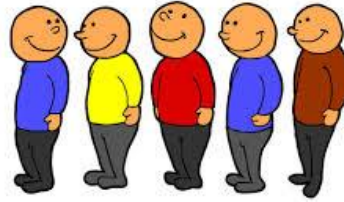


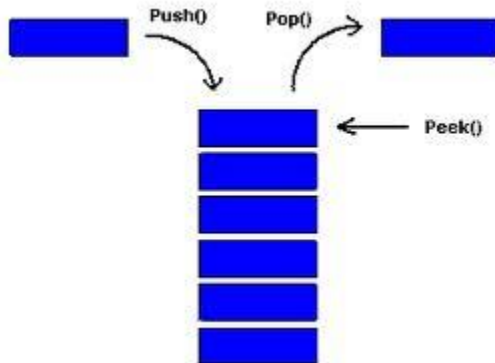


## Unidade 10 – Análise de Algoritmos com Estruturas de Dados Lineares – Parte 4





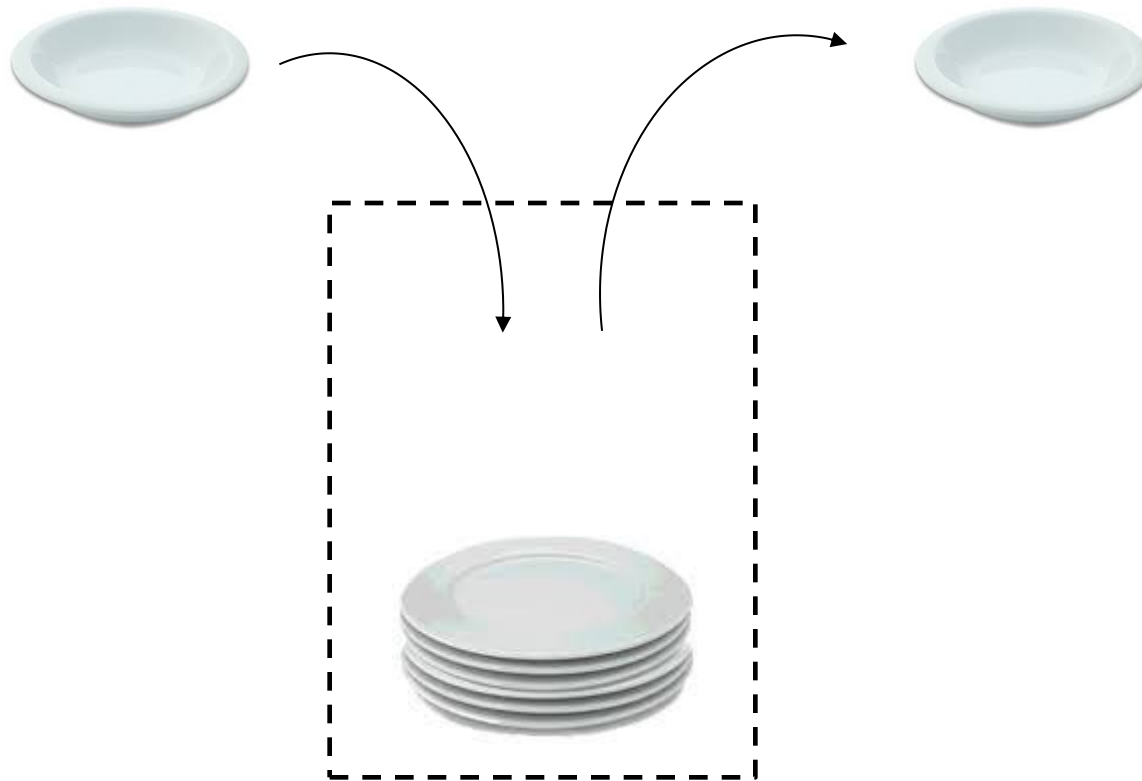
## Pilhas e Filas

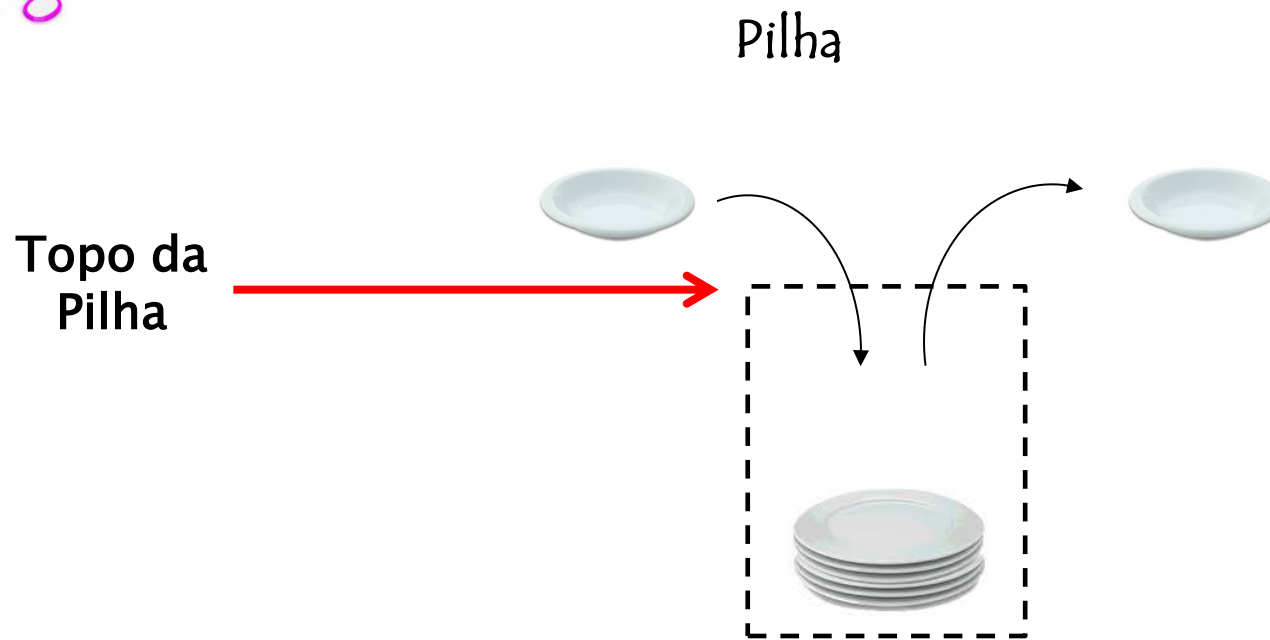




# Pilha

▣ Estrutura de Dados que implementa uma lista **LIFO** (Last Input First Output).



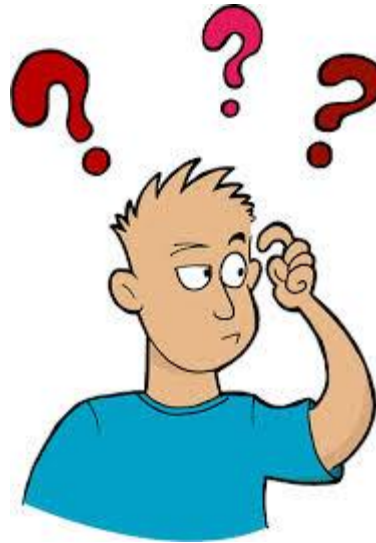


- Estrutura de Dados do tipo **LIFO**
- Inserção de dados: Sempre no Topo da Pilha
- Remoção de dados: Sempre no Topo da Pilha



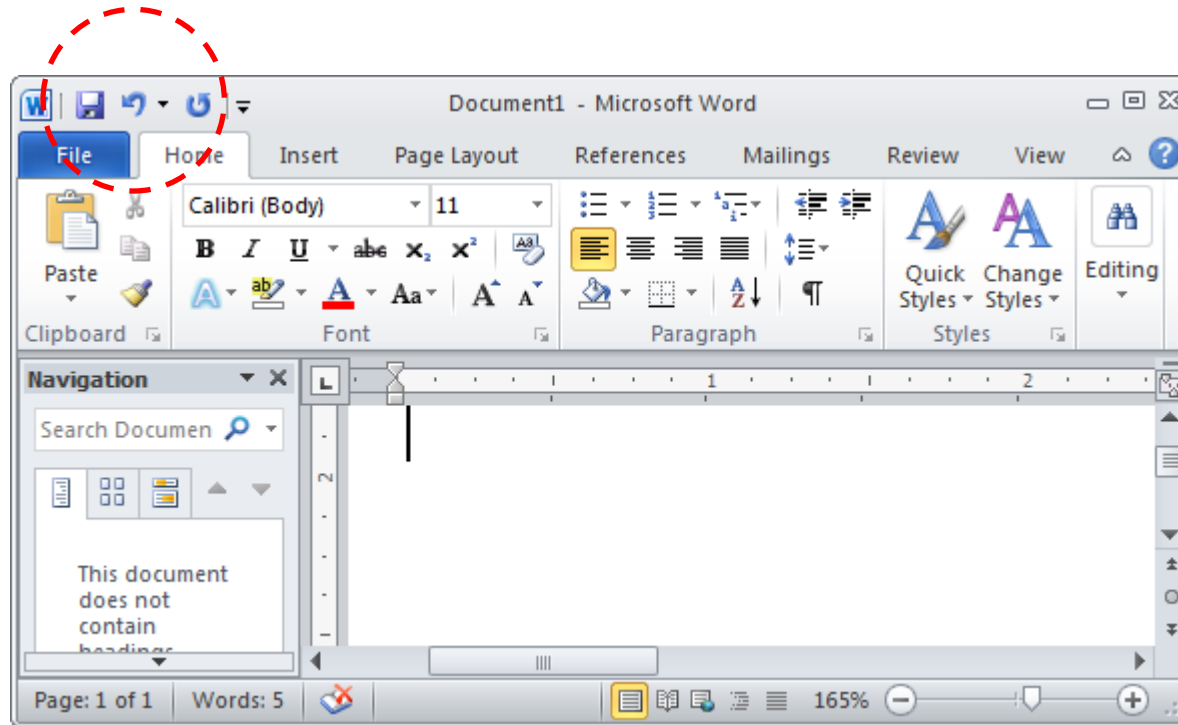


O conceito de Pilha é utilizado em Computação ?





## Pilha – Exemplo de Aplicação



- Recurso “Undo” (desfazer) do MS-Word utiliza uma estrutura de dados do tipo Pilha.





## Pilhas – Aninhamento

```
public static void main (String[] args {  
    int i = 0;  
  
    while ( i < 10 ) {  
        System.println(i);  
        i = i + 1;  
    }  
}
```

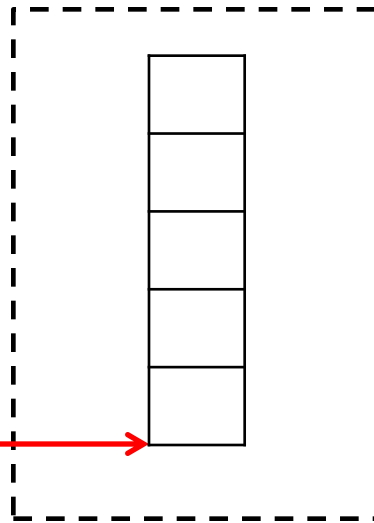
- Podemos empregar uma pilha para checar o aninhamento de blocos no programa acima.





## Pilhas – Aninhamento

```
public static void main (String[] args {  
    int i = 0;  
  
    while ( i < 10 ) {  
        System.println(i);  
        i = i + 1;  
    }  
}
```



O compilador irá  
proceder à operação  
de scanner no texto  
do programa.

