Documentação do Projeto Java: Gerenciamento de Funcionários

Este projeto tem como objetivo principal gerenciar uma lista de funcionários de uma indústria. A estrutura do código é orientada a objetos e utiliza recursos modernos da linguagem Java para garantir eficiência e boas práticas de programação.

Estrutura do Projeto

O projeto é composto por três classes principais:

1. **Pessoa**: A classe base.
2. **Funcionario**: Uma classe que herda de Pessoa.
3. **Principal**: A classe executável, onde a lógica de negócio é implementada.

---

1. Classe Pessoa

A classe Pessoa é a base para qualquer ser humano no nosso sistema. Ela contém os atributos mais básicos e essenciais.

- **Atributos:**
    - nome (String): O nome da pessoa.
    - dataNascimento (LocalDate): A data de nascimento da pessoa. Utilizamos LocalDate para representar a data de forma segura e moderna, evitando problemas de fusos horários e formatação que poderiam ocorrer com a antiga classe Date.
- **Métodos:**
    - Pessoa(String nome, LocalDate dataNascimento): Este é o **construtor** da classe. Ele é responsável por criar uma nova instância de Pessoa e inicializar os atributos nome e dataNascimento com os valores que você fornece.
    - getNome(): Método para obter o valor do atributo nome. É um método "getter".
    - getDataNascimento(): Método para obter o valor do atributo dataNascimento.
    - getDataNascimentoFormatada(): Este é um método auxiliar muito útil. Ele usa a classe DateTimeFormatter para converter a data de nascimento (LocalDate) para uma String no formato dd/MM/yyyy, como solicitado no projeto. Isso garante que a data seja sempre exibida da mesma forma, onde quer que seja usada.

---

2. Classe Funcionario

A classe Funcionario é um exemplo de **herança** em Programação Orientada a Objetos. Ela estende a classe Pessoa, o que significa que um objeto do tipo Funcionario já tem os atributos e métodos de Pessoa (nome e dataNascimento) sem precisar recriá-los.

- **Atributos Adicionais:**
    - salario (BigDecimal): O salário do funcionário. Usamos BigDecimal porque é a classe padrão do Java para lidar com valores monetários. Ela evita os erros de precisão que podem ocorrer com tipos como double ou float em cálculos financeiros.
    - funcao (String): A função ou cargo do funcionário na empresa.
- **Métodos:**
    - Funcionario(String nome, LocalDate dataNascimento, BigDecimal salario, String funcao): O construtor. Ele usa super() para chamar o construtor da classe Pessoa e inicializar os atributos herdados, antes de inicializar seus próprios atributos (salario e funcao).
    - getSalario(): Método "getter" para obter o valor do salário.
    - setSalario(BigDecimal salario): Método "setter" para atualizar o salário. Este método foi fundamental para implementar o aumento de 10% no salário.
    - getFuncao(): Método "getter" para obter a função.

---

3. Classe Principal

A classe Principal é onde o programa começa a ser executado. É nela que criamos as instâncias das outras classes e implementamos toda a lógica do projeto. A maioria dos requisitos do projeto é resolvida usando a API de **Streams** do Java, que oferece uma forma declarativa e concisa de processar coleções de dados.

- **Seções de Lógica (Dentro do método main):**
    - **3.1 - Inserção de Funcionários:** Criamos uma ArrayList de Funcionario e adicionamos cada funcionário com seus respectivos dados. A ArrayList é ideal pois mantém a ordem de inserção.
    - **3.2 - Remoção de Funcionário:** O método removeIf() é utilizado para remover o funcionário "João". Ele itera sobre a lista e remove o primeiro elemento que satisfaz a condição (nome igual a "Joao").
    - **3.3 - Impressão de Funcionários:** O método forEach() é usado para percorrer a lista de funcionários e imprimir as informações de cada um. A formatação do salário e da data é feita de forma elegante com o método auxiliar imprimirFuncionario e printf().
    - **3.4 - Aumento de Salário:** Usamos forEach() novamente para percorrer a lista e o método setSalario() para atualizar o salário de cada funcionário,

aplicando o aumento de 10%. A classe BigDecimal garante que o cálculo seja preciso.

- **3.5 e 3.6 - Agrupamento e Impressão por Função:** A API de Streams brilha aqui. collect(Collectors.groupingBy(Funcionario::getFuncao)) é uma operação poderosa que transforma a lista de funcionários em um Map, onde cada chave é uma função e o valor é uma lista de funcionários daquela função.
- **3.8 - Aniversariantes de Outubro e Dezembro:** O método filter() é usado para filtrar a lista, mantendo apenas os funcionários cujos meses de nascimento são 10 ou 12.
- **3.9 - Funcionário com Maior Idade:** O método min() é usado com um Comparator para encontrar o funcionário com a data de nascimento mais antiga (que é o mais velho). A idade é calculada usando a classe Period.
- **3.10 - Ordenação Alfabética:** O método sorted() com um Comparator ordena a lista de funcionários por nome.
- **3.11 - Total de Salários:** O método reduce() é usado para somar os salários de todos os funcionários da lista.
- **3.12 - Salários Mínimos:** Cada salário é dividido pelo valor do salário mínimo (BigDecimal("1212.00")) para mostrar quantos salários mínimos cada funcionário recebe.