EMPLOYEE MANAGEMENT SYSTEM

Mehul Natu, Natarajan Lekshmi Narayana Pillai, Sayeed Ahmed

AGENDA

Introduction

Operational Instruction

Design Pattern

Implementation

UML Diagram

Member Contribution

Knowledge Applied

Future Enhancements

INTRODUCTION

The Employee management application is aimed at maintaining records of the employees in a company. Some of its features include:

- A Login interface that enables access using username and password.
- A new user registration interface.
- Admin access to enable viewing and manage users.
- A search feature to search using various parameters.
- Teams interface to view and manage employees in individual teams.

OPERATIONAL INSTRUCTION

IDE	Apache Netbeans 12.5, Intellij 2022.2.1
JDK	JDK 18.1
Technologies	MySQL database, Swing UI
Design Pattern	Builder Design Pattern, Singleton Factory Design Pattern

DESIGN PATTERNS

Factory Design Pattern

The factory method design pattern is a creational design pattern that defines an interface for creating an object, but let's subclasses decide which class to instantiate.

It is a way to delegate the instantiation process to subclasses, which allows for greater flexibility and customization.

Builder Design Pattern

The builder design pattern is a design pattern that allows for the creation of complex objects with constituent parts that are created in the same order or using the same signature.

It is often used to create objects that have multiple optional parameters, or to create objects in a step-by-step fashion.

Singleton Design Pattern

Cultivate one-to-one customer service with robust ideas

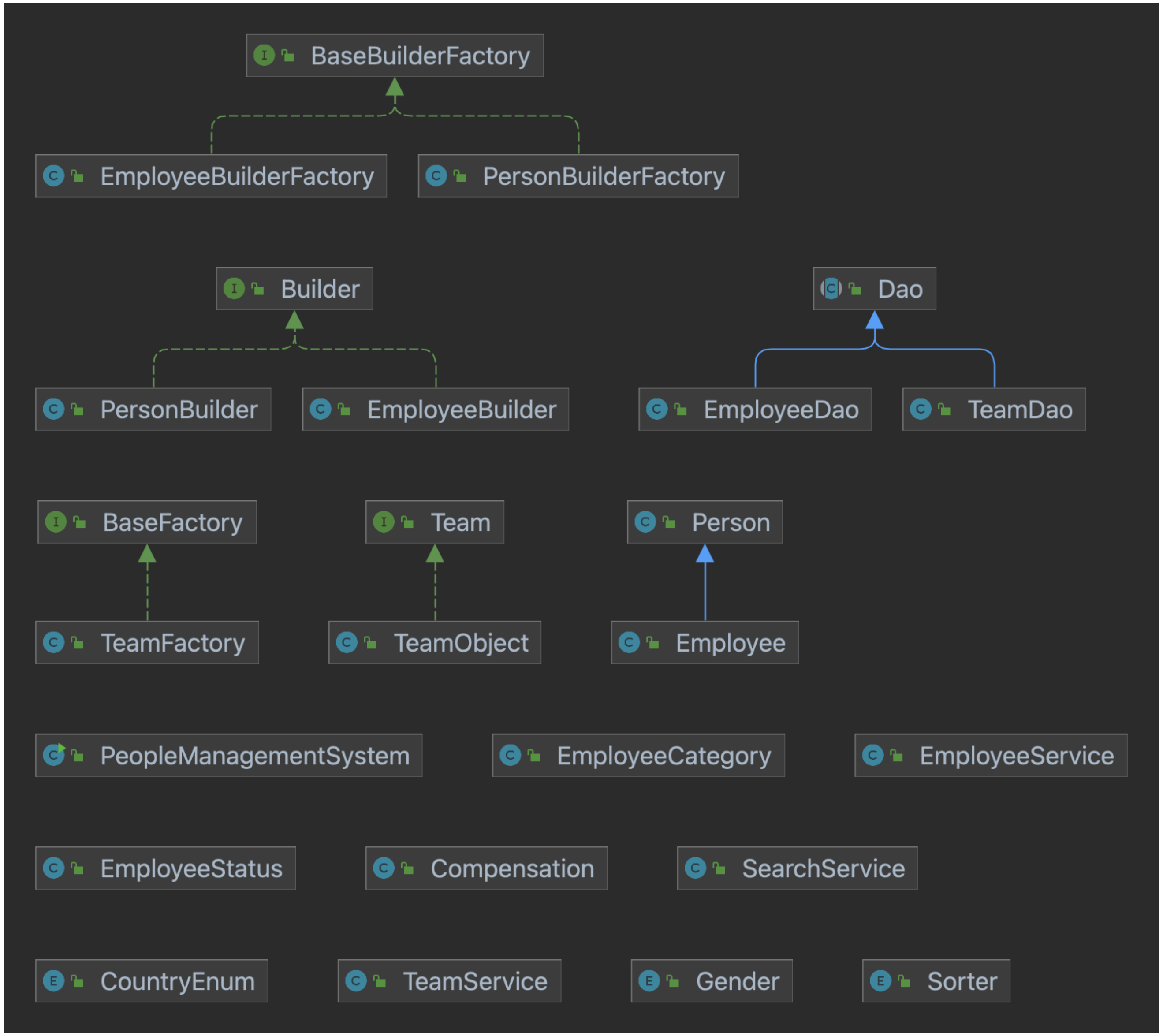
Maximize timely deliverables for real-time schemas

IMPLEMENTATION

First, we will start by explaining what DTO's did we have so majorly we had two DTOs which are **Employee** and **Team**. Since this project is about employee info management system with the possibilities of future functionalities, we need to keep a lot of user(Employee) data we wanted to get and set or build Employee objects easily which extends the Person DTO, a generic constructor will not fill this responsibilities easily as it will lower code readability and usability. We decided to go for Builder pattern just for the DTO's having numerous number of attribute fields, which would provide us with creating DTO's easily with no order for setting the attributes in-turn giving us flexibility when initializing its objects.

For developing DTOs which are flexible to build we used Factories which would provide us with different DTO builders. Thus, giving structure to Builders needed for complex Object creations. These Builder's are implementing a Builder Interface which is coupled with the Builder Factories. Thus, combining DTOs and their Builder, which are extending Person DTO.

UML DIAGRAM


BaseBuilderFactory
EmployeeBuilderFactory
PersonBuilderFactory
Builder
Dao
PersonBuilder
EmployeeBuilder
EmployeeDao
TeamDao
BaseFactory
Team
Person
TeamFactory
TeamObject
Employee
PeopleManagementSystem
EmployeeCategory
EmployeeService
EmployeeStatus
Compensation
SearchService
CountryEnum
TeamService
Gender
Sorter

KNOWLEDGE APPLIED

- **Swing**
- **Builder Design Pattern**
- **Factory Design Pattern**
- **Lambda Expression**
- **Enums**
- **Singleton Design Pattern**
- **MYSQL**
- **JDBC APIs**
- **Inheritance**
- **Dao Layer**
- **Exception Handling**

FUTURE ENHANCEMENTS

Future enhancements can use additional features like:

- Actively track and monitor holiday status of employees.
- Enable storing and retrieving HR documents.
- Create the functionality for Referrals.
- Create a talent pool for prospective employees.
- Track the open positions that is required to filled.
- Budgeting option.

THANK YOU

Mehul Natu - 002743870

Natarajan Lekshmi Narayana Pillai - 002766033

Sayeed Ahmed - 002191535

Git Link - https://github.com/Mehul-Natu/PEEPZ-Management-System/tree/final_branch