Topo da Pilha

Pilha Vazia





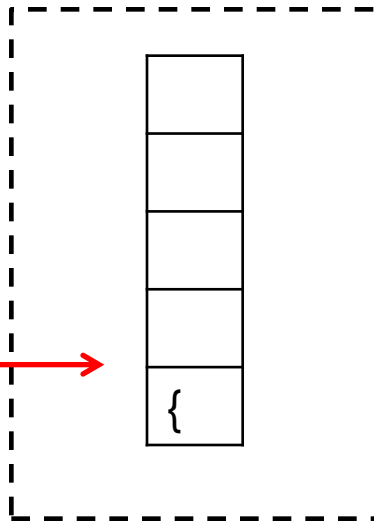


## Pilhas – Aninhamento

```
public static void main (String[] args) {  
    int i = 0;  
  
    while ( i < 10 ) {  
        System.println(i);  
        i = i + 1;  
    }  
}
```



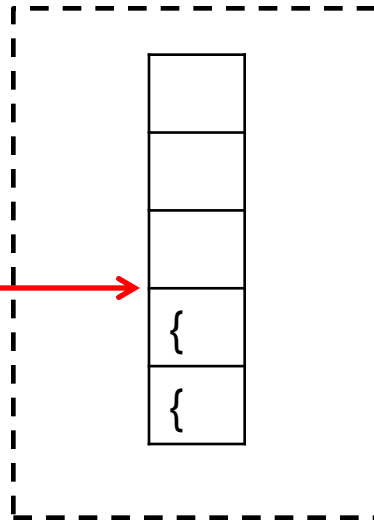
Topo da Pilha





## Pilhas – Aninhamento

```
public static void main (String[] args {  
    int i = 0;  
  
    while ( i < 10 ) {  
        System.println(i);  
        i = i + 1;  
    }  
}
```



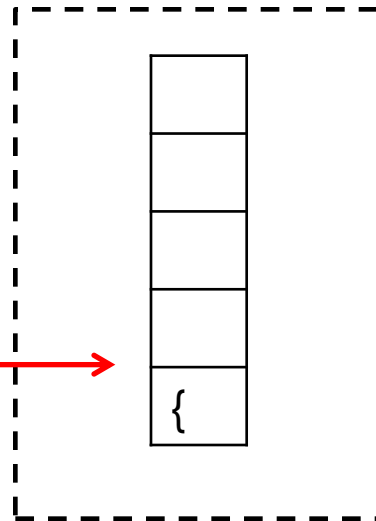
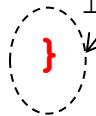
Topo da Pilha





# Pilhas – Aninhamento

```
public static void main (String[] args {  
    int i = 0;  
  
    while ( i < 10 ) {  
        System.println(i);  
        i = i + 1;  
    }  
}
```



Topo da Pilha



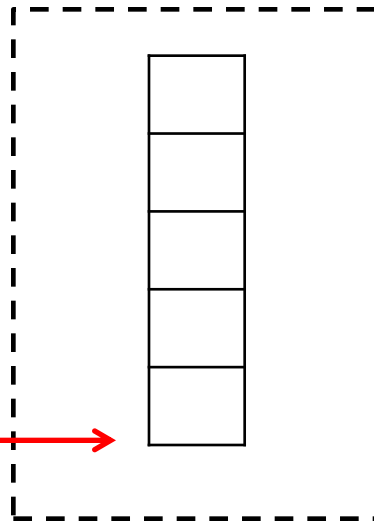


# Pilhas – Aninhamento

```
public static void main (String[] args {  
    int i = 0;  
  
    while ( i < 10 ) {  
        System.println(i);  
        i = i + 1;  
    }  
}
```



**Topo da Pilha**



**Pilha Vazia**

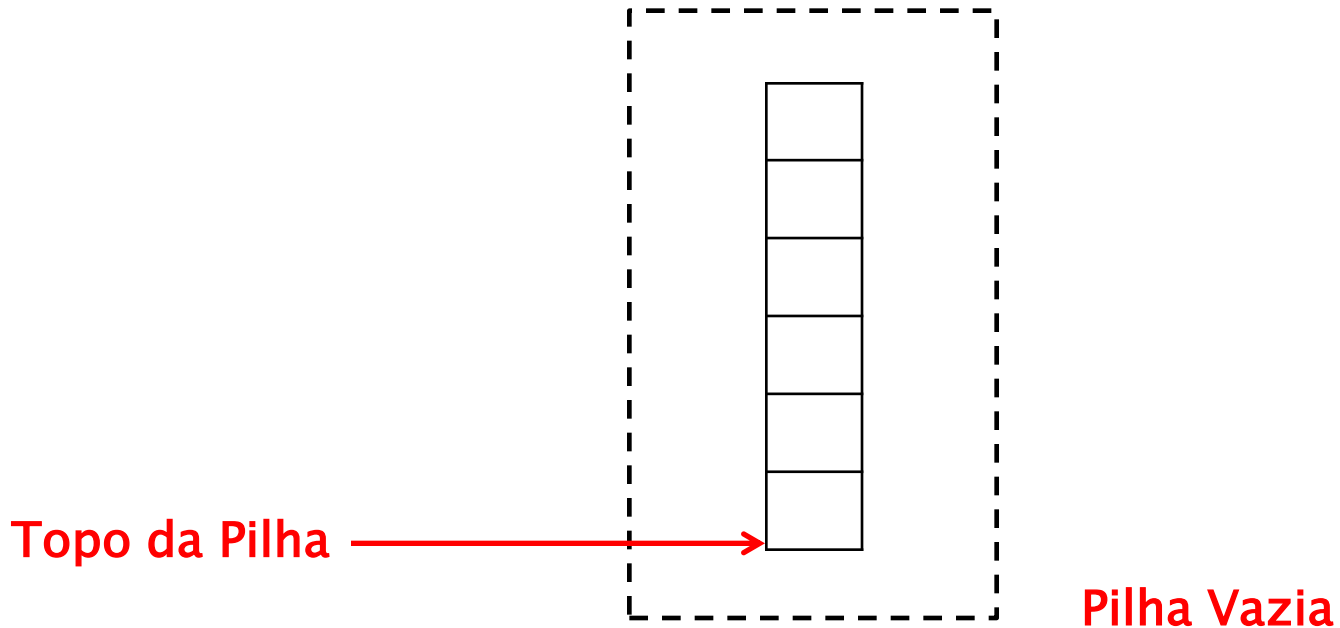
Ao final da operação, se a pilha estiver vazia, a quantidade de { é igual à quantidade de }

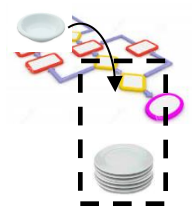




## Pilha – Exemplo de Aplicação

- Considere uma lista de números em ordem crescente: 1,2,3,4,5,6

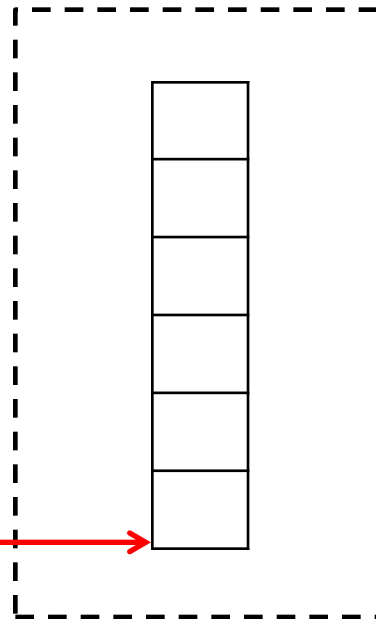




# Empilhamento

1

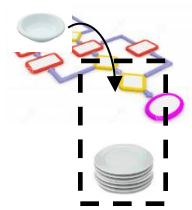
❏ Operação de Empilhamento da lista: **1,2,3,4,5,6**



**Topo da Pilha** →

**Pilha Vazia**





# Empilhamento

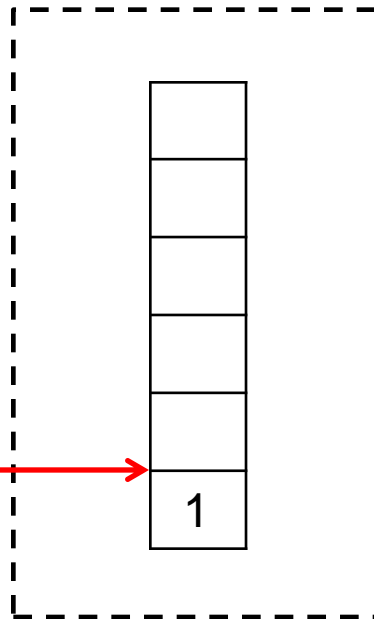
2

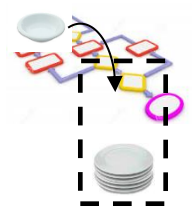
❏ Operação de Empilhamento da lista:

,2,3,4,5,6



Topo da Pilha





# Empilhamento

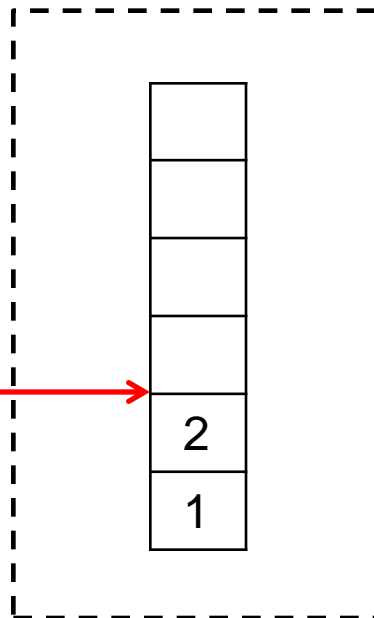
3

❏ Operação de Empilhamento da lista:

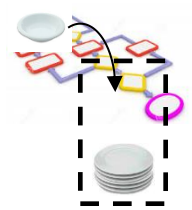
,3,4,5,6



Topo da Pilha





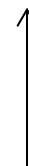


# Empilhamento

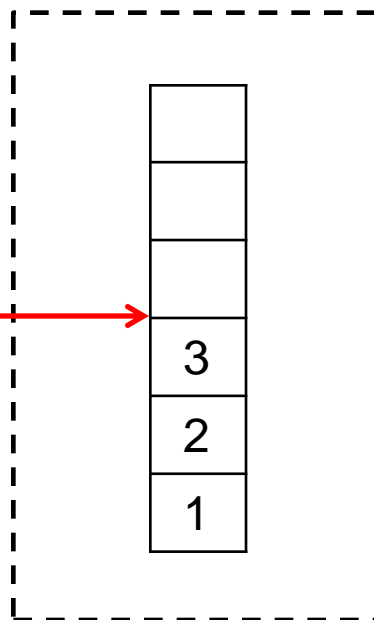
4

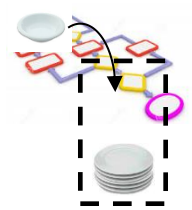
❏ Operação de Empilhamento da lista:

,4,5,6



Topo da Pilha





# Empilhamento

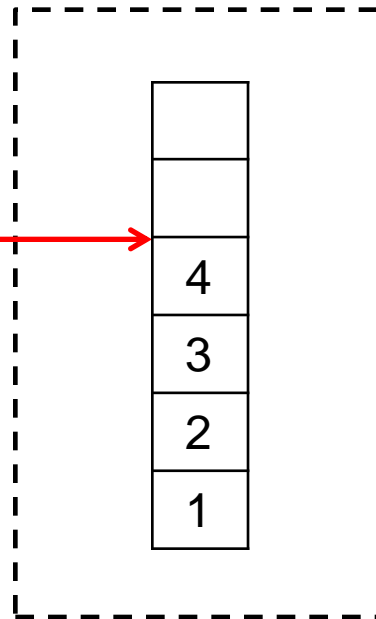
5

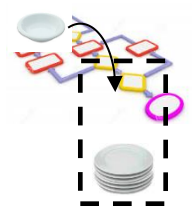
❏ Operação de Empilhamento da lista:

,5,6



Topo da Pilha



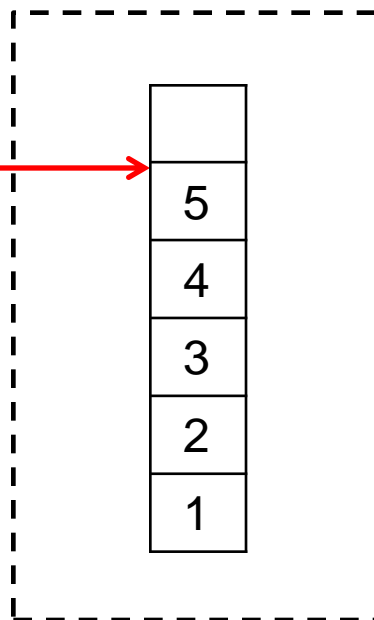


# Empilhamento

6

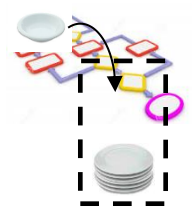
❏ Operação de Empilhamento da lista:

