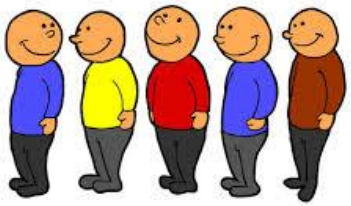
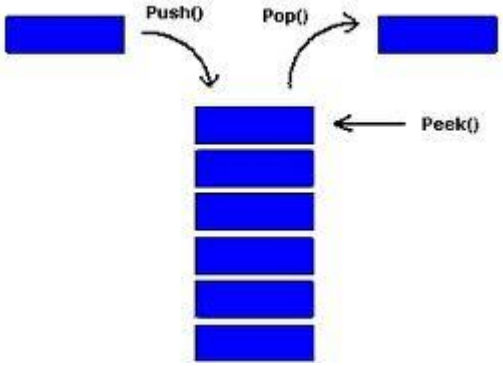


Algoritmos e Estrutura de Dados - IV

Unidade 2 – Revisão - Pilhas e Filas



Prof. Aparecido V. de Freitas
Doutor em Engenharia
da Computação pela EPUSP

Bibliografia

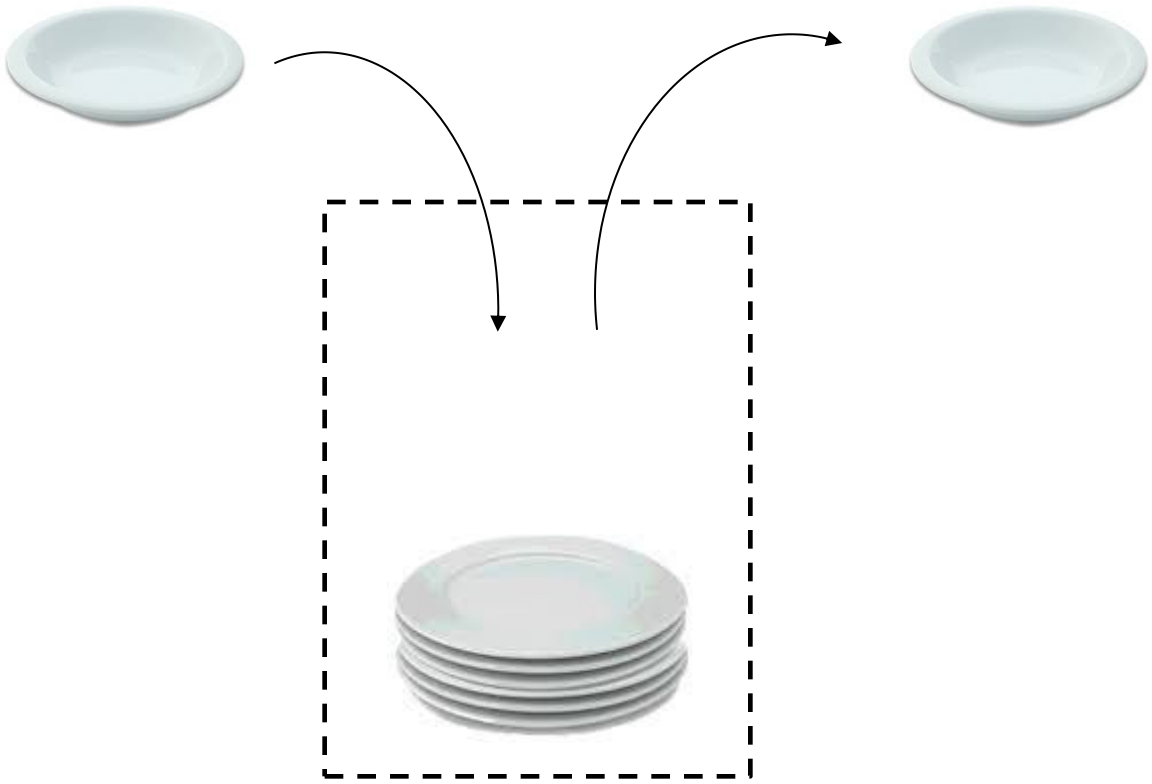
- **Head First Java, 2nd Edition by Kathy Sierra and Bert Bates**
- Core Java Fundamentals – Horstmann / Cornell – PTR- Volumes 1 e 2 – 8th Edition
- Java How to Program - 9th Edition by Paul Deitel and Harvey Deitel
- Beginning Java 2 – Ivor Horton – 1999 WROX
- Java2 – The Complete Reference – 7th Edition – Herbert Schildt – Oracle Press
- Inside the Java 2 – Virtual Machine Venners – McGrawHill
- Understanding Object-Oriented Programming with JAVA – Timothy Budd – Addison Wesley
- Effective Java, 2nd Edition by Joshua Bloch
- Thinking in Java (4th Edition) by Bruce Eckel

