06

Diferença entre ArrayList e LinkedList

E o mistério da `LinkedList`? E se tivéssemos usado `ArrayList` na declaração do atributo `aulas` da classe `Curso`? O resultado seria exatamente o mesmo!

Então qual é a diferença? Basicamente performance. A `ArrayList`, como diz o nome, internamente usa uma array para guardar os elementos. Ela consegue fazer umas operações de maneira muito eficiente, como invocar o método `get(indice)`. Se você precisa pegar o décimo quinto elemento, ele te devolve isso bem rápido. Onde uma `ArrayList` é lenta? Quando você for, por exemplo, inserir um novo elemento na primeira posição. Pois a implementação vai precisar mover todos os elementos que estão no começo da lista para a próxima posição. Se há muitos elementos, isso vai demorar... chamamos isso em computação de consumo de tempo linear.

Já a `LinkedList` possui uma grande vantagem aqui. Ela utiliza a estrutura de dados chamada lista ligada. Ela é muito rápida para adicionar e remover elementos na *cabeça* da lista, isso é, na primeira posição. Mas ela é lenta se você precisar acessar um determinado elemento, pois a implementação precisará percorrer todos os elementos até chegar ao décimo quinto, por exemplo.

Confuso? Não tem problema. Sabe o que é interessante? Você não precisa tomar essa decisão desde já e oficializar para sempre. Como utilizamos a referência a `List`, comprometendo-nos pouco, podemos *sempre* mudar a implementação, isso é, em quem damos `new`, caso percebamos que é melhor uma ou outra lista nesse caso em particular.

Ainda está confuso? Veja código abaixo que testa a inserção de 1 milhão de elementos com `ArrayList` e `LinkedList`, medindo o tempo. Além disso, estamos removendo 100 elementos, sempre tirando do início da lista. Execute e veja a diferença:

```
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;

public class TesteListas {

    public static void main(String[] args) {

        System.out.println("**** ArrayList vs LinkedList ***");

        List<Integer> numerosArrayList = new ArrayList<>();
        List<Integer> numerosLinkedList = new LinkedList<>();
        int quantidadeElementos = 1000000;

        long tempoArrayList  = insereElementosNo(numerosArrayList, quantidadeElementos);
        long tempoLinkedList = insereElementosNo(numerosLinkedList, quantidadeElementos);

        System.out.println("Inserção na ArrayList demorou  " + tempoArrayList);
        System.out.println("Inserção na LinkedList demorou " + tempoLinkedList);

        tempoArrayList = removePrimeirosElementos(numerosArrayList);
        tempoLinkedList = removePrimeirosElementos(numerosLinkedList);

        System.out.println("Remoção da ArrayList demorou  " + tempoArrayList);
        System.out.println("Remoção da LinkedList demorou " + tempoLinkedList);
    }
```

```
    /*
     * removendo 100 elementos sempre na primeira posição
     */
    private static long removePrimeirosElementos(List<Integer> numeros) {
        long ini = System.currentTimeMillis();

        for (int i = 0; i < 100; i++) {
            numeros.remove(0); //removendo sempre o primeiro elemento
        }
        long fim = System.currentTimeMillis();
        return fim-ini;
    }

    private static long insereElementosNo(List<Integer> numeros, int quantidade) {
        long ini = System.currentTimeMillis();
        for (int i = 0; i < quantidade; i++) {
            numeros.add(i);
        }
        long fim = System.currentTimeMillis();
        return fim-ini;
    }

}
```