Código 5 - 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 filas usando listas encadeadas. A classe No serve como base para criar uma estrutura de dados genérica que armazena informações e referências para outros nós, formando a base de uma fila encadeada. A classe Fila implementa uma fila genérica, com operações fundamentais como enfileirar (enqueue), desenfileirar (dequeue) e verificar se a fila está vazia. A classe Principal ilustra o uso da fila, incluindo a inserção de elementos no final, a remoção de elementos do início, e a exibição dos elementos na fila. O material foi elaborado com base no 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 "{ " + 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 "Fila"

```
public class Fila<T>{
    private No<T> primeiroNo;
    private No<T> ultimoNo;
    private String nomeFila;
```

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

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

Este é o construtor padrão da classe **Fila**, que chama o outro construtor com uma string vazia como argumento. Isso inicializa a fila com um nome vazio, e tanto o **primeiroNo** quanto o **ultimoNo** são definidos como null, indicando que a fila está inicialmente vazia.

```
public Fila(String nomeFila){
    this.nomeFila = nomeFila;
    this.primeiroNo = null;
    this.ultimoNo = null;
}
```

Este construtor permite a criação de uma fila com um nome específico. Ele define o nome da fila (**nomeFila**) e inicializa o **primeiroNo** e **ultimoNo** como **null**, indicando que a fila está vazia no momento de sua criação.

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

```
    }
}
```

O método **enfileirar** (**enqueue**) adiciona um novo elemento ao final da fila. Se a fila estiver vazia (ou seja, **ultimoNo** é **null**), o novo nó se torna tanto o primeiro quanto o último nó. Caso contrário, o novo nó é adicionado após o último nó atual, e **ultimoNo** é atualizado para apontar para esse novo nó.

```
public T desenfileirar(){
    if(primeiroNo == null){
        System.out.println("Fila Vazia");
        return null;
    }
    T dadoTemp = primeiroNo.getDado();
    primeiroNo = primeiroNo.getNextNo();
    if(primeiroNo == null){
        ultimoNo = null;
    }
    return dadoTemp;
}
```

O método **desenfileirar** (**dequeue**) remove e retorna o elemento no início da fila. Se a fila estiver vazia, ele exibe uma mensagem e retorna **null**. Caso contrário, ele salva o dado do primeiro nó, atualiza **primeiroNo** para o próximo nó na fila, e, se a fila estiver vazia após a remoção, também redefine **ultimoNo** para **null**.

```
public boolean isEmpty() {
    return primeiroNo == null;
}
```

O método **isEmpty** verifica se a fila está vazia. Ele retorna **true** se **primeiroNo** for **null**, indicando que não há elementos na fila, e **false** caso contrário.

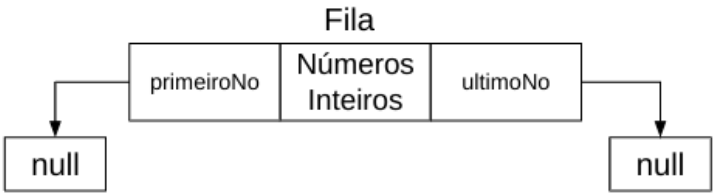
```
public void imprimeFila(){
    if(primeiroNo == null){
        System.out.println("Fila Vazia");
    }else{
        System.out.printf("Dados da fila %s:\n",nomeFila);
        No<T> aux = primeiroNo;
        while (aux != null) {
            System.out.printf("{ %s } ", aux.toString());
            aux = aux.getNextNo();
        }
        System.out.println();
    }
}
```

O método **imprimeFila** exibe os elementos da fila, começando pelo primeiro e seguindo até o último. Se a fila estiver vazia, ele exibe "Fila Vazia". Caso contrário, percorre os nós da fila, exibindo os dados armazenados em cada nó.