Topo da Pilha



,6  
↑



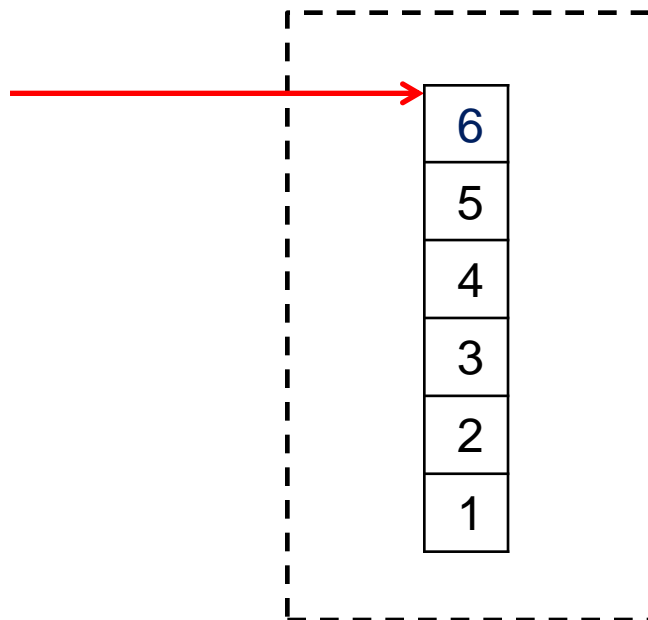


# Empilhamento

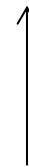
7

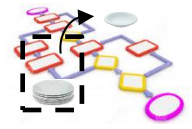
▣ Operação de Empilhamento concluída:

Topo da Pilha



Pilha Cheia





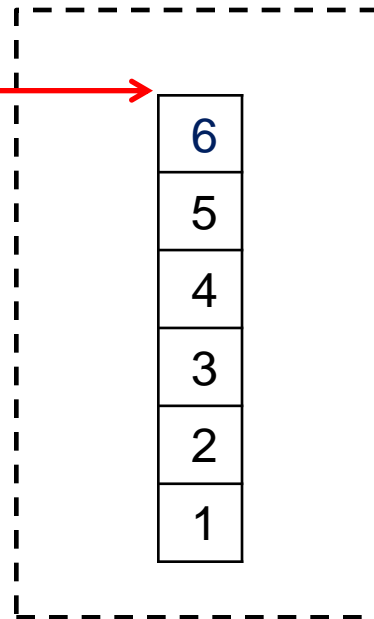
# Desempilhamento

1

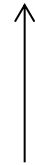


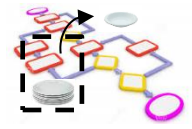
Operação de Desempilhamento da pilha:

Topo da Pilha



Pilha Cheia





# Desempilhamento

2

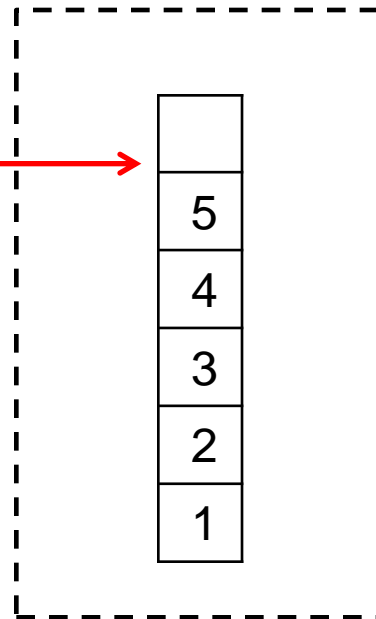


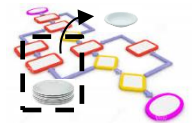
Operação de Desempilhamento da pilha:

6



Topo da Pilha





# Desempilhamento

3

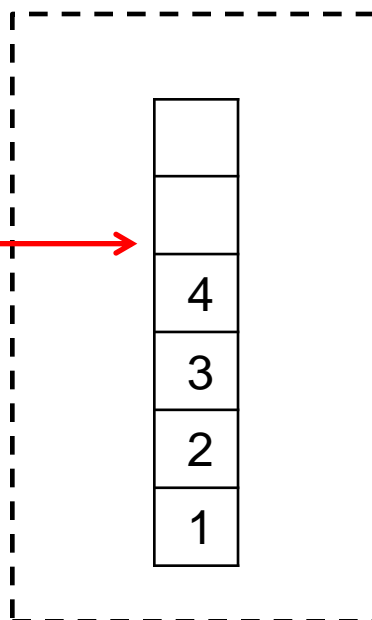


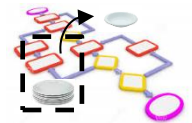
Operação de Desempilhamento da pilha:

6,5



Topo da Pilha





# Desempilhamento

4

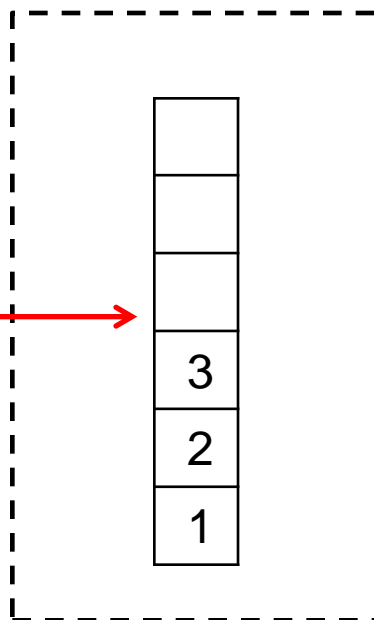


Operação de Desempilhamento da pilha:

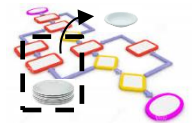
6,5,4



Topo da Pilha







# Desempilhamento

5

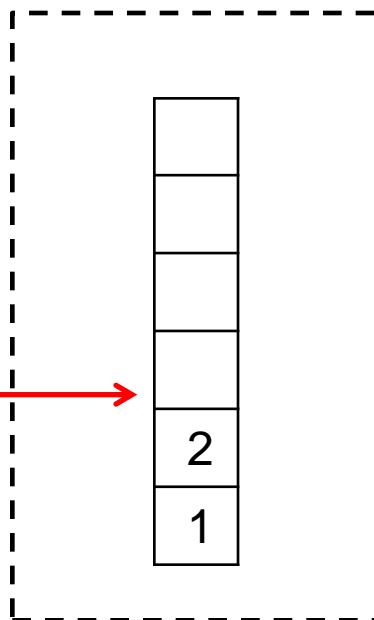


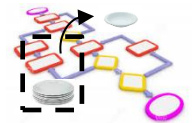
Operação de Desempilhamento da pilha:

6,5,4,3



Topo da Pilha





## Desempilhamento

6

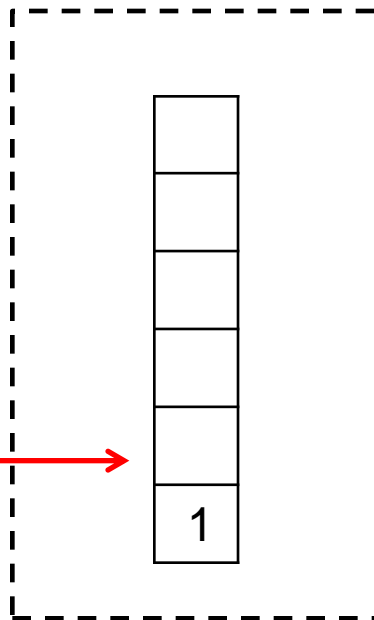


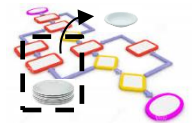
Operação de Desempilhamento da pilha:

6,5,4, 3,2



Topo da Pilha





# Desempilhamento

7

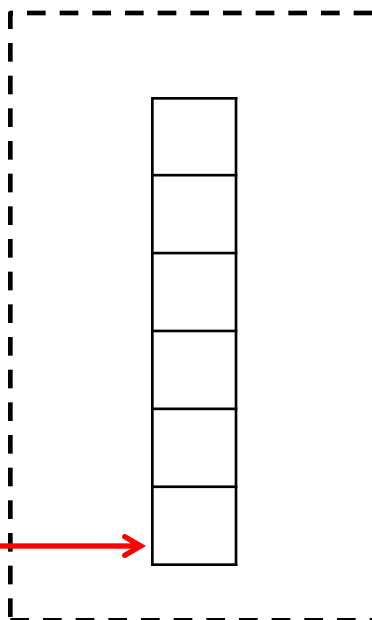


Operação de Desempilhamento da pilha:

6,5,4, 3,2,1



Topo da Pilha



Pilha Vazia





## Aplicação – Pilha

- ▣ Lista original antes do empilhamento: **1,2,3,4,5,6**
- ▣ Lista após o desempilhamento: **6,5,4,3,2,1**

**Portanto, os elementos da lista foram invertidos !**





Como podemos fazer esta operação aritmética ?

