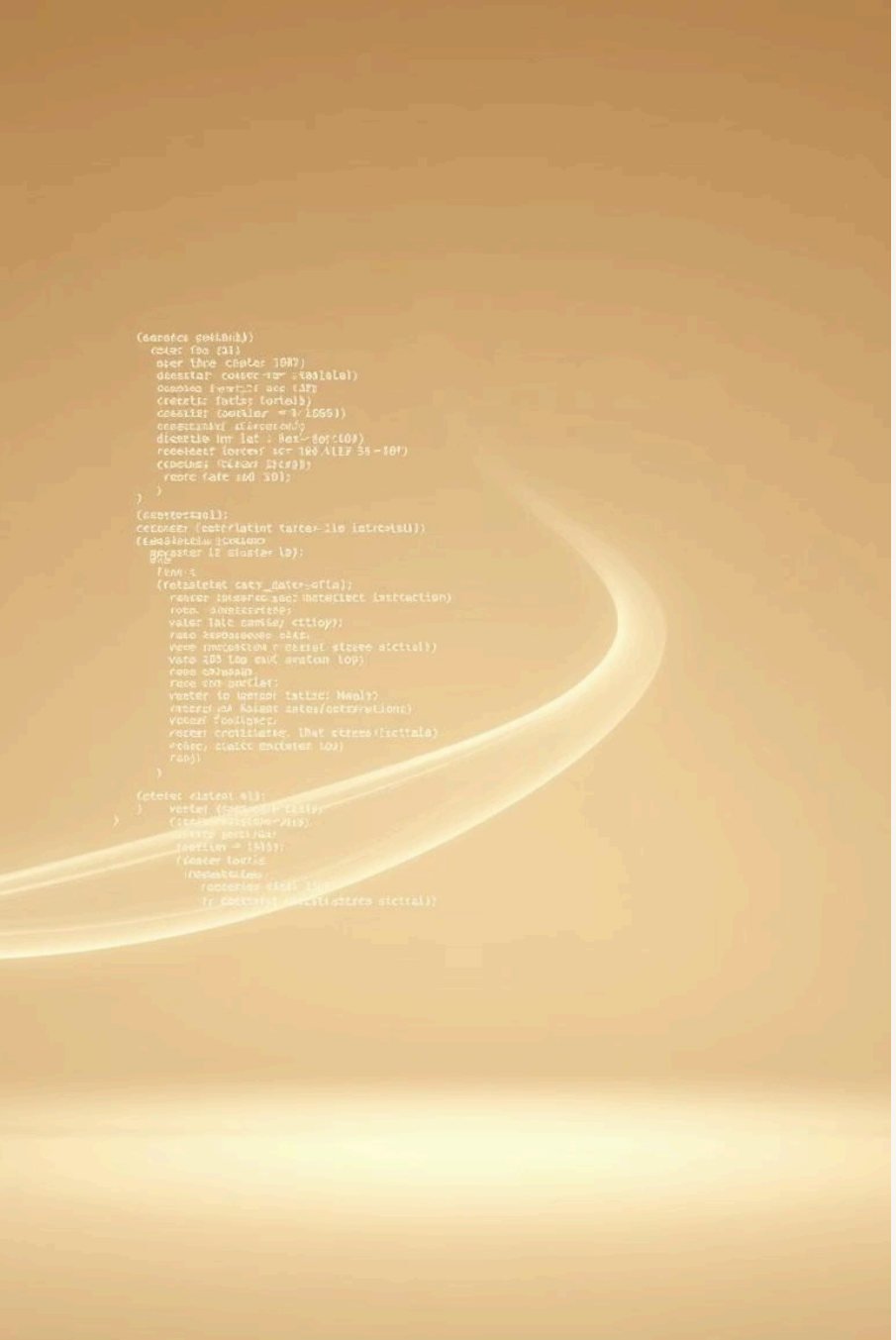

Employee Management System: Project Structure

This presentation outlines the project structure for a console-based Java Employee Management System. Our design emphasizes modularity and maintainability. It effectively demonstrates core Java principles.

AC **by Anshika choudhary**

I. Project Directory Structure

src/

Contains all source code.

- main/java/: Main application code.
- test/java/: Unit tests (JUnit).

lib/

External libraries (JARs).

resources/

Configuration and data files.

docs/

Project documentation.

build.gradle

Gradle build file.

README.md

Project overview.

II. Package Structure (src/main/java)

com.ems.model

Defines data representations, e.g., Employee.java.

com.ems.dao

Handles data access operations via interfaces and implementations.

com.ems.service

Manages business logic and data validation.

com.ems.ui

Manages console input/output.

com.ems.exception

Custom exception classes.

com.ems.config

Application configuration settings.

Main.java

Application entry point.

III. Model Layer (com.ems.model)

Employee.java

Represents an employee with various attributes.

- Attributes: ID, firstName, lastName, email, phoneNumber, hireDate, jobTitle, salary, department.
- Includes getters and setters.
- Overridden toString() for console output.

IV. Data Access Layer (com.ems.dao)

EmployeeDAO.java (Interface)

Defines methods for data operations.

- addEmployee
- getEmployeeById
- updateEmployee
- deleteEmployee
- getAllEmployees

EmployeeDAOImpl.java (Implementation)

Performs CRUD operations.

- Connects to database/file.
- Uses JDBC or file I/O.
- Includes error handling.

V. Service Layer (com.ems.service)

EmployeeService.java (Interface)

Defines business logic methods.

- createEmployee
- findEmployee
- modifyEmployee
- removeEmployee
- listAllEmployees

EmployeeServiceImpl.java (Implementation)

Implements business logic.

- Validates data input.
- Calls DAO methods.
- Handles exceptions and transactions.

VI. User Interface (com.ems.ui)

Presents Menu Options

Provides clear choices to the user.

Takes User Input

Utilizes the Scanner class.

Calls Service Layer

Executes actions based on input.

Displays Output

Presents results to the console.

VII. Main Class (Main.java)

Creates Instances

Initializes EmployeeService and ConsoleUI.

Starts Application

Invokes ConsoleUI.run().

Handles Lifecycle

Manages startup and graceful shutdown.

Key Takeaways & Next Steps

1

Modular Design

Ensures maintainability and scalability.

2

Layered Architecture

Separates concerns effectively.

3

Core Java Principles

Demonstrates fundamental concepts.

This project provides a robust foundation for console-based Java applications. Future enhancements could include database integration, a graphical user interface, and expanded employee functionalities.