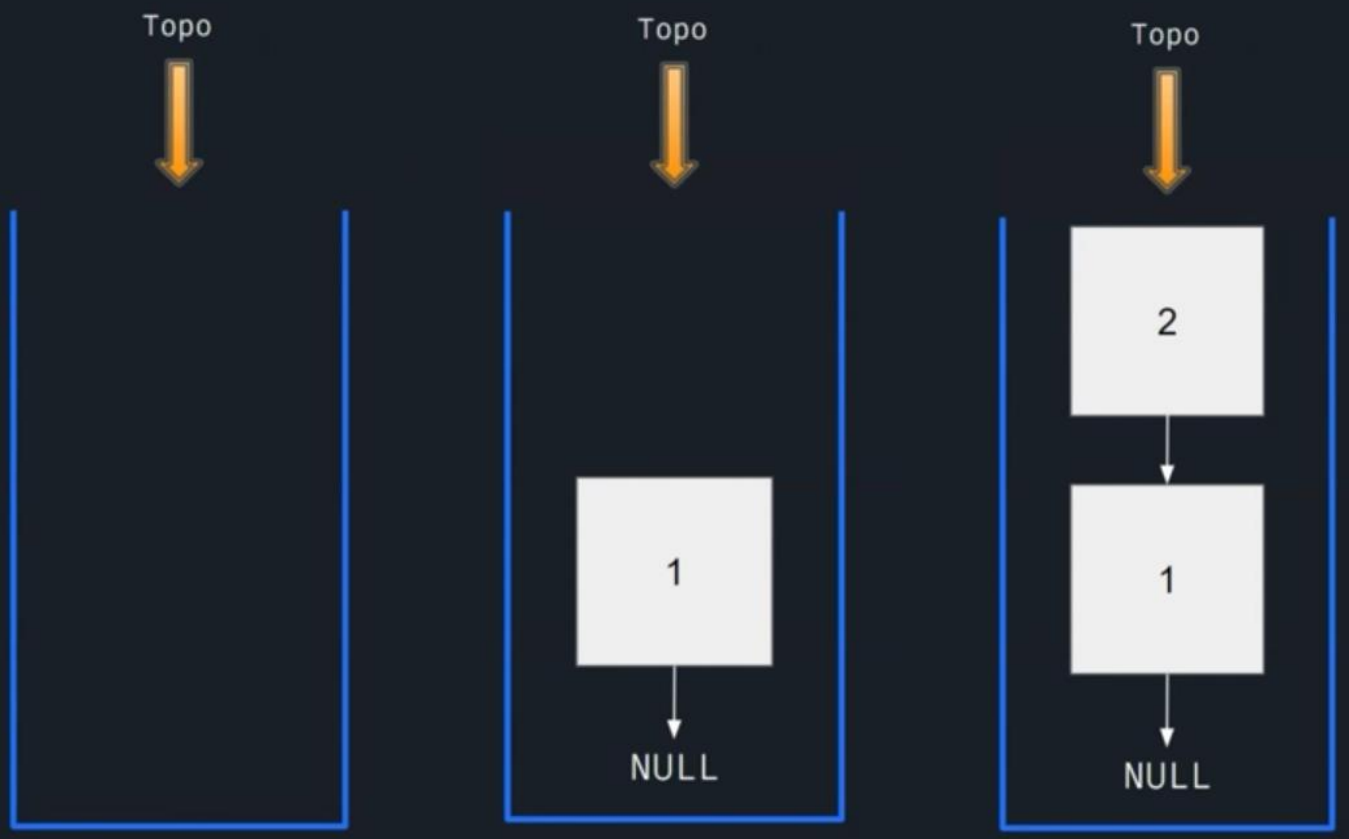
Pilha

Conceito

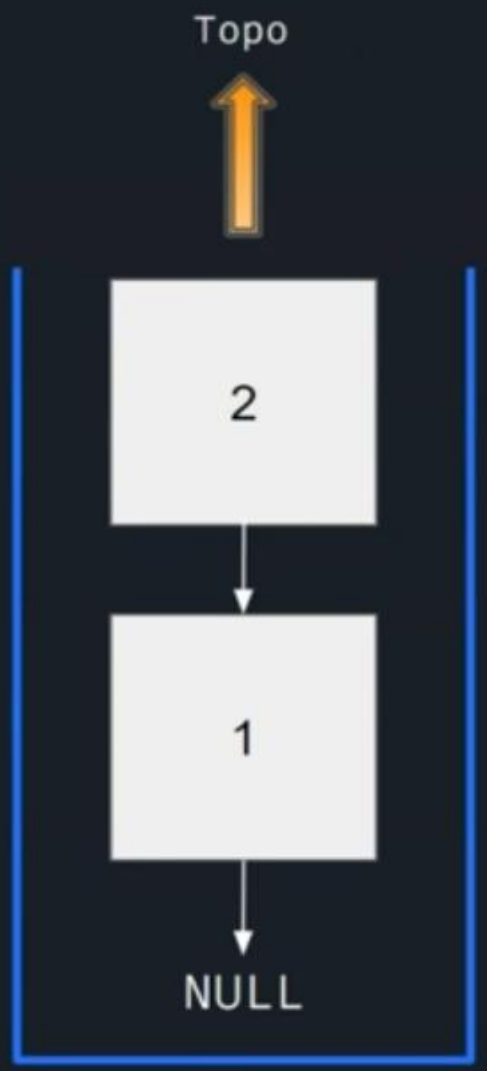
Pilha é uma estrutura de dados ordenada onde o último elemento inserido será o primeiro a ser retirado (*LIFO - last in first out / FILO - first in last out*).

A extremidade por onde é feita a inserção e remoção de elementos é denominada ***top***.

Adicionar o valor 1



