Project Documentation

Employee Management System (Java)

1 Project Overview

Project Title

Employee Management System using Java Collections & File Handling

Description

The Employee Management System is a console-based Java application developed to manage employee records efficiently. The project focuses on applying **Object-Oriented Programming (OOP)** principles along with **Java Collections**, **File Handling**, and **Exception Handling**.

Employee data is stored persistently using **Java Object Serialization**, ensuring that records are preserved between program executions.

2 Project Goals & Objectives

- To understand real-world usage of Java Collections (`ArrayList`, `HashMap`)
- To implement CRUD operations on employee data
- To apply Java file handling using serialization
- To design modular and maintainable Java code
- To gain hands-on experience with exception handling
- To prepare an interview-ready Java mini project

3️⃣ Setup Instructions

System Requirements

- Java Development Kit (JDK) 8 or above
- Any Java IDE (Eclipse / STS / IntelliJ) or Command Line
- Windows / Linux / macOS

Installation Steps

Using IDE

1. Create a new **Java Project**
2. Copy all `.java` files into the `src` folder
3. Ensure all files are in the **default package**
4. Run `EmployeeManagementSystem.java`

Using Command Line

```
javac *.java
java EmployeeManagementSystem
```

4️⃣ Code Structure & File Hierarchy

```
EmployeeManagementSystem/
│
├── Employee.java
├── EmployeeManagementSystem.java
├── EmployeeFileHandler.java
├── EmployeeReportGenerator.java
├── data/
│   └── employees.ser
├── docs/
│   └── Project_Documentation.md
```

```
└── README.md
```

File Responsibilities

- **Employee.java** → Employee model (Serializable)
- **EmployeeManagementSystem.java** → Main menu & control flow
- **EmployeeFileHandler.java** → File save/load operations
- **EmployeeReportGenerator.java** → Report generation & analytics

5️⃣ Data Format Specification

File Name

`employees.ser`

File Type

Binary file generated using Java Serialization

Stored Data

- `ArrayList<Employee>` object

Employee Attributes

Field	Type
id	String
name	String
department	String
position	String
salary	double
joinDate	LocalDate

⚠️ This file is **not human-readable** and should not be edited manually.

6️⃣ File Handling Procedures

Saving Data

- Uses `ObjectOutputStream`
- Writes complete employee list to file

Loading Data

- Uses `ObjectInputStream`
- Reconstructs employee objects at runtime

Error Handling

- Handles `IOException` and `ClassNotFoundException`
- Ensures application stability during file errors

7️⃣ Employee Management Workflow

Add Employee

1. User enters employee details
2. ID uniqueness is validated
3. Employee object is created
4. Data is stored in collection and file

View Employees

- Displays all employees from memory

Update Salary

- Employee searched using HashMap
- Salary updated and saved

Delete Employee

- Employee removed from ArrayList & HashMap
- File updated

Reports

- Salary statistics
- Department-wise summary

8️⃣ Technical Details

Algorithms Used

- Linear search for reporting
- HashMap lookup for O(1) ID-based search

Data Structures

- `ArrayList<Employee>` for ordered storage
- `HashMap<String, Employee>` for fast lookup

Architecture

- Menu-driven console architecture
- Separation of concerns (Model, File, Reports, Main)

9️⃣ Visual Documentation

Screenshots

🔟 Testing Evidence

Sample Test Cases

Test Case	Input	Expected Output
Add Employee	Valid details	Employee added successfully
Duplicate ID	Existing ID	Error message shown
Update Salary	Valid ID	Salary updated
Delete Employee	Valid ID	Employee removed
Load File	Existing file	Data restored

Validation

- Input validation for numeric salary
- Duplicate employee prevention
- File existence check

1️⃣1️⃣ Quality Standards Checklist

✓ Project overview provided

✓ Clear goals & objectives

✓ Setup instructions included

✓ Organized code structure

✓ File handling explained

✓ Data format specified

✓ Workflow documented

✓ Technical details explained

✓ Testing evidence included

✓ Visual documentation guidelines

1️⃣2️⃣ Conclusion

This project successfully demonstrates the use of Java Collections, File Handling, and OOP principles in building a real-world application. It is suitable for academic evaluation and Java interview preparation.

Author: Sayali Shelke