Bibliografia

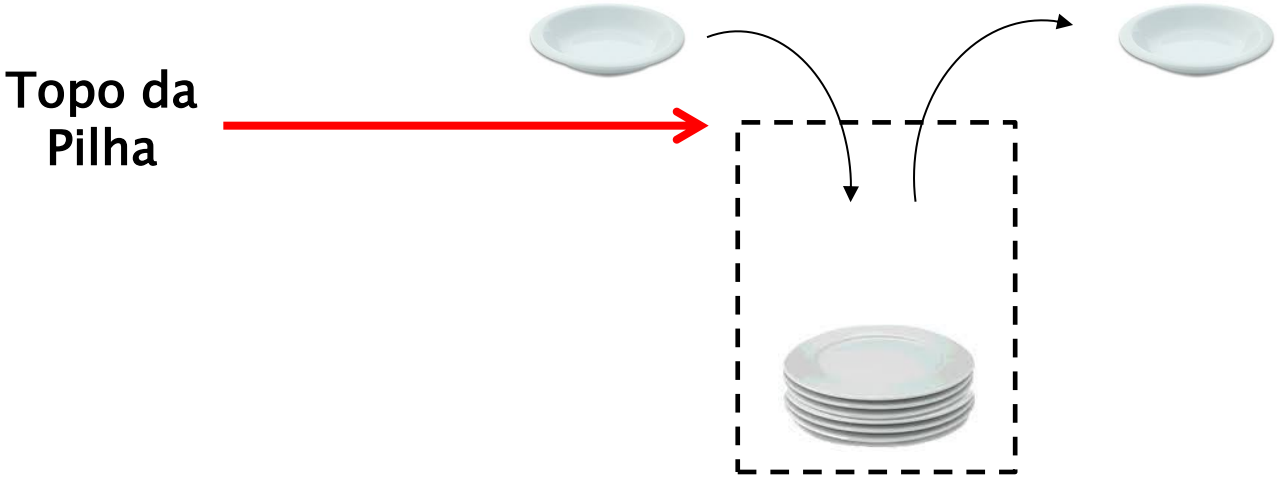
- **Data Structures and Algorithms in Java, Fifth Edition, John Wiley & Sons, Michael Goodrich, Roberto Tamassia, 2010**
- Estrutura de Dados e Algoritmos – Bruno R. Preiss, Editora Campus, 2001
- Estrutura de Dados e Algoritmos em Java – Robert Lafore, Editora Ciência Moderna, 2005
- Algoritmos e Estrutura de Dados – Niklaus Wirth – Editora Prentice Hall do Brasil, 1989

Pilha

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

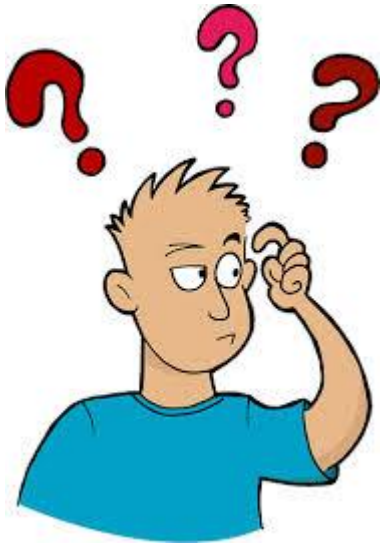


Pilha

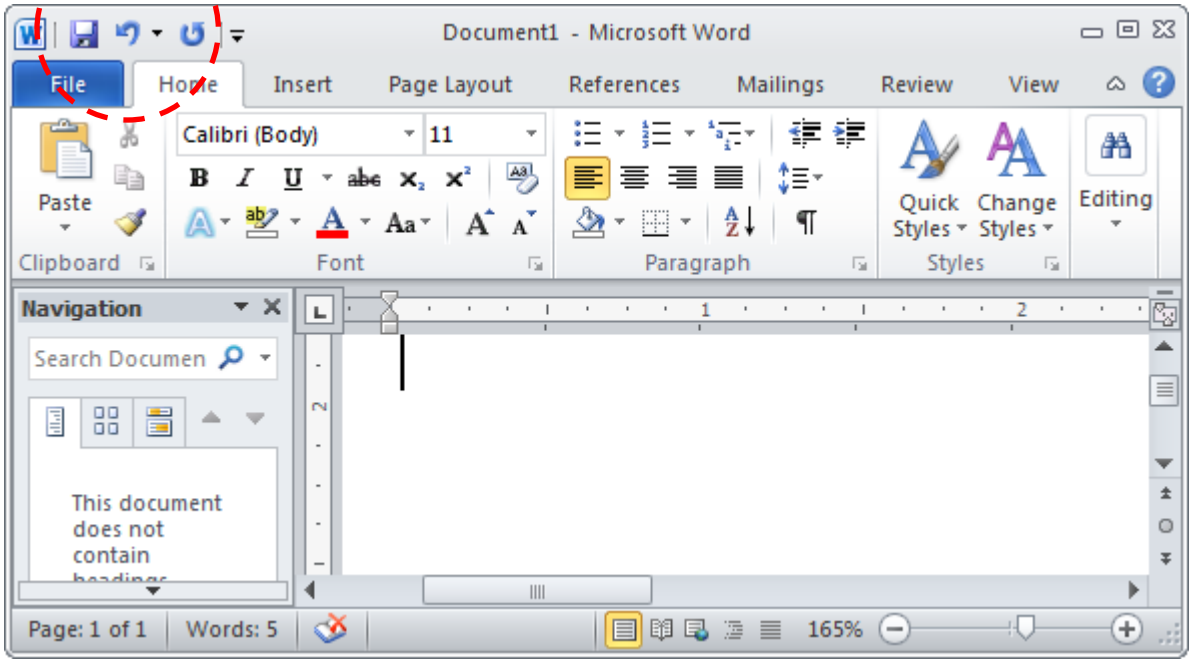


- Estrutura de Dados do tipo LIFO
- Inserção de dados: Sempre no Topo da Pilha
- Remoção de dados: Sempre no Topo da Pilha

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.

