Código 3 - Estruturas de dados genéricas personalizadas e lista encadeada

Resumo: O código apresentado exemplifica o uso de conceitos fundamentais de Java, como generics, encapsulamento, orientação a objetos, e manipulação de listas encadeadas. Ele demonstra a criação de uma estrutura de dados genérica representada pela classe No, que armazena dados e referências para outros nós, formando uma lista encadeada. A classe Lista oferece uma implementação de uma lista encadeada, com métodos para adicionar, remover e imprimir elementos da lista, permitindo a manipulação flexível dos dados. A classe Principal ilustra o uso dessas estruturas, mostrando como criar, modificar e percorrer a lista. As principais referências utilizadas para elaborar este material incluem o livro "Java: Como Programar" (Deitel) para os conceitos de orientação a objetos, encapsulamento, e generics, além da documentação oficial da linguagem Java para detalhes técnicos específicos.

Será detalhado abaixo cada parte das classes ***No*** e ***Principal***, explicando os conceitos usados nos códigos.

Classe “No”

```
class No<T>{
    private T dado;
    private No<T> nextNo;
```

Essa classe define um nó **genérico No<T>** para uma estrutura de dados. O **T** é um parâmetro de tipo genérico que permite que a classe funcione com qualquer tipo de dado. O atributo **dado** armazena o valor do nó, e **nextNo** é uma referência ao próximo nó na lista.

```
public No(T dado){
    this(dado, null);
}
```

Esse construtor cria um novo nó inicializando o **dado** com o valor fornecido e define **nextNo** como **null**, indicando que o nó não está vinculado a outro nó.

```
public No(T dado, No<T> no){
    this.dado = dado;
    this.nextNo = no;
}
```

Esse construtor permite criar um nó e ao mesmo tempo especificar a referência para o próximo nó (**nextNo**). Se **no** for **null**, o nó será o último da lista.

```
public void setDado(T dado){
    this.dado = dado;
}

public T getDado(){
    return this.dado;
}
```

Esses são os métodos de acesso (**getters** e **setters**) para o atributo **dado**. Eles permitem ler e modificar o valor armazenado no nó.

```
public void setNextNo(No<T> nextNo){
    this.nextNo = nextNo;
}

public No<T> getNextNode(){
    return this.nextNo;
}
```

Esses métodos são usados para definir e acessar o próximo nó na lista. **setNextNo** permite vincular o nó atual a outro nó, e **getNextNode** retorna a referência ao próximo nó.

```
@Override
public String toString(){
    return "Dado{ " + getDado() + " }";
}
```

O método **toString** foi sobrescrito para fornecer uma representação em string do objeto **No**. Ele retorna uma **string** que exibe o valor armazenado em **dado**.

Classe "Lista"

```
public class Lista<T>{
    private No<T> primeiroNo;
    private No<T> ultimoNo;
    private String nomeLista;
```

A classe **Lista** é uma implementação genérica de uma lista encadeada, onde **T** representa o tipo de dados que a lista armazenará. Os atributos **primeiroNo** e **ultimoNo** são referências ao primeiro e ao último nó da lista, respectivamente, enquanto **nomeLista** armazena o nome descritivo da lista.

```
public Lista(){
    this("Lista");
}
```

Esse é o construtor padrão que inicializa uma lista com o nome padrão "**Lista**". Ele chama outro construtor da classe, passando o nome "**Lista**" como argumento.

```
public Lista(String nomeLista){
    this.nomeLista = nomeLista;
    this.primeiroNo = null;
    this.ultimoNo = null;
}
```

Esse construtor permite criar uma lista com um nome específico, definido pelo usuário. Inicialmente, a lista está vazia, então **primeiroNo** e **ultimoNo** são definidos como null.

```
public void addInicio(T dado){
    No<T> novoNo = new No<T>(dado);
    if(primeiroNo == null){
        primeiroNo = ultimoNo = novoNo;
    }else{
        novoNo.setNextNo(primeiroNo);
        primeiroNo = novoNo;
    }
}
```