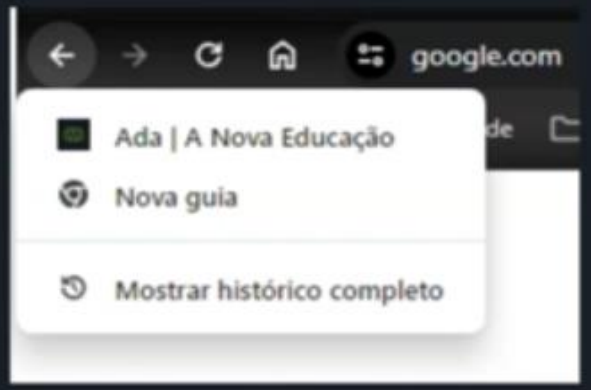
Remover elemento



Pilha

Problemas resolvidos

```
{[5 * (2 + 1)] - [6 / (4 - 2)]} * (3 + 2)
```



Infixa	Prefixa	posfixa
A+B*C	+A*BC	ABC*+
(A+B)*C	*+ABC	AB+C*
(A+B)/(C-D)	/+AB-CD	AB+CD-/
A/(B-C)*D	*/A-BCD	ABC-/D*

Pilha

Principais Operações

Principais operações:

1 - Adicionar item (*push*)

2 - Remover item (*pop*)