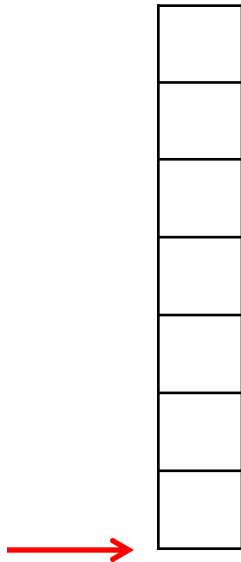
$$(((1 + 2) * 3) + (2 * (1 + 3)))$$





## Pilhas – Operações Aritméticas

Pilha



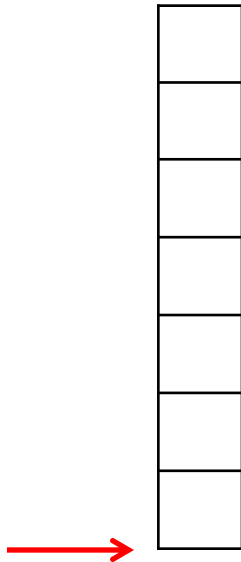
$(( (1 + 2) * 3 ) + ( 2 * (1 + 3) ))$





## Pilhas – Operações Aritméticas

Pilha



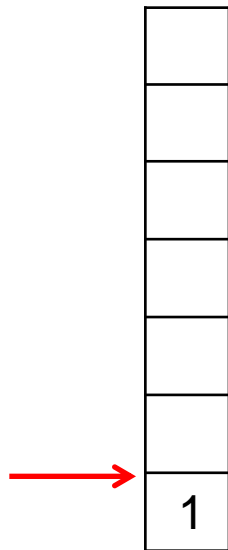
$(( (1 + 2) * 3 ) + ( 2 * (1 + 3) ))$





## Pilhas – Operações Aritméticas

Pilha



$(( (1 + 2) * 3 ) + ( 2 * (1 + 3) ))$

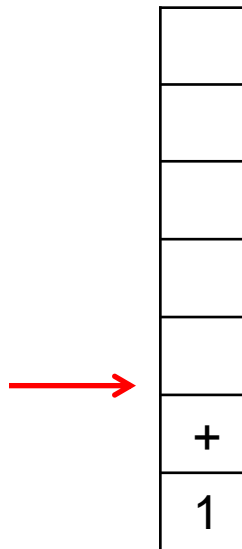






## Pilhas – Operações Aritméticas

Pilha



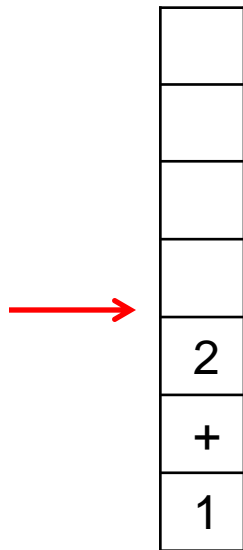
$(( (1 + 2) * 3 ) + ( 2 * (1 + 3) ))$





## Pilhas – Operações Aritméticas

Pilha



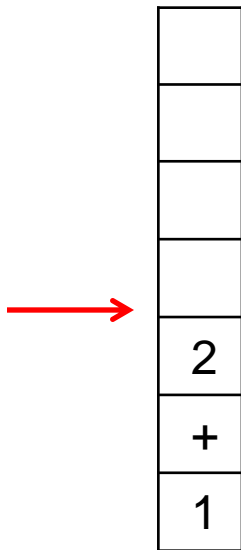
$(( (1 + 2) * 3 ) + ( 2 * (1 + 3) ))$





## Pilhas – Operações Aritméticas

Pilha



$(( (1 + 2) * 3 ) + ( 2 * (1 + 3) ))$

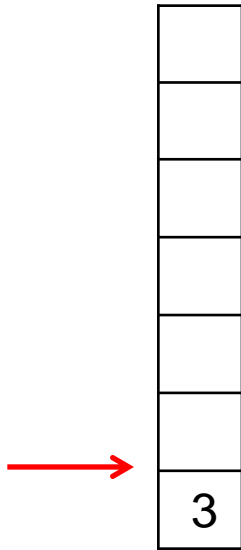
$$2 + 1 = 3$$





## Pilhas – Operações Aritméticas

Pilha



$(( (1 + 2) * 3 ) + ( 2 * (1 + 3) ))$

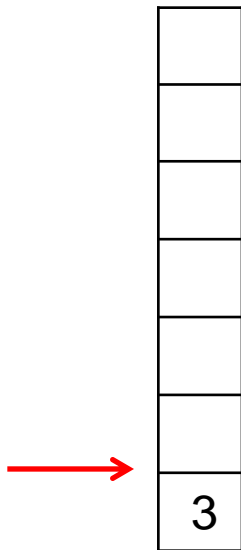
$$2 + 1 = 3$$





## Pilhas – Operações Aritméticas

Pilha



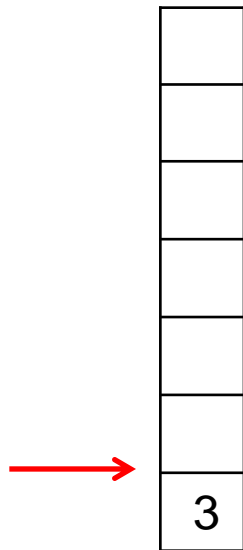
$(( (1 + 2) * 3 ) + ( 2 * (1 + 3) ))$





## Pilhas – Operações Aritméticas

Pilha



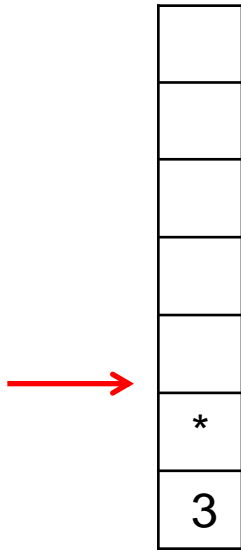
$(( (1 + 2) * 3 ) + ( 2 * (1 + 3) ))$





## Pilhas – Operações Aritméticas

Pilha



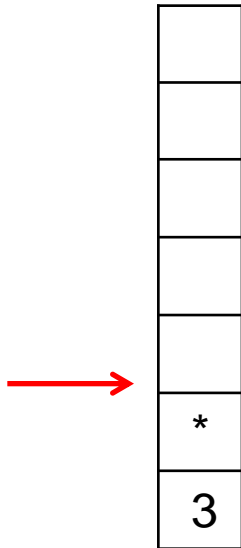
$(( (1 + 2) * 3 ) + ( 2 * (1 + 3) ))$





## Pilhas – Operações Aritméticas

Pilha



$(( (1 + 2) * 3 ) + ( 2 * (1 + 3) ))$

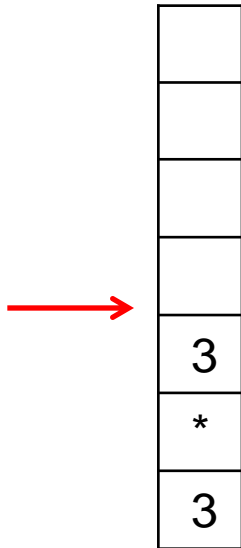






## Pilhas – Operações Aritméticas

Pilha



$(( (1 + 2) * 3 ) + ( 2 * (1 + 3) ))$

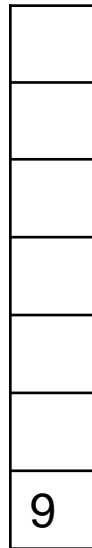
$$3 * 3 = 9$$





## Pilhas – Operações Aritméticas

Pilha



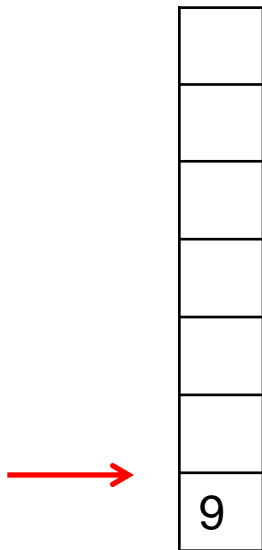
$(( (1 + 2) * 3 ) + ( 2 * (1 + 3) ))$





## Pilhas – Operações Aritméticas

Pilha



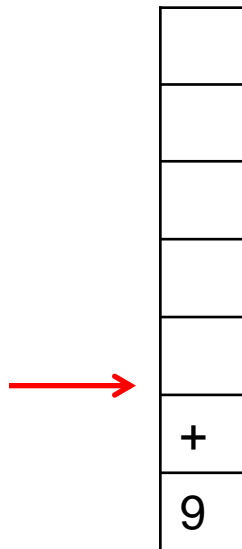
$$(((1 + 2) * 3) + (2 * (1 + 3)))$$





## Pilhas – Operações Aritméticas

Pilha



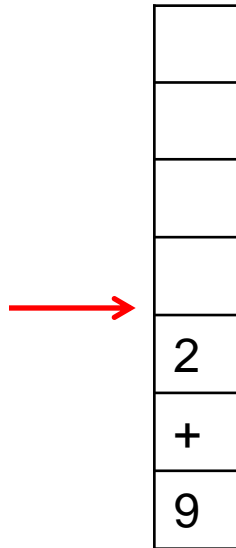
$(( (1 + 2) * 3 ) + ( 2 * (1 + 3) ))$





## Pilhas – Operações Aritméticas

Pilha

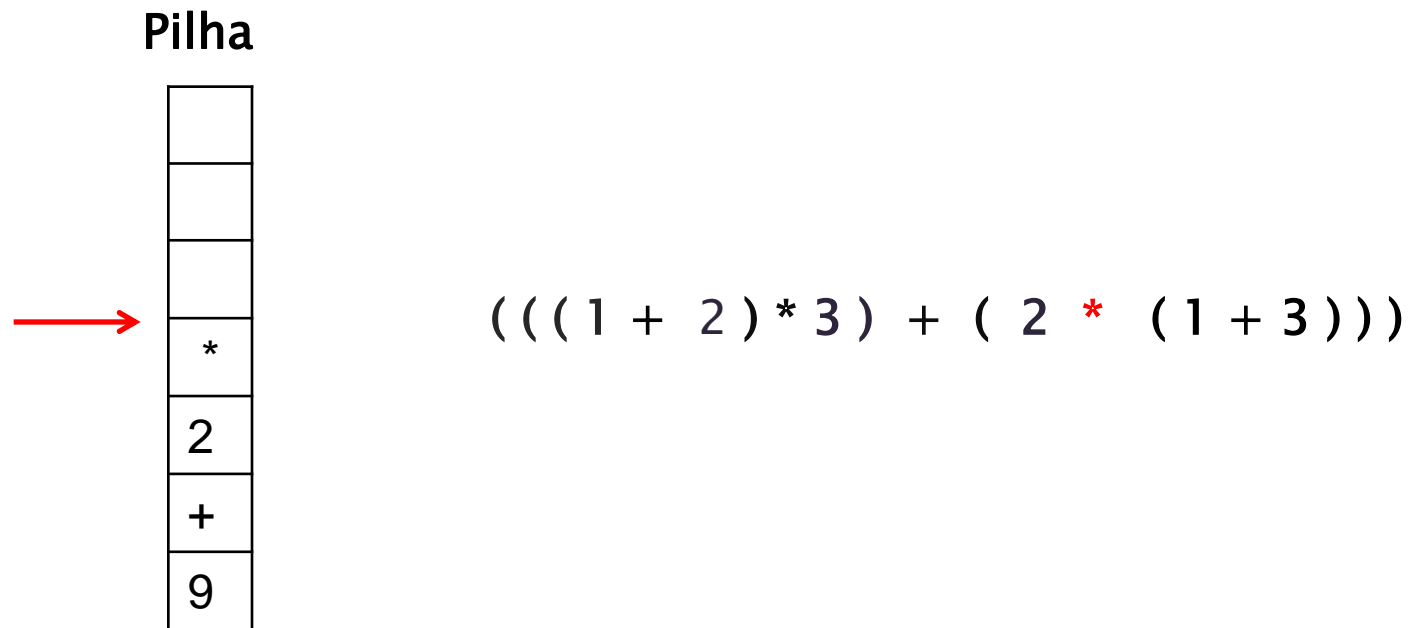


$(( (1 + 2) * 3 ) + ( 2 * (1 + 3) ))$





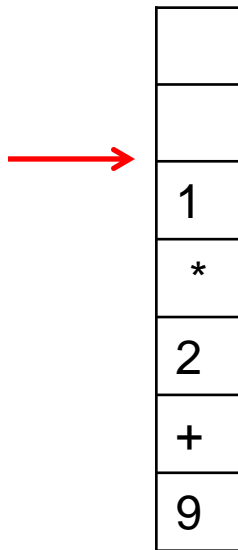
## Pilhas – Operações Aritméticas





## Pilhas – Operações Aritméticas

Pilha



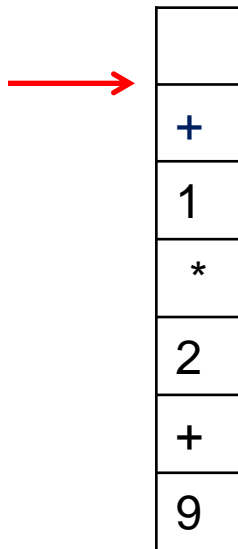
$(( (1 + 2) * 3 ) + ( 2 * (1 + 3) ))$





## Pilhas – Operações Aritméticas

Pilha



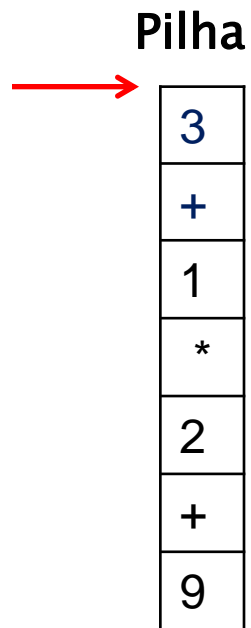
$(( (1 + 2) * 3 ) + ( 2 * (1 + 3) ))$







## Pilhas – Operações Aritméticas

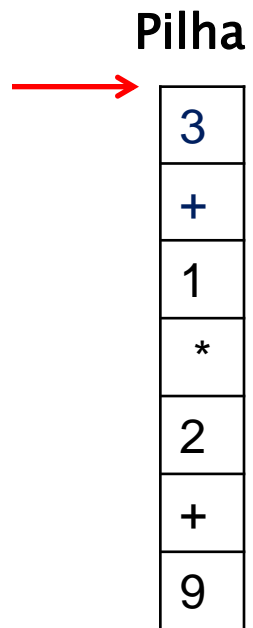


$(( (1 + 2) * 3 ) + ( 2 * (1 + 3) ))$





## Pilhas – Operações Aritméticas



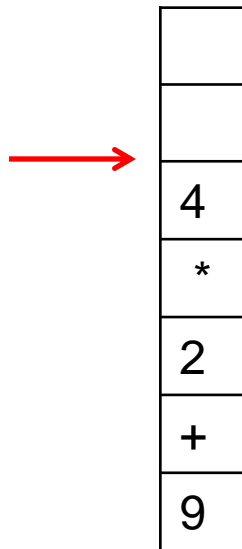
$(( (1 + 2) * 3 ) + ( 2 * (1 + 3) ))$





## Pilhas – Operações Aritméticas

Pilha



$(( (1 + 2) * 3 ) + ( 2 * (1 + 3) ))$

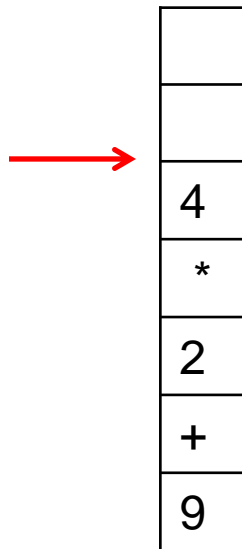
$$1 + 3 = 4$$





## Pilhas – Operações Aritméticas

Pilha



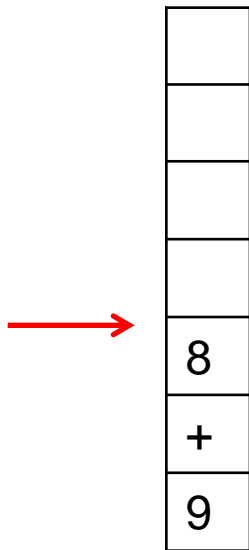
$(( (1 + 2) * 3 ) + ( 2 * (1 + 3) ) )$





## Pilhas – Operações Aritméticas

Pilha