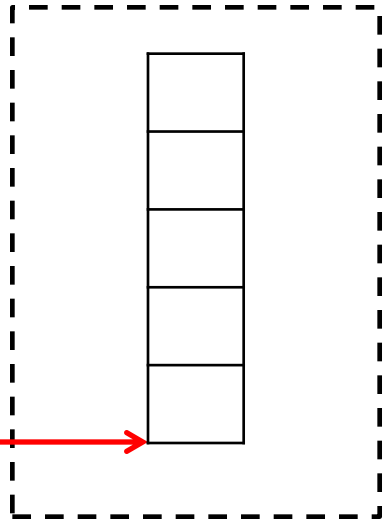
Pilhas - Aninhamento

```
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;
              }
}
```



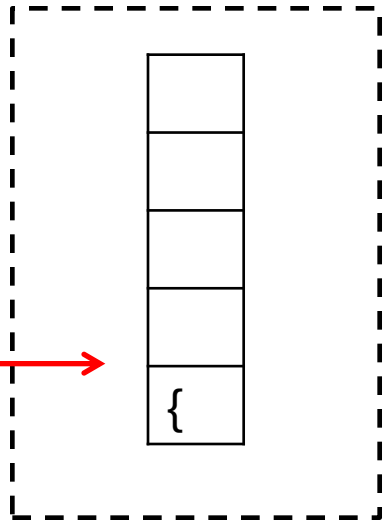
Topo da Pilha

Pilha Vazia

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

Pilhas – Aninhamento

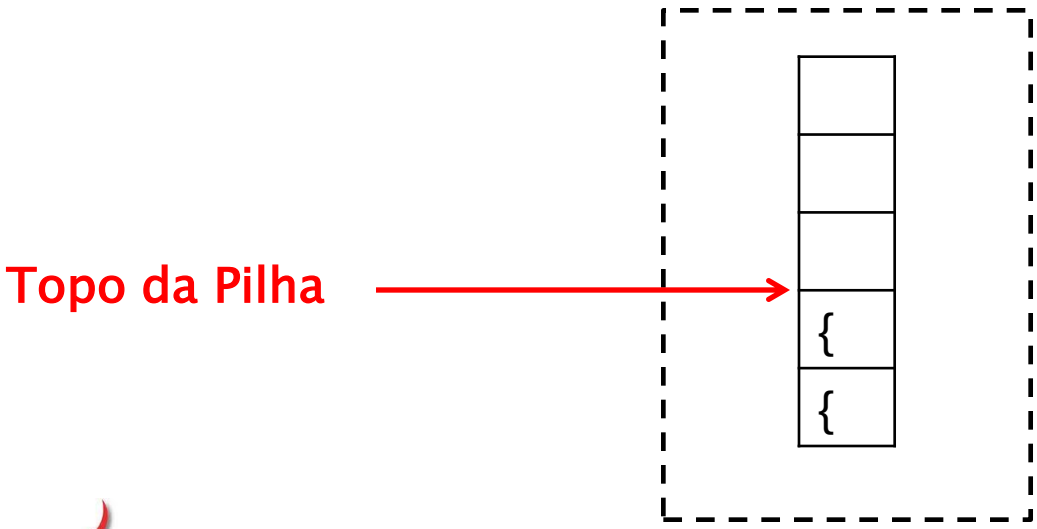
```
public static void main (String[] args) {
        int i = 0;

        while ( i < 10 ) {
                System.println(i);
                i = i + 1;
              }
}
```



Topo da Pilha

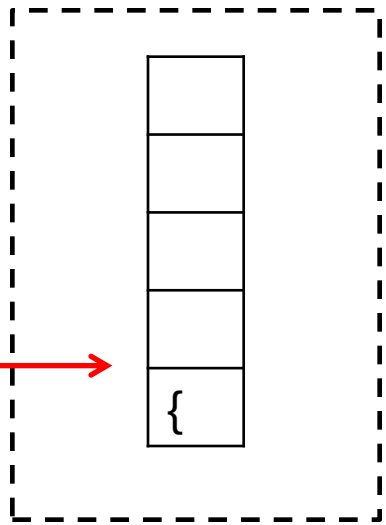
Pilhas – Aninhamento

```
public static void main (String[] args {
        int i = 0;

        while ( i < 10 ) {
                System.println(i);
                i = i + 1;
              }
}
```