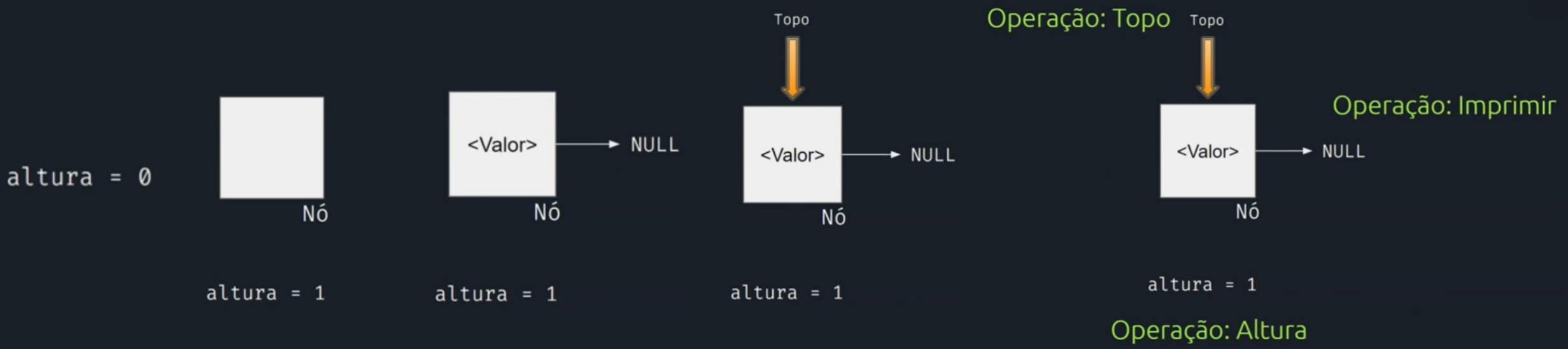
3 - Ler topo (*getTop*)

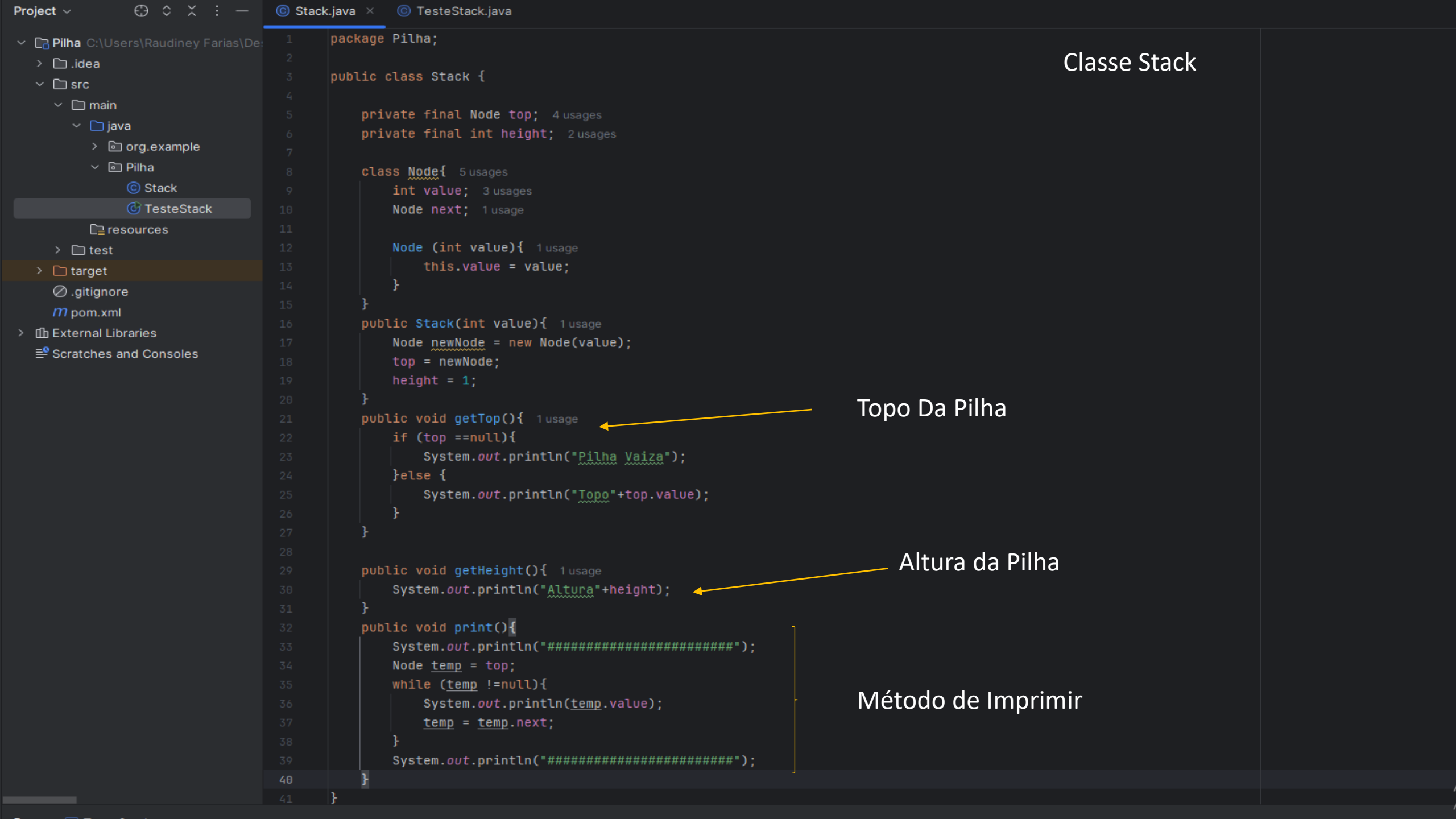
4 - Imprimir (*print*)

