment by the system.

8. **List All Employees:**
    - **Purpose**: To display all employees in the system.
    - **Instructions**:
        - Select option `8` from the menu.
        - It will list all employees with their details.

9. **List All Departments:**
    - **Purpose**: To show all the departments in the system
    - **Instructions**:
        - The user can select option `9` from the menu.
        - The system will list all departments with their details.

10. **Exit the Application:**
    - **Purpose**: The user can safely leave the application.
    - **Procedure:**
        - On the Menu, select option `0`.
        - The system will exit out of the program.

---

4. Description of Implemented Exception Handling

Exception handling of the Employee Management System is the feature that makes the application robust and user-friendly.

Custom Exceptions:

1. **EmployeeNotFoundException:**
    - **Purpose**: This is thrown when a certain operation is being performed on an employee who does not exist within the system.
    - **Usage**:- Methods such as `removeEmployee`, `updateEmployee`, and `assignEmployeeToDepartment` in `EmployeeService` apply this exception to notify the user in case of providing an invalid employee ID.
2. **DepartmentNotFoundException:**
    - **Purpose**: This exception will be thrown when an operation is being conducted on a non-existent department in the system.
        - **Usage**:- Methods like `removeDepartment`, `updateDepartment` and `assignEmployeeToDepartment` in `DepartmentService` uses this exception to notify the user when an invalid department id is being supplied.

Input Validation:

This utility class is used to validate user inputs before processing. For example, it checks that IDs are positive integers, names contain only letters and spaces, and salaries are positive numbers. If in case of failure in such validations, the system will prompt a message for the user to enter valid data.

Error Messages:

- Whenever the program catches an exception, it's meant to print out a friendly error message saying what went wrong; hence, in this way, the user will be in a position to correct their input or even know what exactly happened.
- The system doesn't hang itself but rather does bounces back to the main menu for the user's next course of action, without having to restart the application.