O método **addInicio** insere um novo nó no início da lista. Se a lista estiver vazia (**primeiroNo** é **null**), o novo nó se torna tanto o **primeiro** quanto o **último** nó da lista. Caso contrário, o novo nó é vinculado ao nó que era o **primeiro**, e então se torna o novo **primeiroNo**.

```
public void addFinal(T dado){
    No<T> novoNo = new No<T>(dado);
    if(ultimoNo == null){
        primeiroNo = ultimoNo = novoNo;
    }else{
        ultimoNo.setNextNo(novoNo);
        ultimoNo = novoNo;
    }
}
```

O método **addFinal** insere um novo nó no **final** da lista. Se a lista estiver vazia (**ultimoNo** é **null**), o novo nó se torna tanto o **primeiro** quanto o **último**. Caso contrário, o novo nó é adicionado após o último nó, e então se torna o novo **ultimoNo**.

```
public void removeInicio(){
    if(primeiroNo == null){
        System.out.println("Lista Vazia");
    }else{
        System.out.println("Dado: " + primeiroNo.getDado() + " removido da
lista.");
        primeiroNo = primeiroNo.getNextNode();
    }
}
```

O método **removeInicio** remove o primeiro nó da lista. Se a lista estiver vazia, ele exibe uma mensagem indicando que a lista está vazia. Caso contrário, o nó atual **primeiroNo** é removido, e o próximo nó na sequência se torna o novo **primeiroNo**.

```
public void removeFinal(){
    if(primeiroNo == null){
        System.out.println("Lista Vazia");
    }else{
        System.out.println("Dado: " + ultimoNo.getDado() + " removido da
lista.");

        No<T> aux = primeiroNo;

        while(aux.getNextNode() != ultimoNo){
            aux = aux.getNextNode();
        }
        ultimoNo = aux;
        aux.setNextNo(null);
    }
}
```

O método **removeFinal** remove o último nó da lista. Se a lista estiver vazia, ele exibe uma mensagem indicando que a lista está vazia. Caso contrário, ele percorre a lista para encontrar o **penúltimo** nó, define este nó como o novo **ultimoNo** e desvincula o último nó da lista.

```
public void imprimeLista(){
    if(primeiroNo == null){
        System.out.println("Lista Vazia");
    }else{
        System.out.printf("Dados da lista de %s:\n", nomeLista);

        No<T> aux = primeiroNo;

        while(aux != null){
            System.out.printf("- %s\n", aux.getDado());
            aux = aux.getNextNode();
        }
    }
}
```

O método **imprimeLista** percorre a lista e imprime todos os dados armazenados. Se a lista estiver vazia, uma mensagem indicando isso é exibida. Caso contrário, ele percorre cada nó da lista, imprimindo o dado armazenado em cada um.

Classe “Principal”

```
public class Principal{
    public static void main(String[] args){
```

A classe **Principal** contém o método **main**, que é o ponto de entrada do programa.

```
Lista<String> lista = new Lista<String>("Alunos");
```

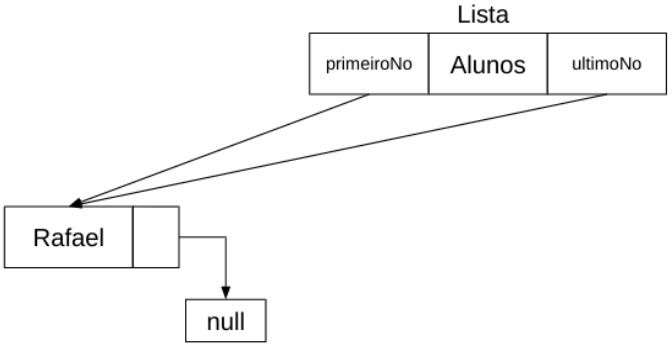
Essa linha cria uma nova instância da classe **Lista**, destinada a armazenar objetos do tipo **String**, e a nomeia como "**Alunos**".


```
lista.imprimeLista();
```

Essa linha chama o método **imprimeLista** do objeto **lista** da classe **Lista**. Esse método verifica se a lista está vazia. Se estiver, ele imprime "Lista Vazia". Caso contrário, ele percorre todos os nós da lista e imprime os dados armazenados em cada nó. Como essa é a primeira chamada a **imprimeLista**, e nenhum elemento foi adicionado ainda, a saída será "Lista Vazia".

```
lista.addInicio("Rafael");
```

O método **addInicio** é chamado com o argumento "Rafael". Esse método cria um novo nó contendo o **dado** "**Rafael**" e o insere no início da lista. Se a lista estava vazia, esse nó se torna o primeiro e o último nó da lista, como ilustrado na Figura abaixo.