Operação opcional:

1 - Altura da pilha (*getHeight*)

Operação: Criar Pilha






```java
package Pilha;

public class Stack {

    private final Node top;  4 usages
    private final int height;  2 usages

    class Node{  5 usages
        int value;  3 usages
        Node next;  1 usage

        Node (int value){  1 usage
            this.value = value;
        }
    }
    public Stack(int value){  1 usage
        Node newNode = new Node(value);
        top = newNode;
        height = 1;
    }
    public void getTop(){  1 usage
        if (top ==null){
            System.out.println("Pilha Vaiza");
        }else {
            System.out.println("Topo"+top.value);
        }
    }

    public void getHeight(){  1 usage
        System.out.println("Altura"+height);
    }
    public void print(){
        System.out.println("##########################");
        Node temp = top;
        while (temp !=null){
            System.out.println(temp.value);
            temp = temp.next;
        }
        System.out.println("##########################");
    }
}
```

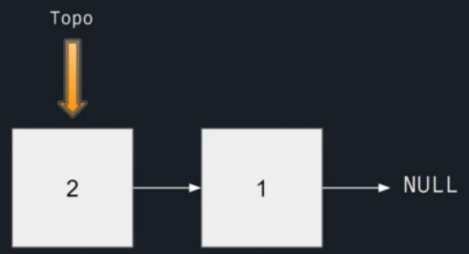
Classe Main

Stack.java | TesteStack.java

```java
package Pilha;

import Pilha.Stack;

public class TesteStack {


    public static void main(String[] args) {
        Stack myStack = new Stack( value: 4);

        myStack.getTop();
        myStack.getHeight();

        myStack.print();

    }

}
```

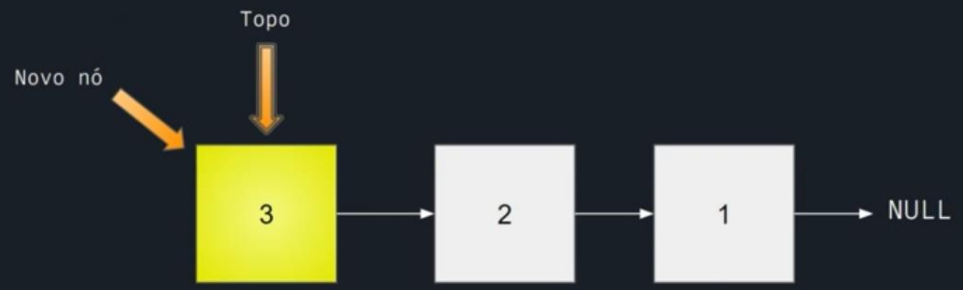
Chamando os métodos

Operação: Inserir (*push*)



altura = 2

Operação: Inserir (*push*)



altura = 2

Código do método

```
}
public void push (int value){  1 usage
    Node newNode = new Node(value);
    if (height ==0){
        top = newNode;
    }else{
        newNode.next = top;
        top = newNode;
    }
    height++;
}
```