$(( (1 + 2) * 3 ) + ( 2 * (1 + 3) ) )$

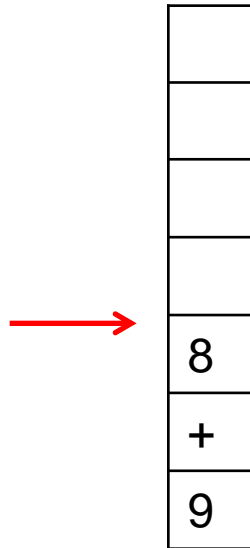
$$2 * 4 = 8$$





# Pilhas – Operações Aritméticas

Pilha



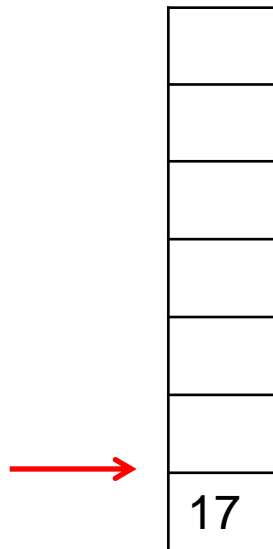
$(( (1 + 2) * 3 ) + ( 2 * (1 + 3) ))$





## Pilhas – Operações Aritméticas

Pilha



$(( (1 + 2) * 3 ) + ( 2 * (1 + 3) ))$

$$8 + 9 = 17$$





# Implementação de Pilhas

- Em uma pilha, todo o acesso à seus elementos se dão no topo.
- Inserções e remoções sempre são feitas pelo topo.
- Existem duas operações básicas na pilha: a operação para empilhar um novo elemento (**push**) e a operação para desempilhar um elemento (**pop**).
- Quando se conhece a priori o tamanho da pilha, podemos implementá-la com o emprego de arrays.
- No entanto, quando o número máximo de elementos da pilha é desconhecido, devemos implementá-la com uma estrutura de dados dinâmica, lista ligada ou encadeada.

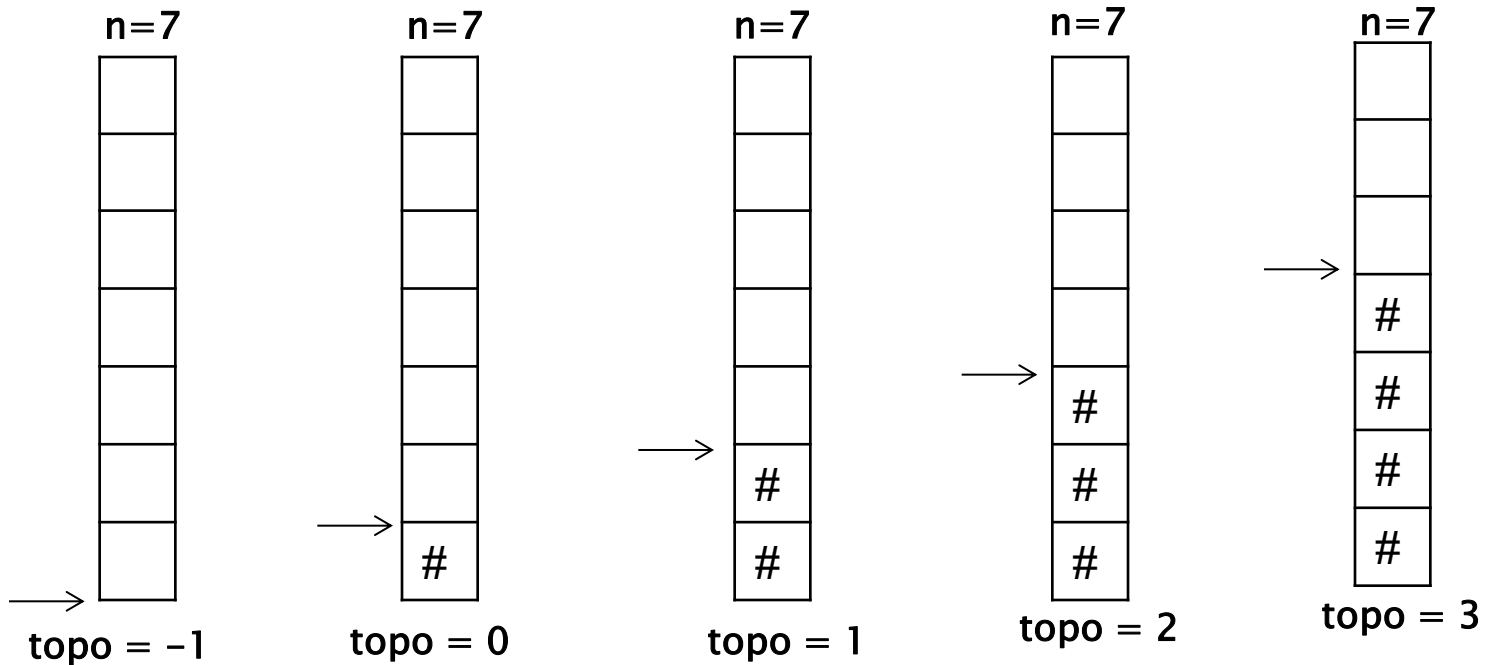






## Implementação de Pilhas com arrays

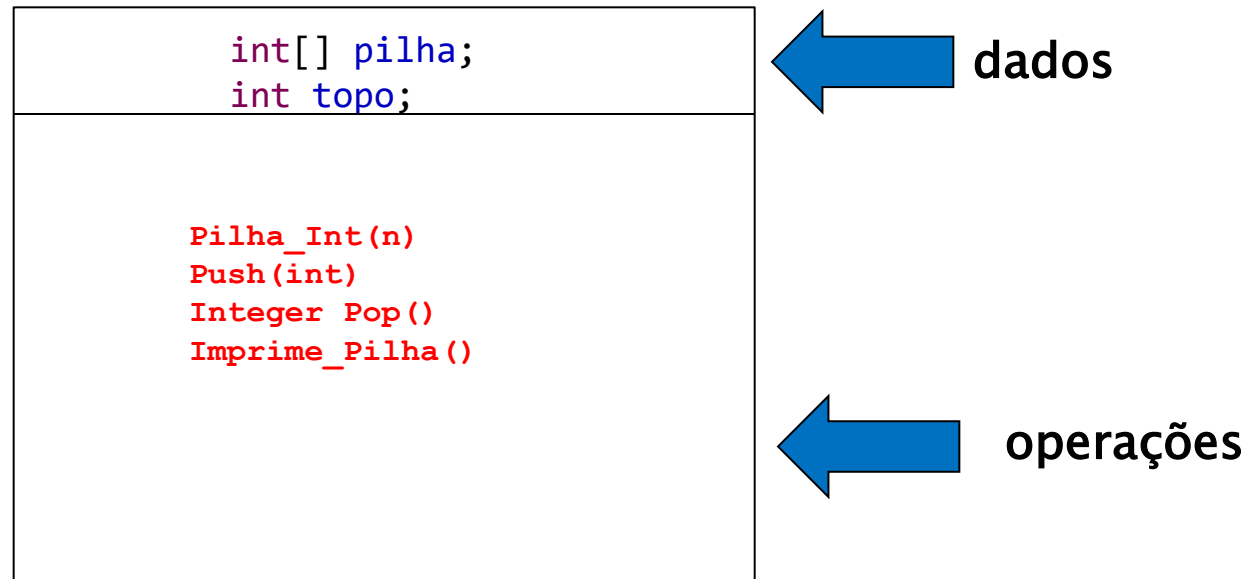
- A estrutura que representa a pilha deve ser composta pelo array (com um tamanho previamente conhecido) e por um número (índice) que representa o topo da pilha.





## Implementação de Pilhas com arrays

# Tipo Abstrato de Dados: Pilha\_Int





## Implementação de Pilhas com arrays

```
package maua;  
  
public class Pilha_Int {  
  
    int[] pilha;  
    int topo;  
  
    public Pilha_Int(int n) {  
        pilha = new int[n];  
        topo = -1;  
    }  
}
```





## Implementação de Pilhas com arrays

```
public void Push(int item) {  
  
    if (topo >= pilha.length - 1)  
  
        System.out.println ("Stack Overflow! Erro no push...!");  
  
    else {  
        topo = topo + 1;  
        pilha[topo] = item;  
    }  
  
}
```





## Implementação de Pilhas com arrays

```
public Integer Pop() {  
  
    if (topo <= -1) {  
        System.out.println("Stack empty! Erro no pop...!");  
        return null;  
    }  
    else {  
        topo = topo - 1;  
        return (pilha[topo+1]);  
    }  
}
```





## Implementação de Pilhas com arrays

```
public void Imprime_Pilha() {  
    System.out.print("Pilha: ");  
    int trab = topo;  
    if (trab <= -1 )  
        System.out.print(" vazia!");  
    else {  
        while (trab >= 0) {  
            System.out.print(" " + pilha[trab]);  
            trab = trab - 1;  
        }  
    }  
    System.out.println(" ");  
}
```





# Implementação de Pilhas com arrays

```
public static void main(String[] args) {  
    Pilha_Int x = new Pilha_Int(3);  
    x.Imprime_Pilha();  
  
    x.Push(9);  
    x.Push(4);  
    x.Push(3);  
  
    x.Imprime_Pilha();  
  
    x.Pop();  
    x.Imprime_Pilha();  
  
    x.Pop();  
    x.Imprime_Pilha();  
  
    x.Pop();  
    x.Imprime_Pilha();  
  
    }  
}
```

**Pilha: vazia!**  
**Pilha: 3 4 9**  
**Pilha: 4 9**  
**Pilha: 9**  
**Pilha: vazia!**









# Fila

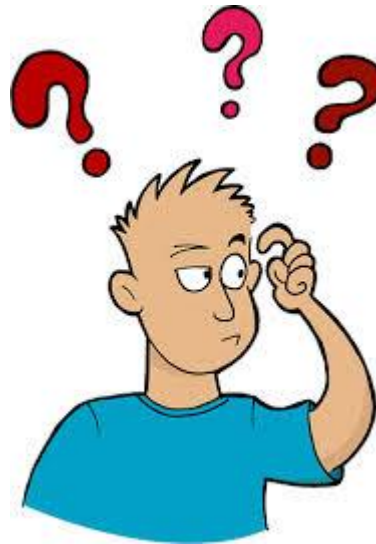


- ▣ Quem primeiro entra na fila, primeiro será atendido.
- ▣ Inserções sempre são feitas no final da fila.
- ▣ Remoções sempre são feitas no início da fila.