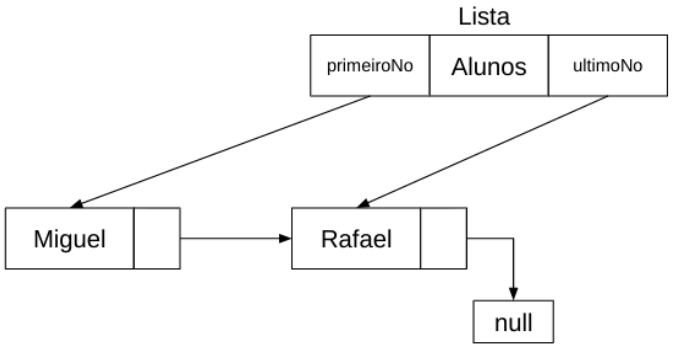


```
lista.imprimeLista();
```

Após adicionar "**Rafael**" à lista, o método **imprimeLista** é chamado novamente. Dessa vez, o método vai percorrer a lista e imprimir apenas os dados do nó "**Rafael**", que foi adicionado no início da lista.

```
lista.addInicio("Miguel");
```

Essa linha adiciona o dado "Miguel" no início da lista, assim como foi feito com "Rafael". Agora, "Miguel" se torna o primeiro nó da lista, e o nó que contém "Rafael" se torna o segundo.

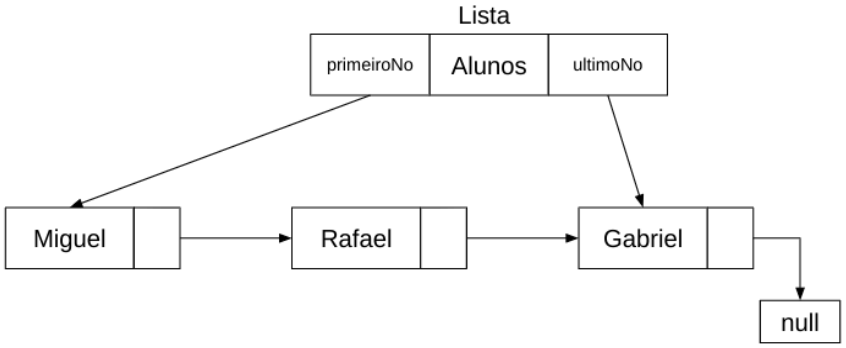



```
lista.imprimeLista();
```

Após adicionar "**Miguel**", a lista contém dois elementos. Ao chamar **imprimeLista**, o método imprime os dados dos nós em ordem: "**Miguel**" (primeiro) e "**Rafael**" (segundo).

```
lista.addFinal("Gabriel");
```

Nessa linha, o método **addFinal** é chamado com o parâmetro "**Gabriel**". Esse método adiciona um novo nó contendo "**Gabriel**" no final da lista. Se a lista estivesse vazia (o que não é o caso aqui), "**Gabriel**" se tornaria tanto o primeiro quanto o último nó. Como a lista já tem dois elementos, "**Gabriel**" se torna o **novo** último nó, e o nó anterior que continha "Rafael" se torna o penúltimo.

