

Exercício 1: Filtrar Clientes com Idade Acima de 30 Anos

Objetivo:

Neste exercício, você deve criar uma lista de objetos da classe Cliente e utilizar a **Stream API** do Java para filtrar apenas aqueles clientes cuja idade é maior que 30 anos. Depois, você deverá coletar esses resultados em uma nova lista e exibi-los.

Descrição das Classes

1. Interface Entidade

- A interface Entidade serve para garantir que todas as classes que a implementam possuam um método `getNome()`, que retorna o nome do objeto. Essa interface será implementada por diversas classes, como Cliente, Funcionario, etc.
- **Método:**
 - `String getNome()`: Retorna o nome do objeto.

Classe Abstrata Pessoa

- A classe Pessoa é uma classe abstrata que implementa a interface Entidade. Ela contém atributos comuns a todos os tipos de pessoa, como nome e idade.
- **Atributos:**
 - `String nome`: Nome da pessoa.
 - `int idade`: Idade da pessoa.
- **Métodos:**
 - O método `getNome()` é implementado aqui, pois a classe Pessoa é uma implementação da interface Entidade.

Classe Cliente

- A classe Cliente herda da classe Pessoa e, portanto, possui os atributos e métodos da classe Pessoa. Ela é especializada para representar um cliente em um sistema.
- **Atributos:** Herda de Pessoa (portanto, tem nome e idade).
- **Métodos:** Herda `getNome()` e `getIdade()`.

Descrição do Exercício

1. Criação da Lista de Clientes:

Você deve criar uma lista de objetos da classe Cliente. Cada cliente possui um nome e uma idade.

2. Filtragem da Lista com Stream API:

Utilize a Stream API para filtrar apenas os clientes cuja idade seja superior a 30 anos. Para isso, você pode usar o método `filter()`.

3. Coleta do Resultado:

Após a filtragem, use o método `collect()` para coletar os clientes que atendem à condição de idade em uma nova lista.

4. Exibição dos Resultados:

Finalmente, percorra a lista resultante e exiba o nome dos clientes que possuem idade superior a 30 anos.

Exercício 2: Filtrar Funcionários com Salário Maior que R\$ 5000 e Ordenar

Objetivo:

Neste exercício, você deve criar uma lista de objetos da classe Funcionario, filtrar os funcionários cujo salário seja superior a R\$ 5000 e, em seguida, ordenar esses funcionários do maior para o menor salário.

Descrição das Classes

1. Interface Entidade

- A interface Entidade garante que todas as classes que a implementam possuam o método `getNome()`, que retorna o nome do objeto.
- **Método:**
 - `String getNome():` Retorna o nome do objeto.

Classe Abstrata Pessoa

- A classe Pessoa é uma classe abstrata que implementa a interface Entidade e contém atributos comuns a todas as pessoas, como nome e idade.
- **Atributos:**
 - `String nome:` Nome da pessoa.
 - `int idade:` Idade da pessoa.
- **Métodos:**

- O método `getNome()` é implementado aqui, pois a classe `Pessoa` é uma implementação da interface `Entidade`.

Classe Funcionario

- A classe `Funcionario` herda de `Pessoa` e representa um funcionário, incluindo um salário.
- **Atributos:**
 - `String nome`: Nome do funcionário (herdado de `Pessoa`).
 - `int idade`: Idade do funcionário (herdado de `Pessoa`).
 - `double salario`: Salário do funcionário.
- **Métodos:**
 - O método `getNome()` e `getIdade()` são herdados de `Pessoa`.
 - Método adicional `getSalario()` para obter o salário do funcionário.

Descrição do Exercício

1. **Criação da Lista de Funcionários:**

Você deve criar uma lista de objetos da classe `Funcionario`. Cada funcionário possui um nome, idade e salário.
2. **Filtragem da Lista com Stream API:**

Utilize a Stream API para filtrar apenas os funcionários que possuem salário superior a R\$ 5000. Para isso, você pode usar o método `filter()`.
3. **Ordenação dos Funcionários:**

Após filtrar os funcionários com salário superior a R\$ 5000, você deve ordená-los em ordem decrescente de salário. Para isso, você pode usar o método `sorted()` com um comparador reverso.
4. **Coleta do Resultado:**

Após a filtragem e ordenação, use o método `collect()` para coletar os funcionários que atendem ao critério em uma nova lista.
5. **Exibição dos Resultados:**

Finalmente, percorra a lista resultante e exiba o nome e o salário dos funcionários que possuem salário superior a R\$ 5000.

