Listas simples - Subturma A

Arquivo 1: No.java

Java

```java
// A classe que representa um único elemento da lista.
public class No {
    int dado;
    No proximo; // A "seta" que aponta para o próximo nó.

    // Construtor para criar um novo nó.
    public No(int dado) {
        this.dado = dado;
        this.proximo = null; // Quando um nó é criado, ele ainda não aponta para ninguém.
    }
}
```

Arquivo 2: ListaEncadeada.java

```java
// A classe que gerencia todos os nós e a lógica da lista.
public class ListaEncadeada {
    No inicio; // O ponto de partida da nossa lista. Equivalente ao "head".

    // Método já ensinado
    public void adicionarNoInicio(int dado) {
        No novoNo = new No(dado);
        novoNo.proximo = inicio;
        inicio = novoNo;
    }

    // Método já ensinado
    public void imprimirLista() {
        System.out.print("Lista: [ ");
        No atual = this.inicio; // Começamos a percorrer pelo início.
        while (atual != null) {
            System.out.print(atual.dado + " -> ");
            atual = atual.proximo; // Pulamos para o próximo nó.
        }
        System.out.println("NULL ]");
    }

    // ---------- ATIVIDADE PRÁTICA COMEÇA AQUI ---------- //

    /*
     * OBJETIVO 1: Contar os elementos
```

```
     * Implemente um método que percorra a lista e retorne
     * o número total de nós.
     */
    public int tamanho() {
        // Seu código vem aqui
        // Dica: Você vai precisar de uma variável contadora e de um laço `while`,
        // muito parecido com o do método `imprimirLista`.
        return 0; // Apague esta linha e substitua pelo seu código.
    }

    /*
     * OBJETIVO 2: Adicionar um elemento no final
     * Implemente um método que adicione um novo nó no FINAL da lista.
     */
    public void adicionarNoFinal(int dado) {
        No novoNo = new No(dado);

        // Caso especial: a lista está vazia?
        // Se `inicio` for nulo, o que você deve fazer?
        if (inicio == null) {
            // Seu código para o caso de lista vazia vem aqui.
        } else {
            // Se a lista não estiver vazia, você precisa encontrar o ÚLTIMO nó.
            // O último nó é aquele cujo `proximo` é `null`.
            No ultimo = this.inicio;

            // Crie um laço `while` para caminhar até o último nó.
            while (ultimo.proximo != null) {
                // Seu código para avançar na lista vem aqui.
            }

            // Quando o laço terminar, a variável `ultimo` guardará o último nó.
            // Agora, faça o `proximo` dele apontar para o `novoNo`.
        }
    }
}
```

Arquivo 3: `Principal.java` (para testar)

```
public class Principal {
    public static void main(String[] args) {
        ListaEncadeada lista = new ListaEncadeada();
        lista.adicionarNoInicio(10);
        lista.adicionarNoInicio(20);
        lista.adicionarNoInicio(30);
        lista.imprimirLista(); // Deve imprimir: Lista: [ 30 -> 20 -> 10 -> NULL ]
```

```
        System.out.println("--- Testando a Atividade ---");

        // Testando o Objetivo 1
        System.out.println("Tamanho da lista: " + lista.tamanho()); // Deve imprimir 3

        // Testando o Objetivo 2
        lista.adicionarNoFinal(5);
        lista.imprimirLista(); // Deve imprimir: Lista: [ 30 -> 20 -> 10 -> 5 -> NULL ]
        System.out.println("Novo tamanho da lista: " + lista.tamanho()); // Deve imprimir 4

        // Teste extra para o addLast
        ListaEncadeada listaVazia = new ListaEncadeada();
        listaVazia.adicionarNoFinal(100);
        System.out.print("Testando addLast em lista vazia: ");
        listaVazia.imprimirLista(); // Deve imprimir: Lista: [ 100 -> NULL ]
    }
}
```