Classe “Principal”

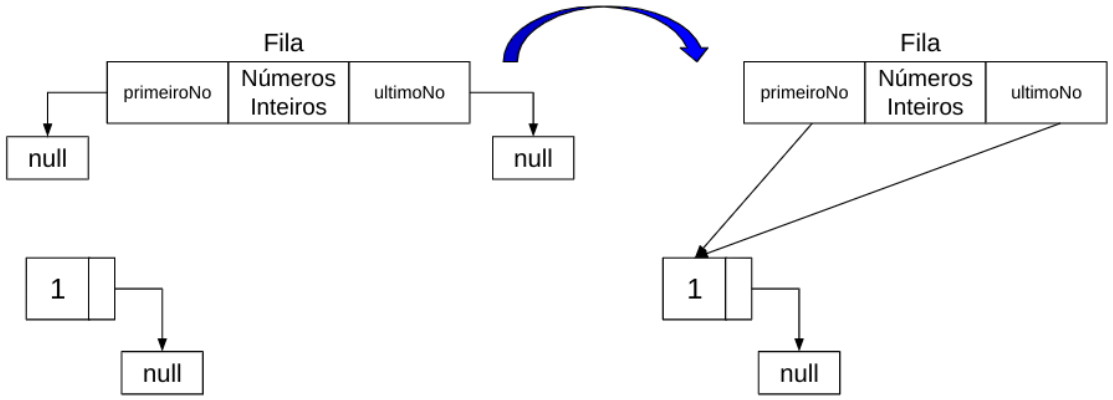
```
public class Principal{
    public static void main(String[] args){
        System.out.println("=== Testando a Fila ===");
        Fila<Integer> fila = new Fila<Integer>("Números Inteiros");
```

O método **main** é o ponto de entrada do programa. Ele cria uma nova **fila** de **Integer** chamada "Números Inteiros" e imprime uma mensagem indicando que a fila está sendo testada.



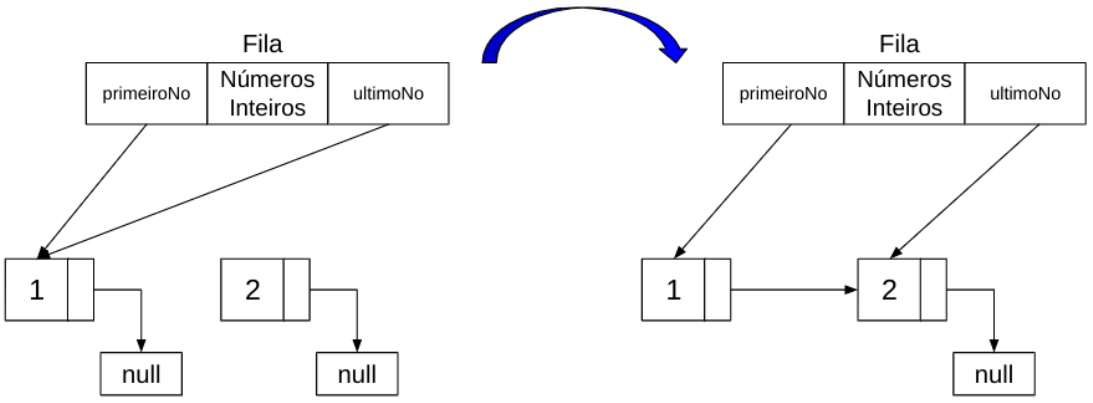

```
fila.enfileirar(1);
```

O método **enfileirar** é chamado com o argumento **(1)**. Esse método cria um novo nó contendo o dado inteiro **(1)** e o insere no início da fila. Se a fila estava vazia, esse nó se torna o primeiro nó e também o último da fila, como ilustrado na Figura abaixo.



