Expressão Lambda

As expressões lambda, introduzidas no Java 8, representam uma evolução significativa na linguagem, permitindo aos desenvolvedores adotar um estilo de programação mais funcional e conciso. Uma expressão lambda é uma maneira de representar um método ou comportamento como uma função anônima, ou seja, um bloco de código que pode ser passado como argumento para métodos, atribuído a variáveis ou retornado de métodos. Elas são especialmente úteis em contextos onde é necessário realizar operações sobre coleções, como em operações de filtragem, mapeamento e iteração, com uma sintaxe muito mais limpa e expressiva em comparação com as abordagens tradicionais.

Uma expressão lambda segue a seguinte estrutura básica:

- **Lista de parâmetros**: Uma lista de parâmetros entre parênteses, que pode ser opcionalmente omitida quando houver um único parâmetro.
- **Operador "->"**: Utilizado para separar os parâmetros do corpo da expressão lambda.
- **Corpo da expressão**: Pode ser uma única linha de código ou um bloco de código com múltiplas instruções.

Exemplo: (int a, int b) -> a + b

Antes da introdução das expressões lambda, a principal forma de passar comportamento como argumento ou implementá-lo inline no Java era utilizando **classes anônimas**. Uma classe anônima é uma classe sem nome, geralmente utilizada para criar instâncias de interfaces ou classes abstratas. Embora funcional, o uso de classes anônimas exige uma estrutura mais verbosa, incluindo a palavra-chave new e a implementação completa dos métodos.

As expressões lambda oferecem diversas vantagens, principalmente quando usadas em operações funcionais em coleções, como em streams e iteração sobre listas. Entre as principais vantagens, destacam-se:

1. **Concorrência e clareza**: O uso de lambdas reduz a complexidade do código, tornando-o mais legível e direto. Isso facilita a manutenção e compreensão do fluxo do programa.

2. **Funcionalidade em Coleções e Streams**: Lambdas são amplamente usadas em operações como filtragem, mapeamento e transformação de dados, especialmente em conjunto com a API de Streams, permitindo operações de alto nível em coleções de forma expressiva e eficiente.

3. **Integração com interfaces funcionais**: Lambdas são compatíveis com **interfaces funcionais**, ou seja, interfaces que possuem apenas um método abstrato, como Runnable, Comparator, Function, entre outras. Essa característica as torna especialmente úteis para implementação de callbacks e para usar bibliotecas modernas de Java que se beneficiam da programação funcional.

As expressões lambda no Java representam um avanço significativo para a linguagem, proporcionando uma forma mais moderna e funcional de manipulação de dados e comportamentos. Sua introdução não só simplificou o código, mas também habilitou a adoção de padrões de programação funcional no Java. Comparadas às classes anônimas, as Lambdas são mais concisas, expressivas e eficientes, contribuindo para a legibilidade e manutenção do código. Em conjunto com as referências de método e a API de Streams, as expressões lambda oferecem aos desenvolvedores um poderoso conjunto de ferramentas para criar aplicações mais limpas, eficientes e modernas.