Chamando o método:

```
public class TesteStack {


    public static void main(String[] args) {
        Stack myStack = new Stack( value: 2);

        myStack.getTop();
        myStack.getHeight();
        myStack.print();


        myStack.push( value: 5);
        myStack.print();
        myStack.getTop();
        myStack.getHeight();


    }
```

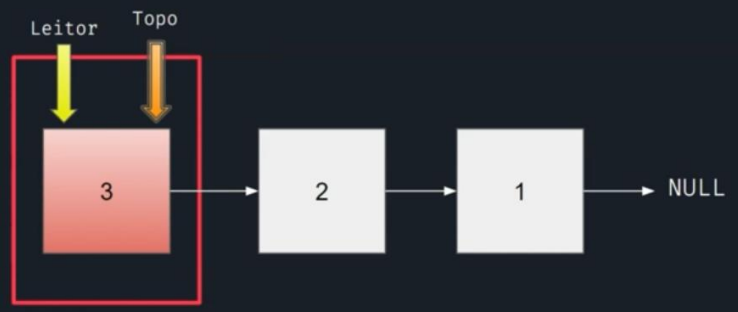
```
Topo - 2
Altura - 1
########################
2
########################
########################
5
2
########################
Topo - 5
Altura - 2

Process finished with exit code 0
```

Imprimindo

Operação: Remover (*pop*)

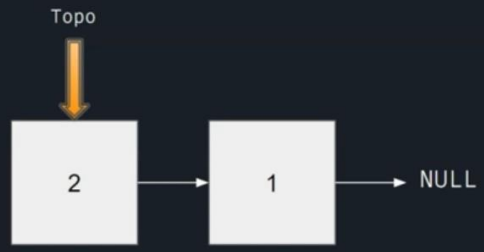
altura = 3

Operação: Remover (*pop*)



altura = 3

Operação: Remover (*pop*)



altura = 2

Linha de Código

```java
public Node pop(){

    if (height == 0) return null;

    Node temp = top;
    top = top.next;
    temp.next = null;
    height--;

    return temp;
}

}
```

Chamando o método:

```java
public static void main(String[] args) {
    Stack myStack = new Stack( value: 2);
    myStack.push( value: 1);

    System.out.println(myStack.pop().value);
    System.out.println(myStack.pop().value);

    System.out.println(myStack.pop() == null);

}
```

```
"E:\Program Files\Java\jdk-21\bin\j
1
2
true

Process finished with exit code 0
```

Problema: Inverta um conjunto

Utilizando uma pilha, inverta um conjunto de dados fornecido pelo usuário.

```java
package InvertendoPilha;

import Pilha.Stack;

public class Main {


    public static void main(String[] args) {
        int [] numeros = {5,4,3,2,1};

        inverter(numeros);
    }
    private static void inverter(final int[] numeros){  1 usage

        Stack stack = new Stack(numeros[0]);
        for (int i=1; i < numeros.length; i++){
            stack.push(numeros[i]);
        }
        var node = stack.pop();
        while(node != null){
            System.out.println(node.getValue());
            node = stack.pop();
        }
    }
}
```

```
"E:\Program Files\Java\jdk-21\bin\java.exe" "-j
1
2
3
4
5

Process finished with exit code 0
```

Pilhas
Conclusão

Agora, implemente uma pilha que utiliza um array / vetor como estrutura de armazenamento!!!

1 - Estrutura dinâmica;

2 - Eficiente na resolução de problemas LIFO;

3 - Simplicidade de implementação.