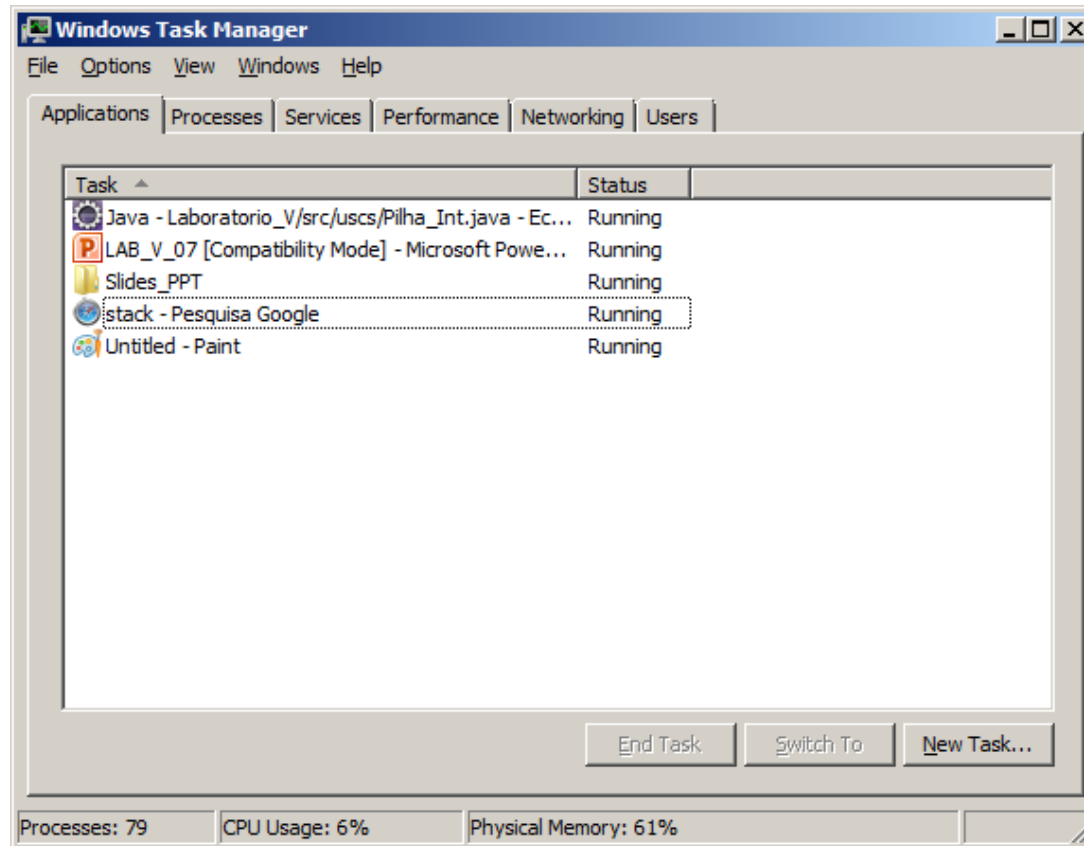


O conceito de Fila é utilizado em Software ?