Pilhas – Aninhamento

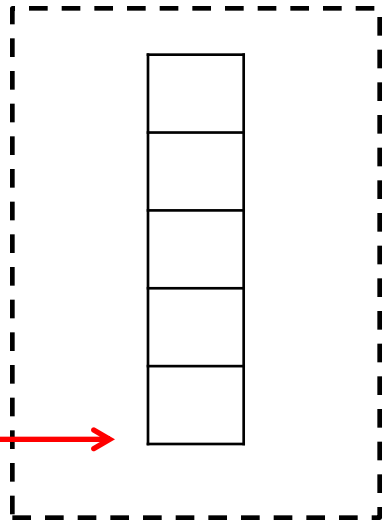
```
public static void main (String[] args {
        int i = 0;

        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;
        }
}
```



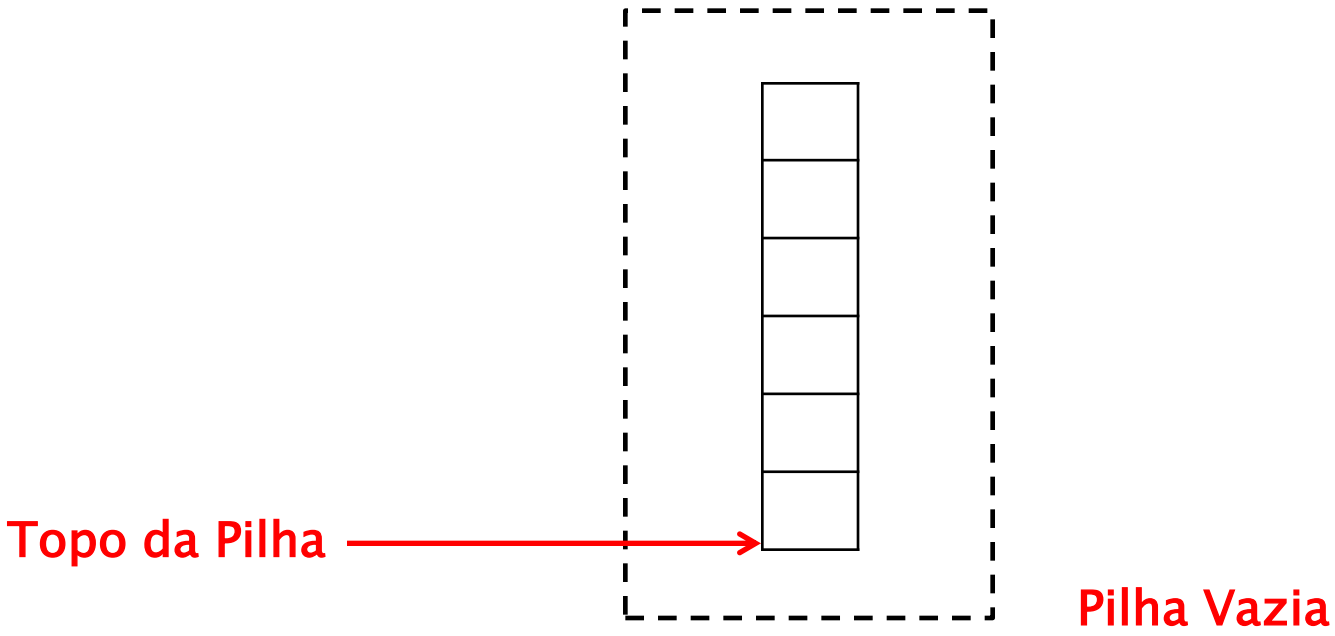
Topo da Pilha

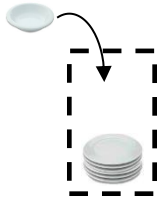
Pilha Vazia

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

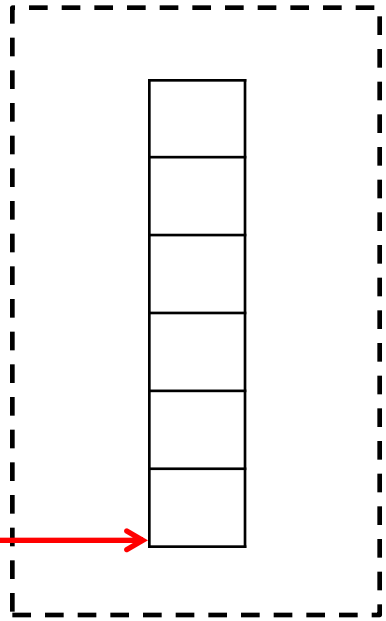




Empilhamento

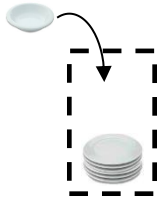
1

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