Exercício 3: Filtrar Clientes com Mais de 5 Anos de Cadastro

Objetivo:

Neste exercício, você deve criar uma lista de objetos da classe `Cliente`, filtrar os

clientes que possuem mais de 5 anos de cadastro e exibir esses clientes com seu nome e tempo de cadastro.

Descrição das Classes

1. Interface Entidade

- A interface Entidade garante que todas as classes que a implementam possuam o método `getNome()`, que retorna o nome do objeto.
- **Método:**
 - `String getNome()`: Retorna o nome do objeto.

Classe Abstrata Pessoa

- A classe Pessoa é uma classe abstrata que implementa a interface Entidade e contém atributos comuns a todas as pessoas, como nome e idade.
- **Atributos:**
 - `String nome`: Nome da pessoa.
 - `int idade`: Idade da pessoa.
- **Métodos:**
 - O método `getNome()` é implementado aqui, pois a classe Pessoa é uma implementação da interface Entidade.

Classe Cliente

- A classe Cliente herda de Pessoa e representa um cliente, incluindo a data de cadastro.
- **Atributos:**
 - `String nome`: Nome do cliente (herdado de Pessoa).
 - `int idade`: Idade do cliente (herdado de Pessoa).
 - `LocalDate dataCadastro`: Data em que o cliente foi cadastrado.
- **Métodos:**
 - O método `getNome()` e `getIdade()` são herdados de Pessoa.
 - Método adicional `getDataCadastro()` para obter a data de cadastro do cliente.

- Método `getTempoCadastro()` que calcula o tempo de cadastro em anos.

Descrição do Exercício

1. Criação da Lista de Clientes:

Você deve criar uma lista de objetos da classe `Cliente`. Cada cliente possui um nome, idade e data de cadastro.

2. Filtragem da Lista com Stream API:

Utilize a Stream API para filtrar os clientes cujo tempo de cadastro seja superior a 5 anos. Para isso, você pode usar o método `filter()` e o método `getTempoCadastro()` da classe `Cliente`.

3. Exibição dos Resultados:

Após filtrar os clientes com mais de 5 anos de cadastro, você deve exibir o nome e o tempo de cadastro dos clientes.

Exercicio 4: Filtrar e Ordenar Produtos pelo Preço

Objetivo:

Neste exercício, você deve criar uma lista de objetos da classe `Produto`, filtrar os produtos que estão em estoque e, em seguida, ordenar os produtos pelo preço em ordem crescente.

Descrição das Classes

1. Interface Entidade

- A interface `Entidade` é a mesma utilizada no exercício anterior, com um método `getNome()` que retorna o nome do objeto.
- **Método:**
 - `String getNome():` Retorna o nome do objeto.

Classe Abstrata `ProdutoBase`

- A classe `ProdutoBase` é uma classe abstrata que implementa a interface `Entidade` e contém atributos comuns a todos os produtos, como nome e `quantidadeEmEstoque`.
- **Atributos:**
 - `String nome:` Nome do produto.
 - `int quantidadeEmEstoque:` Quantidade de unidades em estoque.
- **Métodos:**

- O método `getNome()` é implementado aqui, pois a classe `ProdutoBase` implementa a interface `Entidade`.

Classe Produto

- A classe `Produto` herda de `ProdutoBase` e representa um produto com um preço.
- **Atributos:**
 - `String nome`: Nome do produto (herdado de `ProdutoBase`).
 - `int quantidadeEmEstoque`: Quantidade de unidades em estoque (herdado de `ProdutoBase`).
 - `double preco`: Preço do produto.
- **Métodos:**
 - O método `getNome()` e `getQuantidadeEmEstoque()` são herdados de `ProdutoBase`.
 - Método adicional `getPreco()` que retorna o preço do produto.