# Fila – Exemplo de Aplicação

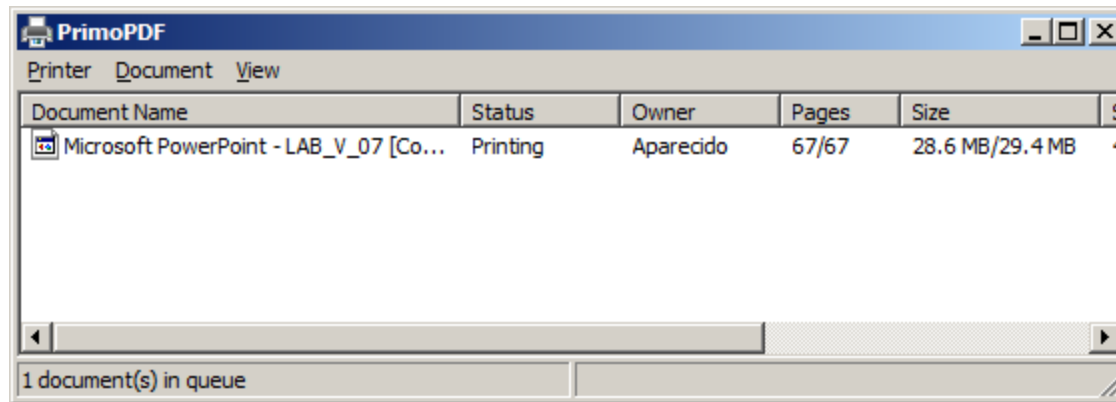


## Fila de Processos





## Fila – Exemplo de Aplicação

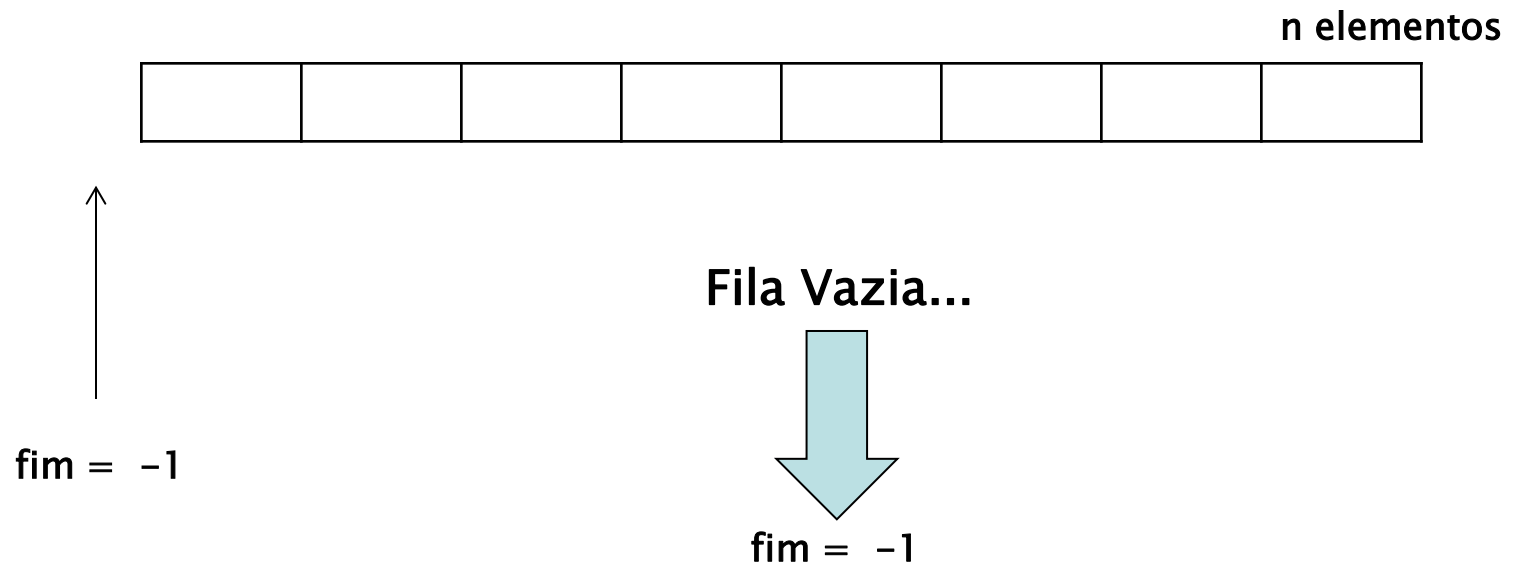


### Fila de Impressão



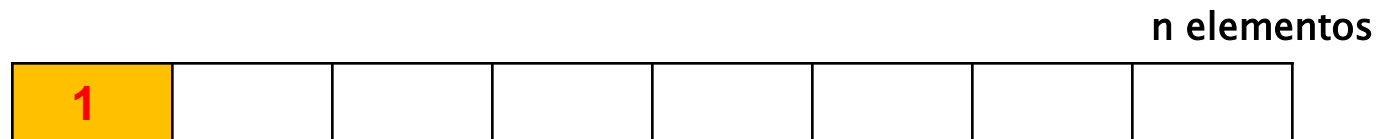


## Implementação de Filas com Arrays



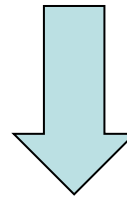


## Inserção na Fila



fim = 0

Fila com 1 elemento

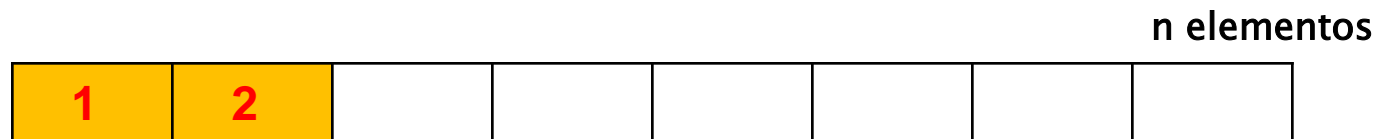


fim = 0





## Inserção na Fila



↑  
fim = 1

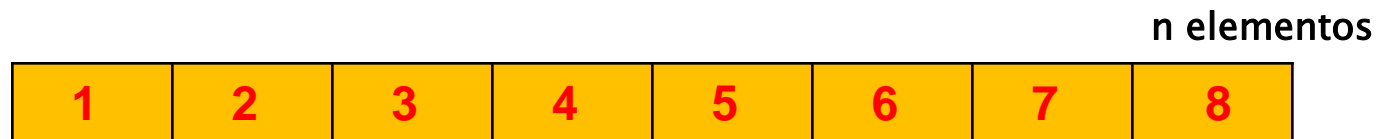
Fila com 2 elementos

↓  
fim = 1



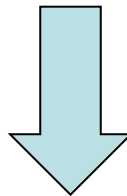


## Inserção na Fila



↑  
 $\text{fim} = n - 1$

Fila Cheia



$\text{fim} = n - 1$

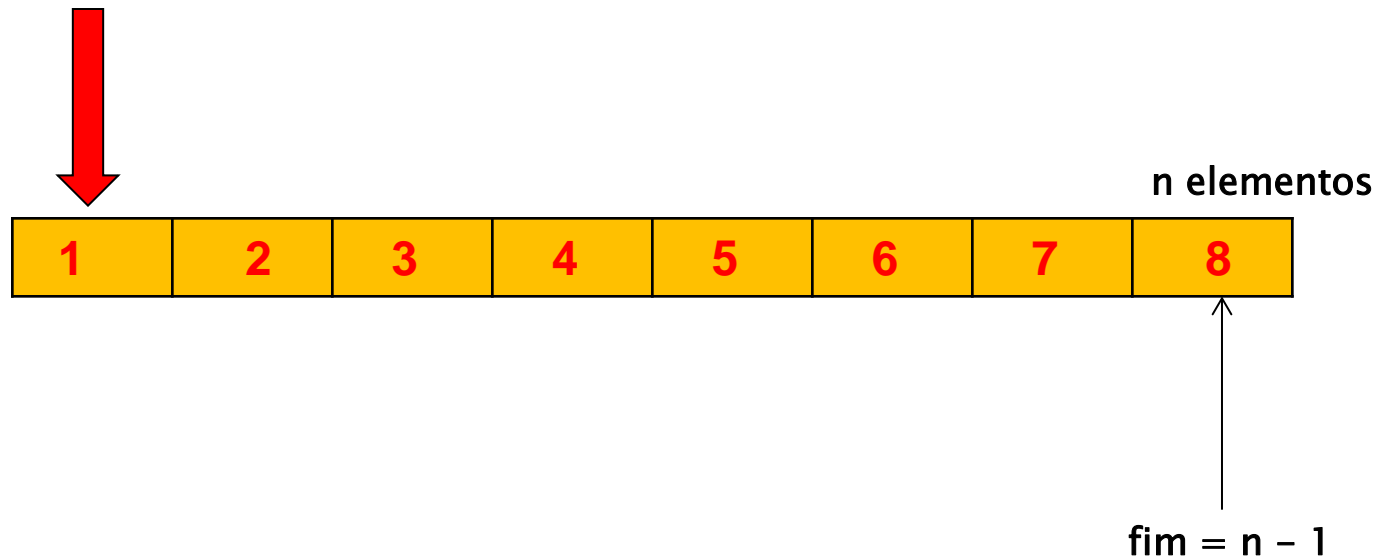






## Remoção da Fila

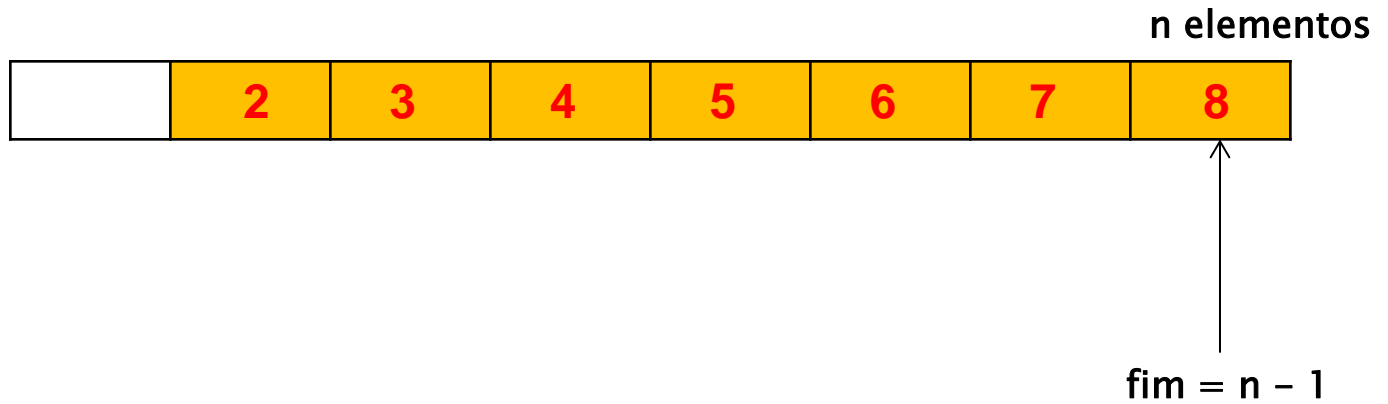
Elemento a ser removido





## Remoção da Fila

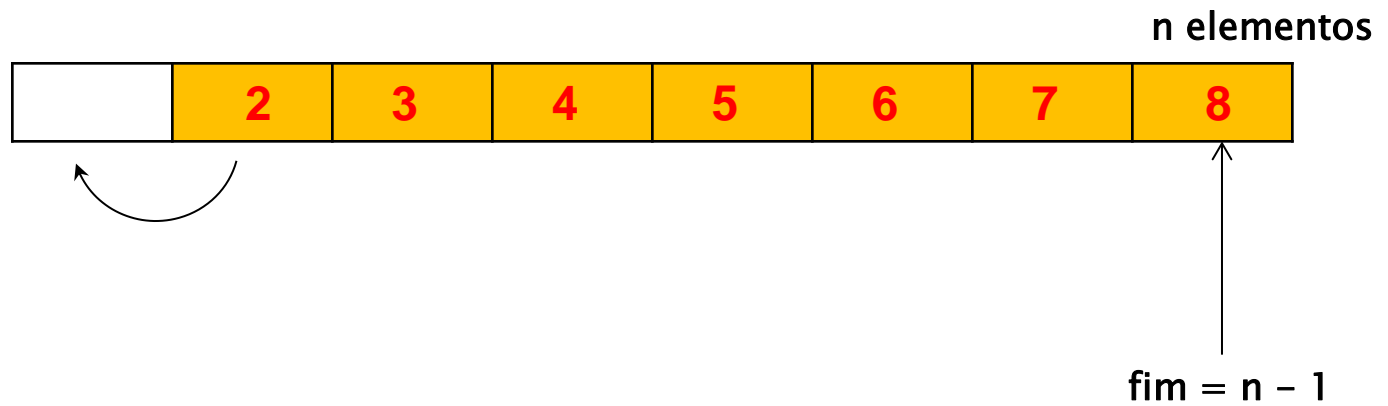
**A fila anda !!!**





## Remoção da Fila

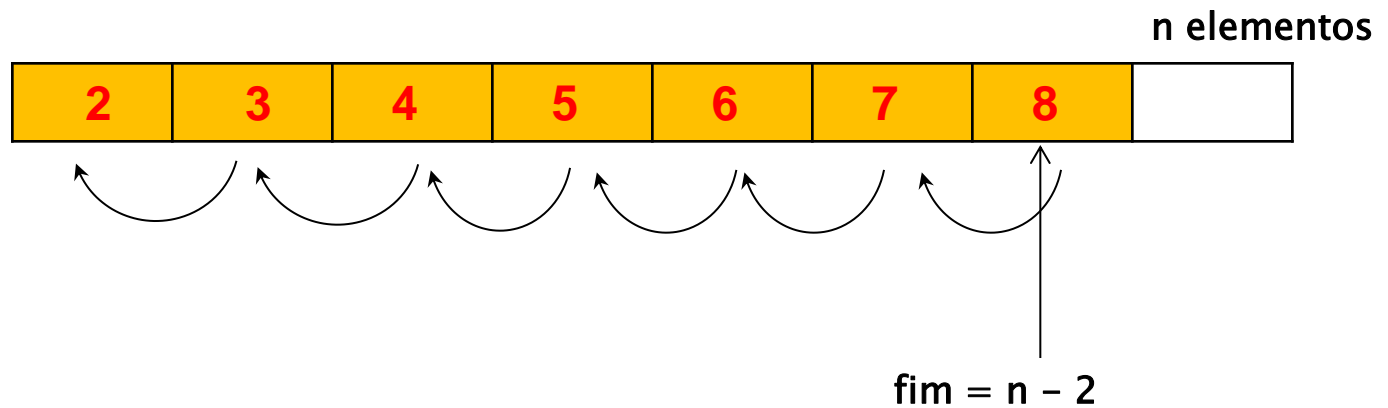
**A fila anda !!!**





## Remoção da Fila

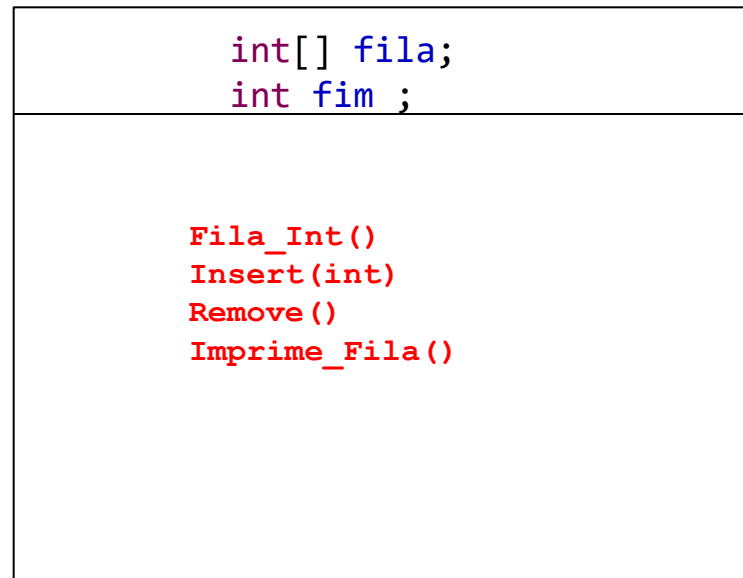
**A fila anda !!!**





# Implementação de Filas com arrays

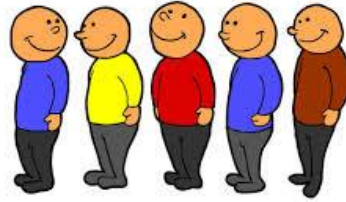
## Tipo Abstrato de Dados: Fila\_Int



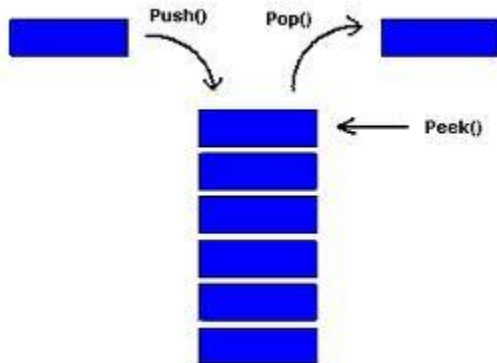
← dados

← operações





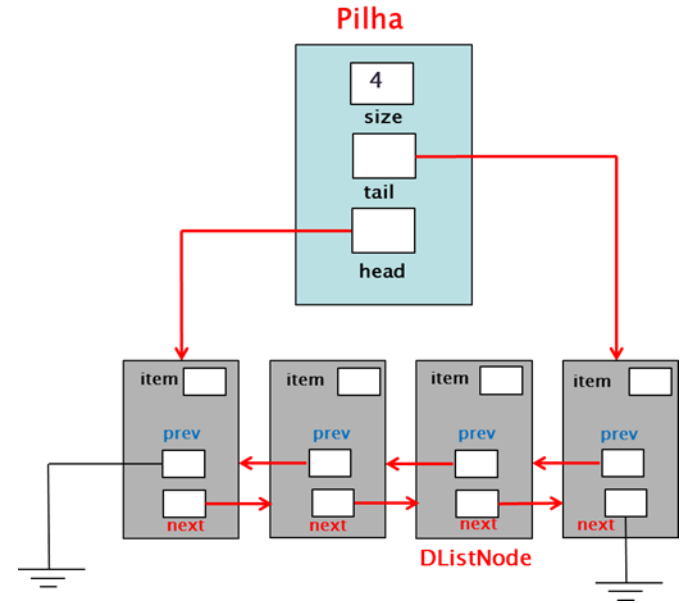
## Implementação Dinâmica de Pilhas e Filas





## Tipo Abstrato de Dados Pilha

- `public Integer size;`
- `public DListNode head;`
- `public DListNode tail;`
- `public void push(Integer item);`
- `public Integer pop();`
- `public void imprimePilha();`





# Classe DListNode

```
package maua;
```

```
public class DListNode {
```

```
    public Integer item;
```

```
    public DListNode next;
```

```
    public DListNode prev;
```

```
    public DListNode() {
```

```
        this.item = 0;  
        this.next = null;  
        this.prev = null;
```

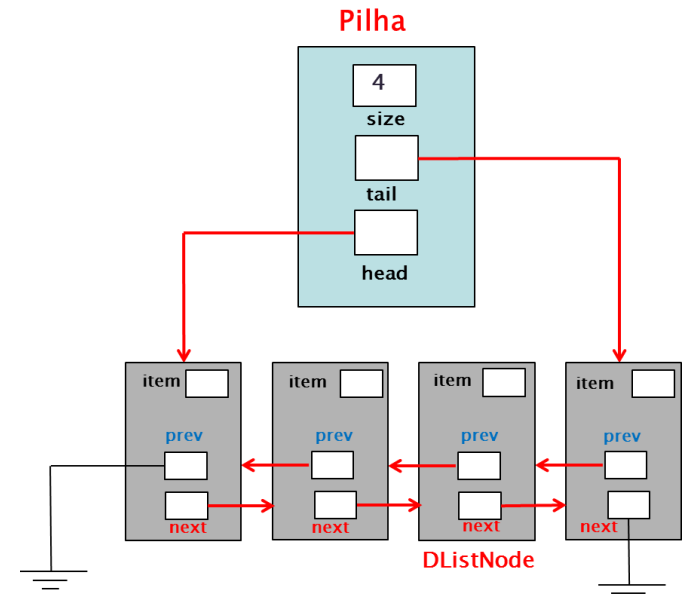
```
    }
```

```
    public DListNode(Integer item) {
```

```
        this.item = item;  
        this.next = null;  
        this.prev = null;
```

```
    }
```

```
}
```







# Classe Pilha

```
package maua;

public class Pilha {

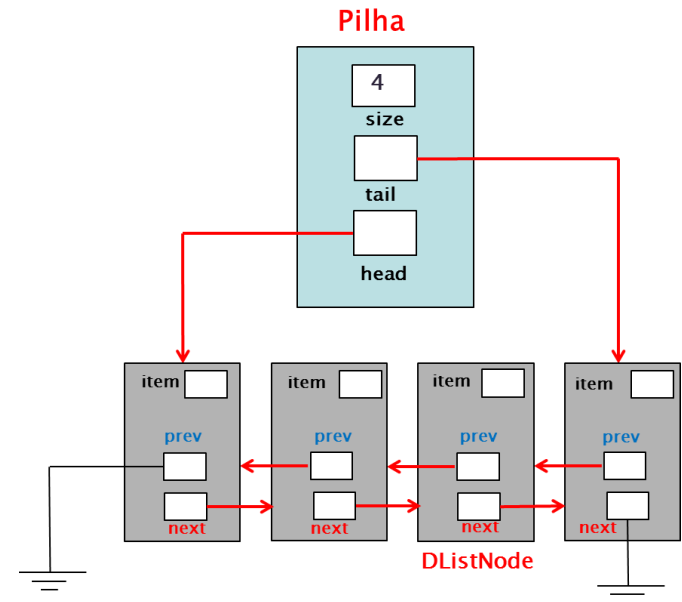
    public int size;
    public DListNode head;
    public DListNode tail;

    public Pilha() {

        this.size = 0;
        this.head = null;
        this.tail = null;

    }

}
```