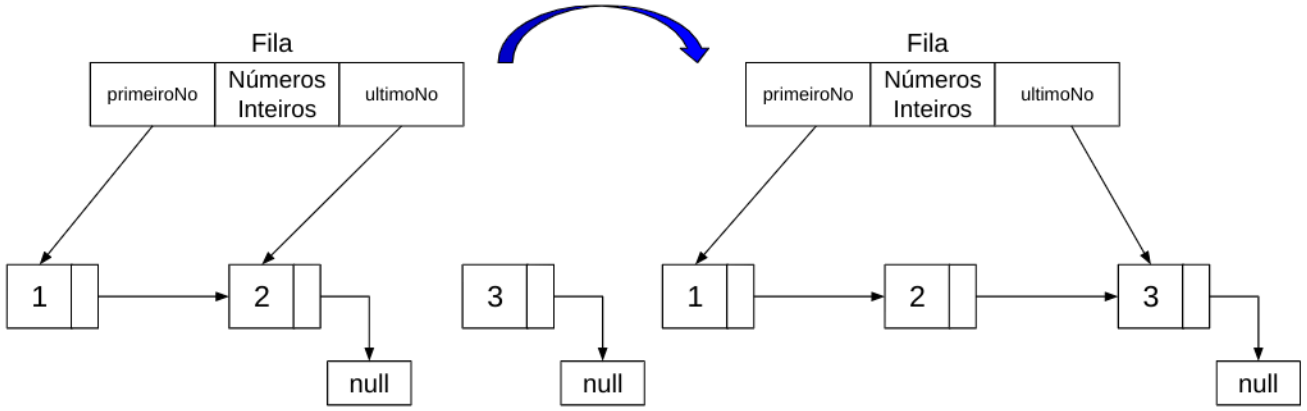

```
fila.enfileirar(2);
```

Essa linha adiciona o dado **(2)** na última posição da fila, assim como foi feito com **(1)**. Agora, **(2)** se torna o último nó da fila, e o nó que contém **(1)** permanece como primeiro da fila.



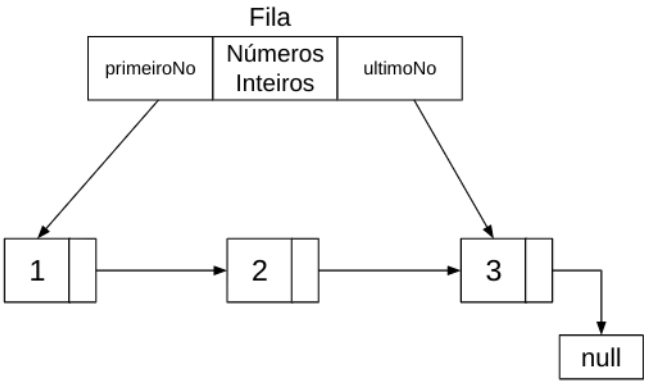

```
fila.enfileirar(3);
```

Essa linha adiciona o dado **(3)** na última posição da fila, assim como foi feito com **(1)** e o **(2)**. Agora, **(3)** se torna o último nó da fila, e o nó que contém **(1)** permanece como primeiro da fila.



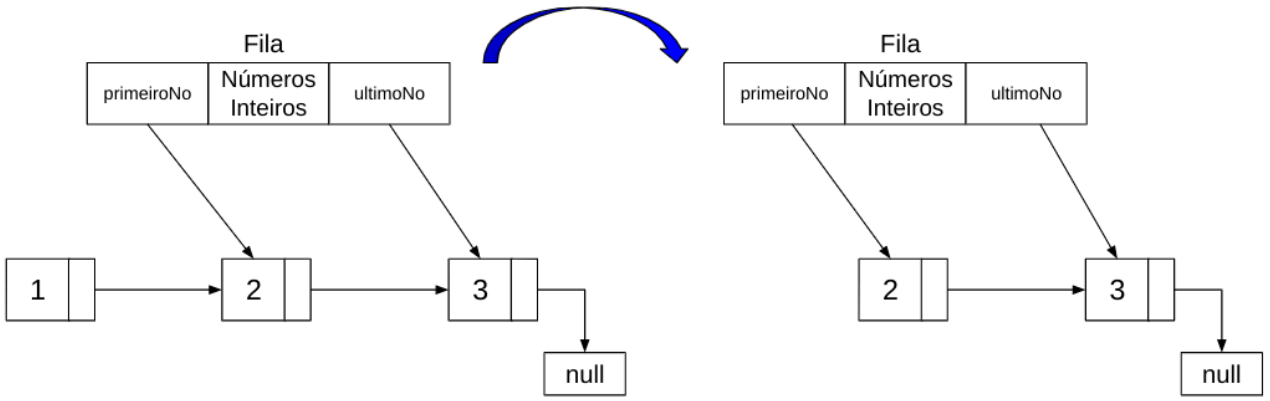

```
fila.imprimeFila();
```

Essa linha exibe os elementos da fila através do método **imprimeFila**, mostrando **(1)** na frente, seguido por **(2)** e **(3)**.




```
System.out.println("Elemento removido: " + fila.desenfileirar());
```

O método **desenfileirar** é chamado para remover o elemento no início da fila **(1)**, e o valor removido é impresso.




```
fila.imprimeFila();
```

Em seguida, **imprimeFila** exibe os elementos restantes na fila **(2 e 3)**.