Descrição do Exercício

1. **Criação da Lista de Produtos:**

Você deve criar uma lista de objetos da classe `Produto`. Cada produto possui um nome, quantidade em estoque e preço.
2. **Filtragem da Lista com Stream API:**

Utilize a Stream API para filtrar os produtos que estão em estoque (quantidade em estoque maior que 0). Para isso, você pode usar o método `filter()` e o método `estaEmEstoque()` da classe `Produto`.
3. **Ordenação da Lista com `sorted()`:**

Após filtrar os produtos, você deve ordenar a lista de produtos pelo preço de forma crescente utilizando o método `sorted()`.
4. **Exibição dos Resultados:**

Após filtrar e ordenar os produtos, exiba o nome, quantidade em estoque e preço de cada produto.

Exercício 5: Filtrar e Agrupar Funcionários por Departamento

Objetivo:

Neste exercício, você deve criar uma lista de objetos da classe `Funcionario`, filtrar os funcionários que possuem um salário dentro de uma faixa específica e, em seguida, agrupar os funcionários por departamento.

Descrição das Classes

1. Interface Entidade

- A interface Entidade será reutilizada dos exercícios anteriores. Ela possui um método `getNome()`, que retorna o nome do objeto.
- **Método:**
 - `String getNome()`: Retorna o nome do objeto.

Classe Abstrata Pessoa

- A classe Pessoa é abstrata e herda a interface Entidade. Ela contém os atributos e comportamentos comuns a todos os tipos de pessoas, como nome e idade.
- **Atributos:**
 - `String nome`: Nome da pessoa.
 - `int idade`: Idade da pessoa.
- **Métodos:**
 - `String getNome()`: Retorna o nome da pessoa.

Classe Funcionario

- A classe Funcionario herda de Pessoa e representa um funcionário com salário e departamento.
- **Atributos:**
 - `String nome`: Nome do funcionário (herdado de Pessoa).
 - `int idade`: Idade do funcionário (herdado de Pessoa).
 - `double salario`: Salário do funcionário.
 - `String departamento`: Departamento onde o funcionário trabalha.
- **Métodos:**
 - `getNome()`: Retorna o nome do funcionário (herdado de Pessoa).
 - `getSalario()`: Retorna o salário do funcionário.
 - `getDepartamento()`: Retorna o departamento do funcionário.

Descrição do Exercício

1. **Criação da Lista de Funcionários:**

Você deve criar uma lista de objetos da classe Funcionario. Cada funcionário possui um nome, idade, salário e departamento.

2. **Filtragem da Lista com Stream API:**

Utilize a Stream API para filtrar os funcionários que têm um salário dentro de uma faixa específica (por exemplo, salário entre 2000 e 5000).

3. **Agrupamento da Lista por Departamento:**

Depois de filtrar os funcionários, agrupe-os pelo departamento utilizando o método `Collectors.groupingBy()`. O resultado deve ser um mapa, onde as chaves são os departamentos e os valores são as listas de funcionários que pertencem a esses departamentos.

4. **Exibição dos Resultados:**

Após filtrar e agrupar os funcionários, exiba o nome e o salário de cada funcionário, agrupado pelo departamento.

Descrição do Exercício

1. **Criação da Lista de Funcionários:**

Você deve criar uma lista de objetos da classe Funcionario. Cada funcionário possui um nome, idade, salário e departamento.

2. **Filtragem da Lista com Stream API:**

Utilize a Stream API para filtrar os funcionários que têm um salário dentro de uma faixa específica (por exemplo, salário entre 2000 e 5000).

3. **Agrupamento da Lista por Departamento:**

Depois de filtrar os funcionários, agrupe-os pelo departamento utilizando o método `Collectors.groupingBy()`. O resultado deve ser um mapa, onde as chaves são os departamentos e os valores são as listas de funcionários que pertencem a esses departamentos.

4. **Exibição dos Resultados:**

Após filtrar e agrupar os funcionários, exiba o nome e o salário de cada funcionário, agrupado pelo departamento.