## Método push(Integer)

```
public void push(Integer item) {  
  
    DListNode novoNode = new DListNode(item);  
  
    if (this.size == 0) {  
        this.head = novoNode;  
        this.tail = novoNode;  
        this.size++;  
    }  
    else {  
        this.head.prev = novoNode;  
        novoNode.next = this.head;  
        this.head = novoNode;  
        this.size++;  
    }  
  
}
```





# Método pop()

```
public Integer pop() {  
    Integer trab;  
    if (this.size == 0) {  
        System.out.println("Stack empty ...");  
        return null;  
    }  
    else {  
        if (this.size == 1) {  
            trab = this.head.item;  
            this.head = null;  
            this.tail = null;  
            this.size = 0;  
            return trab;  
        }  
        else {  
            trab = this.head.item;  
            this.head = this.head.next;  
            this.head.prev = null;  
            this.size--;  
            return trab;  
        }  
    }  
}
```

}





## Método `imprimePilha()`

```
public void imprimePilha() {  
    DListNode p;  
  
    p = this.head;  
  
    if (this.size == 0 )  
        System.out.println("Stack empty...");  
    else {  
        System.out.print( "(" );  
        while ( p != null ) {  
            System.out.print ( p.item + " ");  
            p = p.next;  
        }  
        System.out.print(")" + "\n");  
    }  
}
```





# Execução

```
package maua;

public class TestPilha {

    public static void main(String[] args) {

        Pilha x = new Pilha();
        x.imprimePilha();

        for(int i = 0; i < 20 ; i++ ) {
            x.push(i);
            x.imprimePilha();
        }
        int tamanho = x.size;

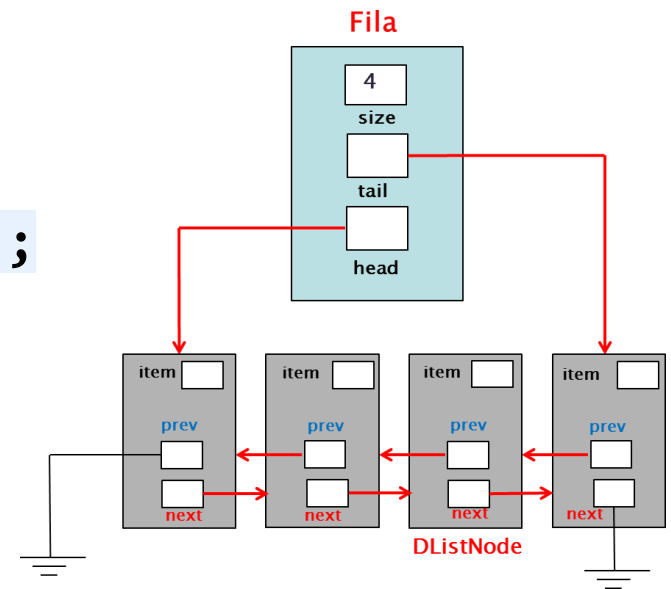
        x.imprimePilha();
        for (int i = 0; i < tamanho ; i++ ) {
            System.out.println("Valor retirado: " + x.pop());
            x.imprimePilha();
        }
        x.imprimePilha();
    }
}
```





# Tipo Abstrato de Dados Fila

- `public Integer size;`
- `public DListNode head;`
- `public DListNode tail;`
- `public void enqueue(Integer item);`
- `public Integer dequeue();`
- `public void imprimeFila();`





# Classe DListNode

```
package maua;
```

```
public class DListNode {
```

```
    public Integer item;
```

```
    public DListNode next;
```

```
    public DListNode prev;
```

```
    public DListNode() {
```

```
        this.item = 0;
```

```
        this.next = null;
```

```
        this.prev = null;
```

```
    }
```

```
    public DListNode(Integer item) {
```

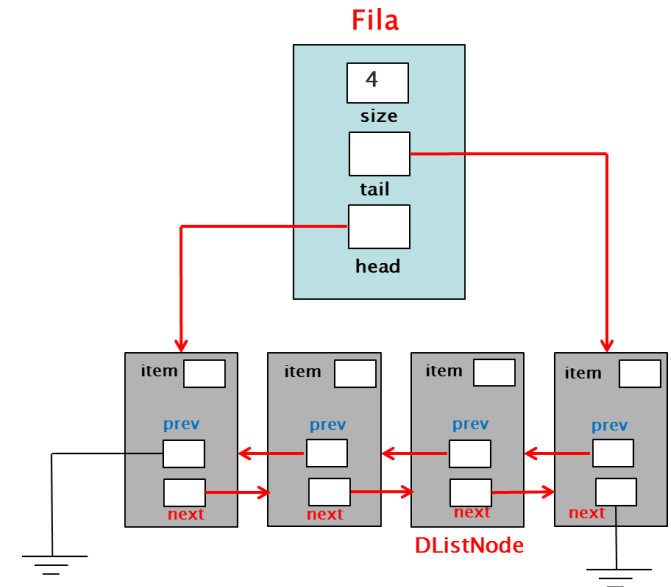
```
        this.item = item;
```

```
        this.next = null;
```

```
        this.prev = null;
```

```
    }
```

```
}
```



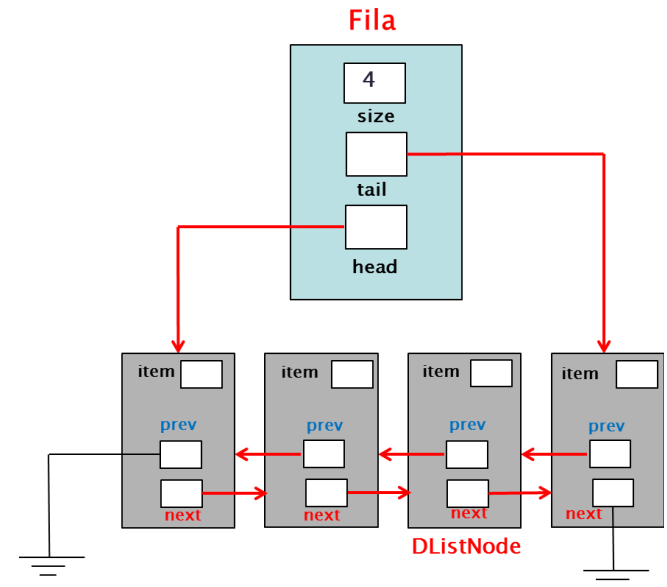


# Classe Fila

```
package maua;
```

```
public class Fila {  
  
    public int size;  
    public DListNode head;  
    public DListNode tail;
```

```
public Fila() {  
  
    this.size = 0;  
    this.head = null;  
    this.tail = null;  
  
}
```

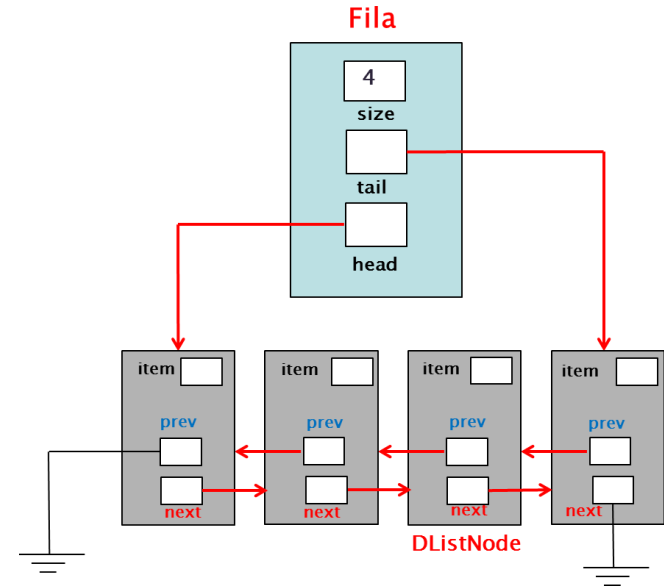






# Método enqueue(Integer)

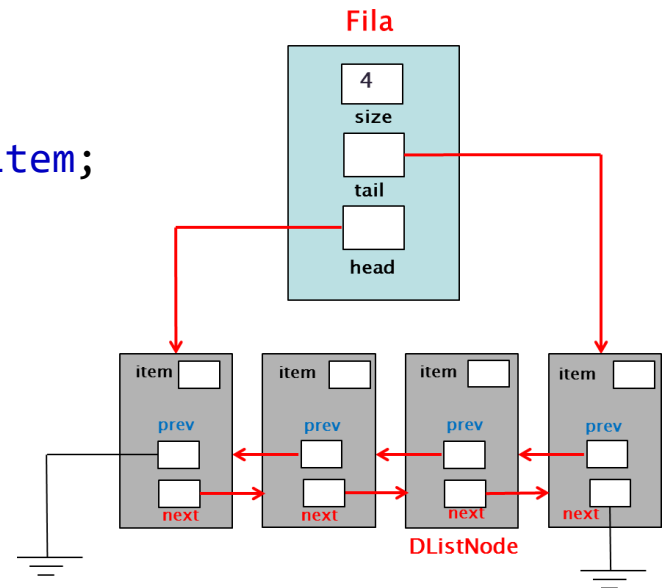
```
public void enqueue(Integer item) {  
  
    DListNode novoNode = new  
    DListNode(item);  
  
    if (this.size == 0) {  
        this.head = novoNode;  
        this.tail = novoNode;  
        this.size++;  
    }  
    else {  
        this.tail.next = novoNode;  
        this.tail = novoNode;  
        this.size++;  
    }  
}
```





## Método enqueue(Integer)

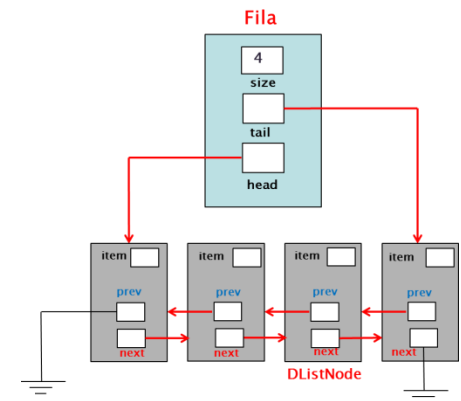
```
public Integer dequeue() {  
    Integer trab;  
    if (this.size == 0) {  
        System.out.println("Stack empty ...");  
        return null;  
    }  
    else {  
        if (this.size == 1) {  
            trab = this.head.item;  
            this.head = null;  
            this.tail = null;  
            this.size = 0;  
            return trab;  
        }  
        else {  
            trab = this.head.item;  
            this.head = this.head.next;  
            this.head.prev = null;  
            this.size--;  
            return trab;  
        }  
    }  
}
```





## Método imprimeFila(Integer)

```
public void imprimeFila() {  
    DListNode p;  
  
    p = this.head;  
  
    if (this.size == 0 )  
        System.out.println("Stack empty...");  
    else {  
        System.out.print( "(" );  
        while ( p != null ) {  
            System.out.print ( p.item + " ");  
            p = p.next;  
        }  
        System.out.print(")" + "\n");  
    }  
}
```





```
package maua;
```

## Execução

```
public class TestFila {
```

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

```
        Fila x = new Fila();
```

```
        x.imprimeFila();
```

```
        for(int i = 0; i < 20 ; i++ ) {  
            x.enqueue(i);  
            x.imprimeFila();  
        }
```

```
        int tamanho = x.size;
```

```
        x.imprimeFila();
```

```
        for (int i = 0; i < tamanho ; i++ ) {  
            System.out.println("Valor retirado: " + x.dequeue());  
            x.imprimeFila();  
        }  
        x.imprimeFila();
```

```
    }
```

```
}
```