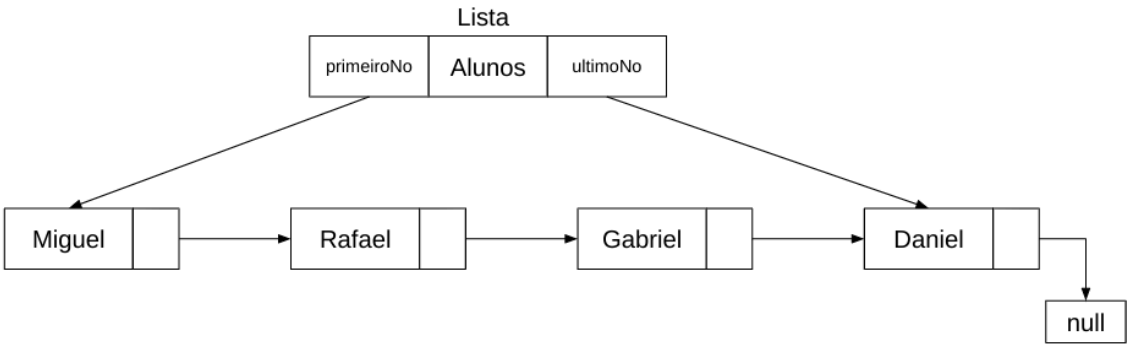



```
lista.imprimeLista();
```

Agora, ao chamar **imprimeLista**, a lista contém três elementos: "**Miguel**", "**Rafael**", e "**Gabriel**", nessa ordem. O método imprime os dados dos três nós.

```
lista.addFinal("Daniel");
```

Essa linha adiciona o dado "**Daniel**" ao final da lista, da mesma forma que foi feito com "**Gabriel**". Agora, "Daniel" se torna o último nó da lista, como ilustrado na Figura abaixo.

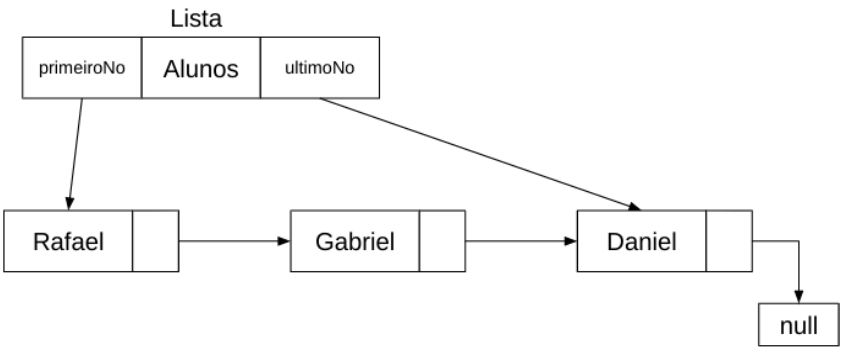



```
lista.imprimeLista();
```

Com quatro elementos na lista, a chamada ao método **imprimeLista** imprime os dados dos nós em ordem: "**Miguel**", "**Rafael**", "**Gabriel**", e "**Daniel**".

```
lista.removeInicio();
```

Nessa linha, o método **removeInicio** é chamado para remover o primeiro nó da lista. Se a lista estiver vazia (o que não é o caso), uma mensagem indicando que a lista está vazia seria exibida. Como a lista contém elementos, o método remove o nó que contém "**Miguel**" e o próximo nó (que contém "**Rafael**") se torna o novo primeiro nó.

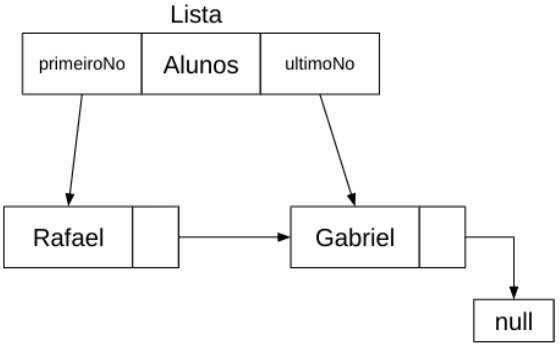



```
lista.imprimeLista();
```

Após a remoção de "**Miguel**", a lista agora contém três elementos: "**Rafael**", "**Gabriel**", e "**Daniel**". O método **imprimeLista** imprime os dados desses três nós.

```
lista.removeFinal();
```

Essa linha chama o método **removeFinal**, que remove o último nó da lista. Se a lista estiver vazia, uma mensagem de "Lista Vazia" é exibida. Como a lista tem elementos, o método remove o nó que contém "**Daniel**", e o penúltimo nó (que contém "**Gabriel**") se torna o novo último nó.




```
lista.imprimeLista();
```

Após a remoção de "**Daniel**", a lista contém dois elementos: "Rafael" e "Gabriel". O método **imprimeLista** imprime os dados desses dois nós.