Topo da Pilha

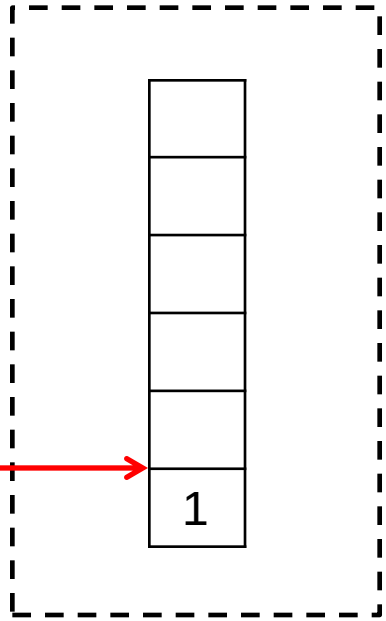
Pilha Vazia



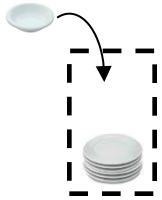
Empilhamento

2

- Operação de Empilhamento da lista: ,2,3,4,5,6



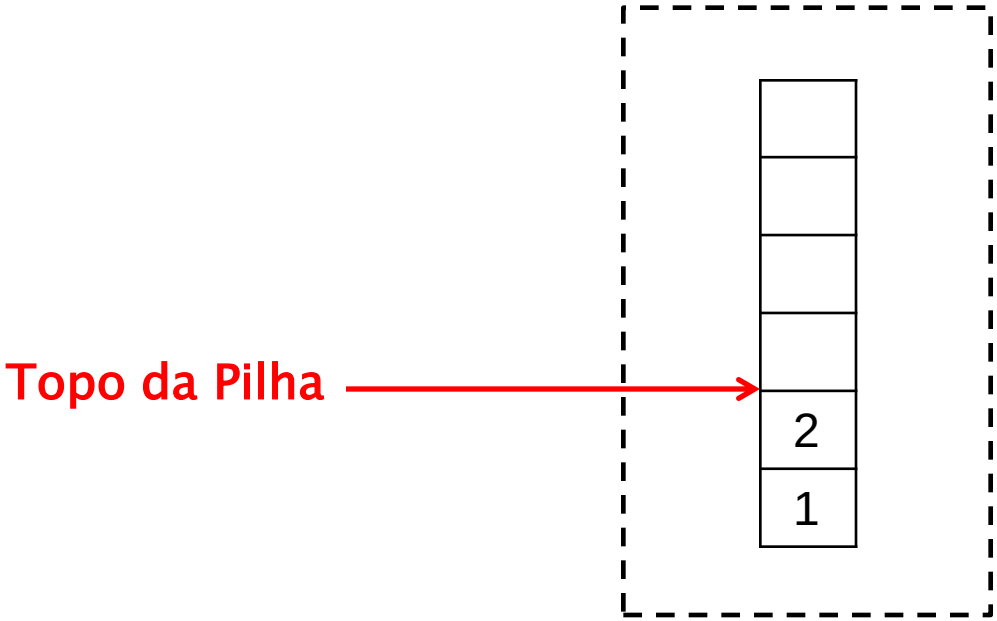
Topo da Pilha

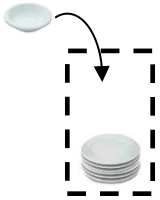


Empilhamento

3

- Operação de Empilhamento da lista: ,3,4,5,6



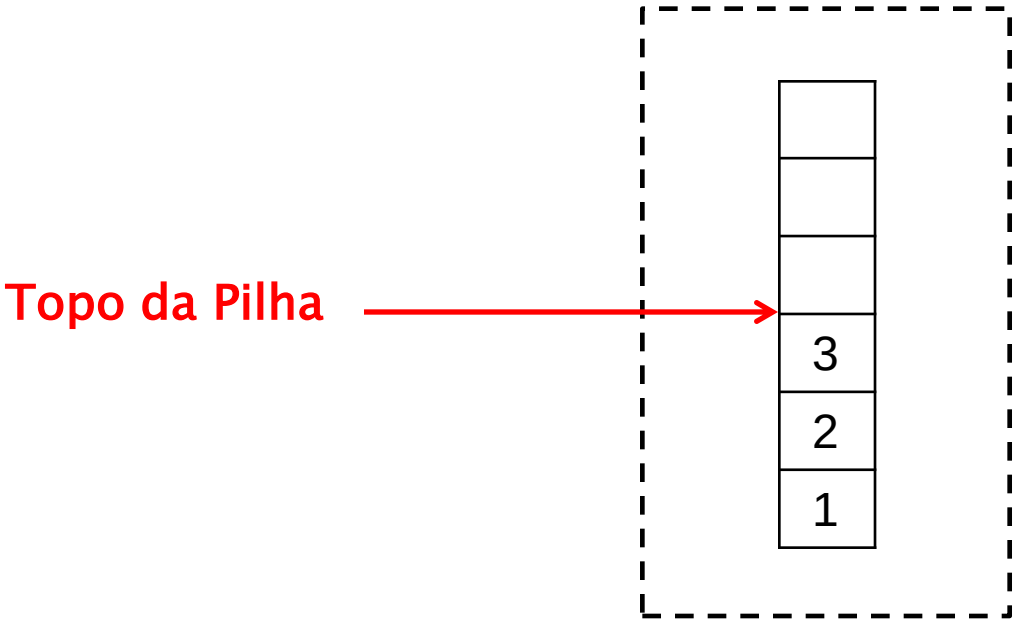


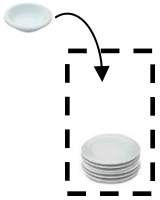
Empilhamento

4

- Operação de Empilhamento da lista:

,4,5,6



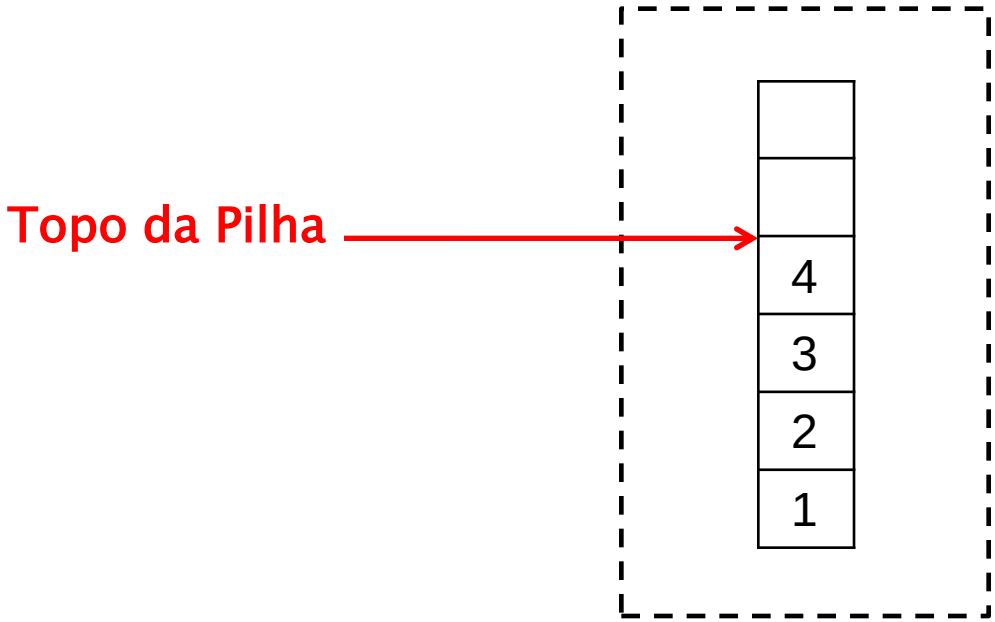


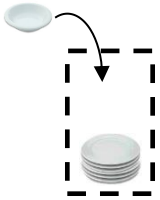
Empilhamento

5

- Operação de Empilhamento da lista:

,5,6



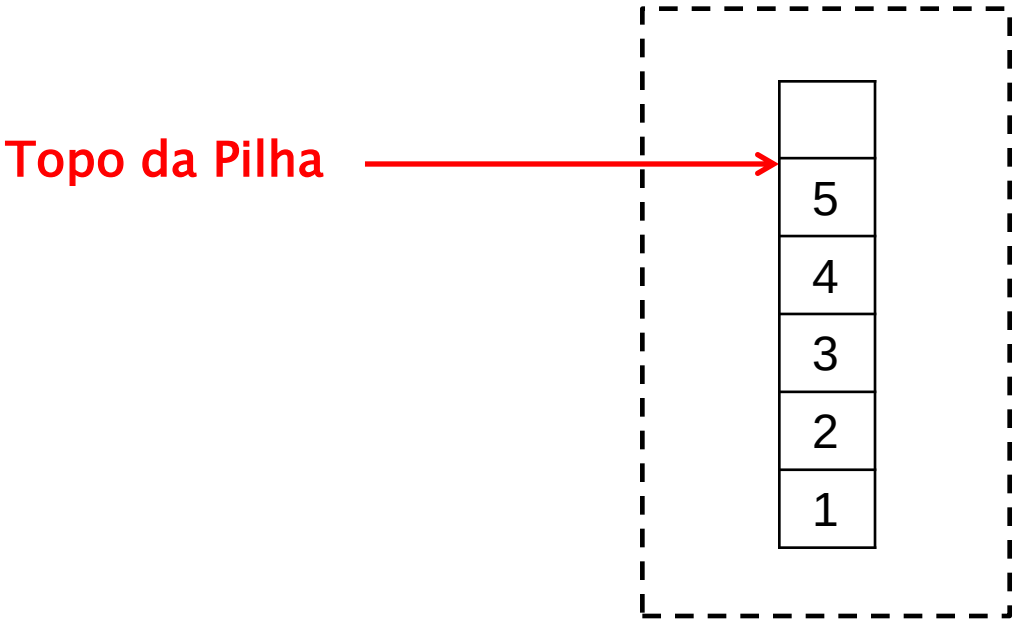


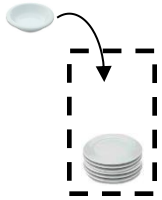
Empilhamento

6

- Operação de Empilhamento da lista:

,6



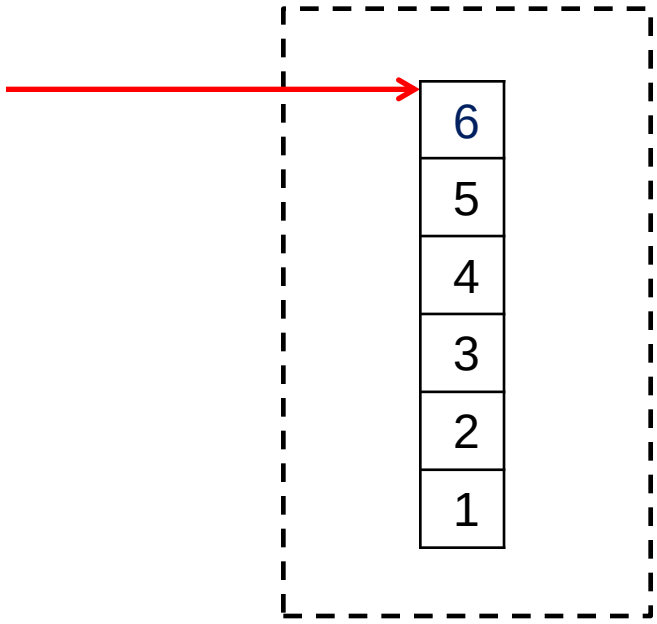


Empilhamento

7

- Operação de Empilhamento concluída:

Topo da Pilha

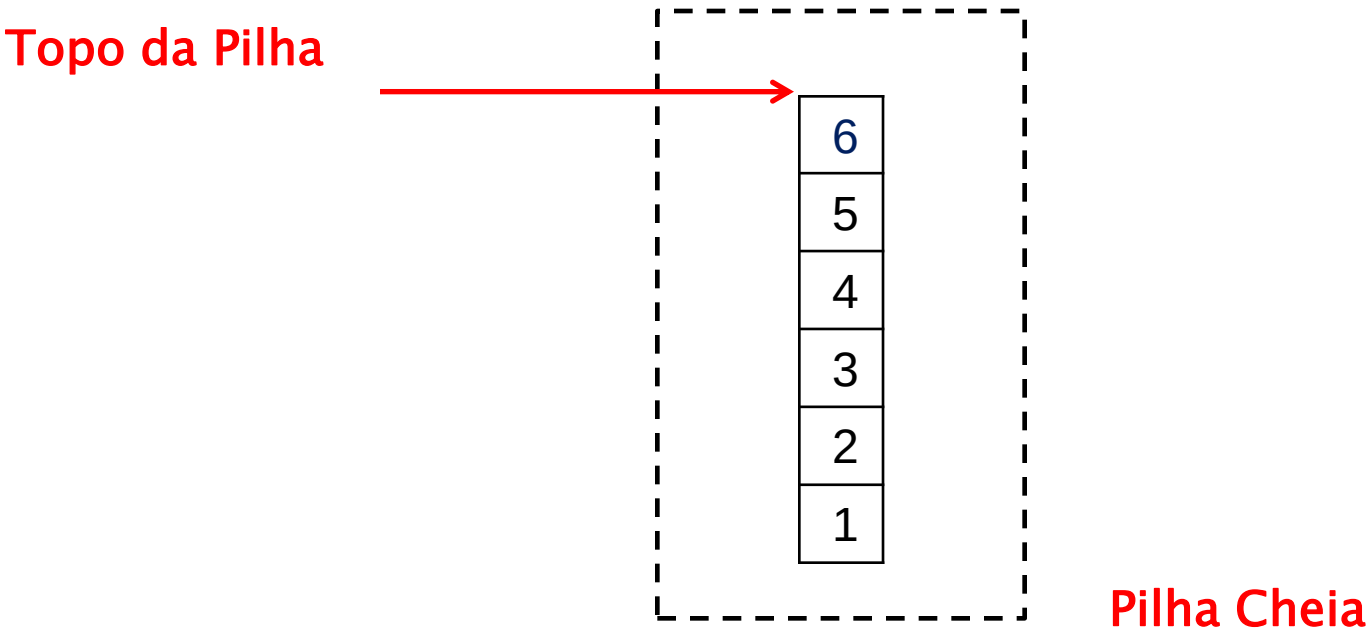


Pilha Cheia

Desempilhamento

1

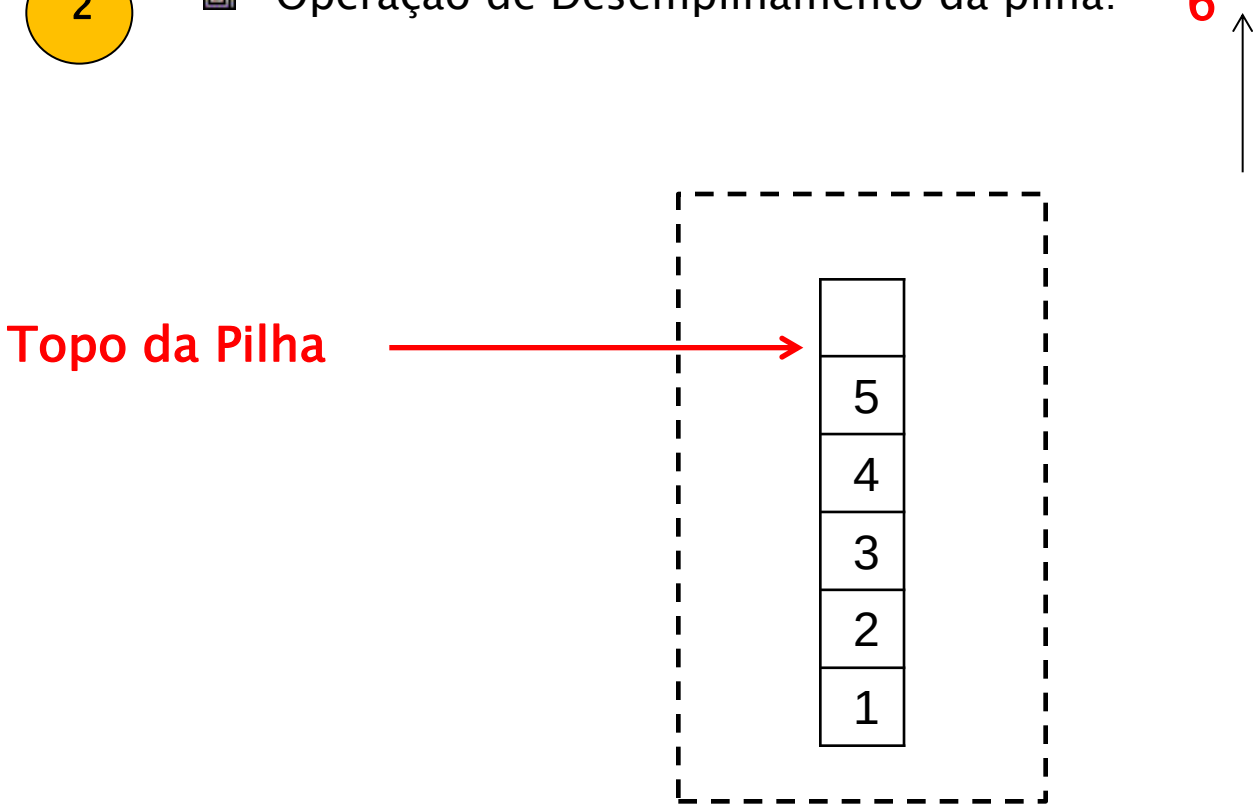
- Operação de Desempilhamento da pilha:



Desempilhamento

2

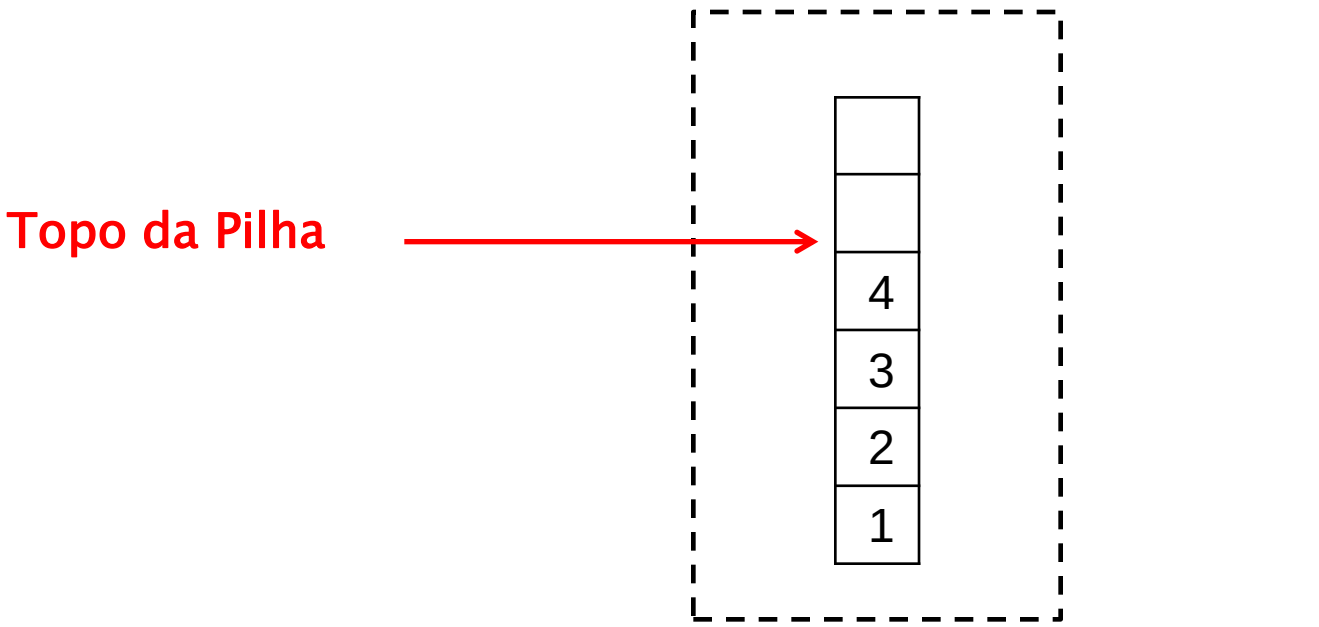
- Operação de Desempilhamento da pilha:



Desempilhamento

3

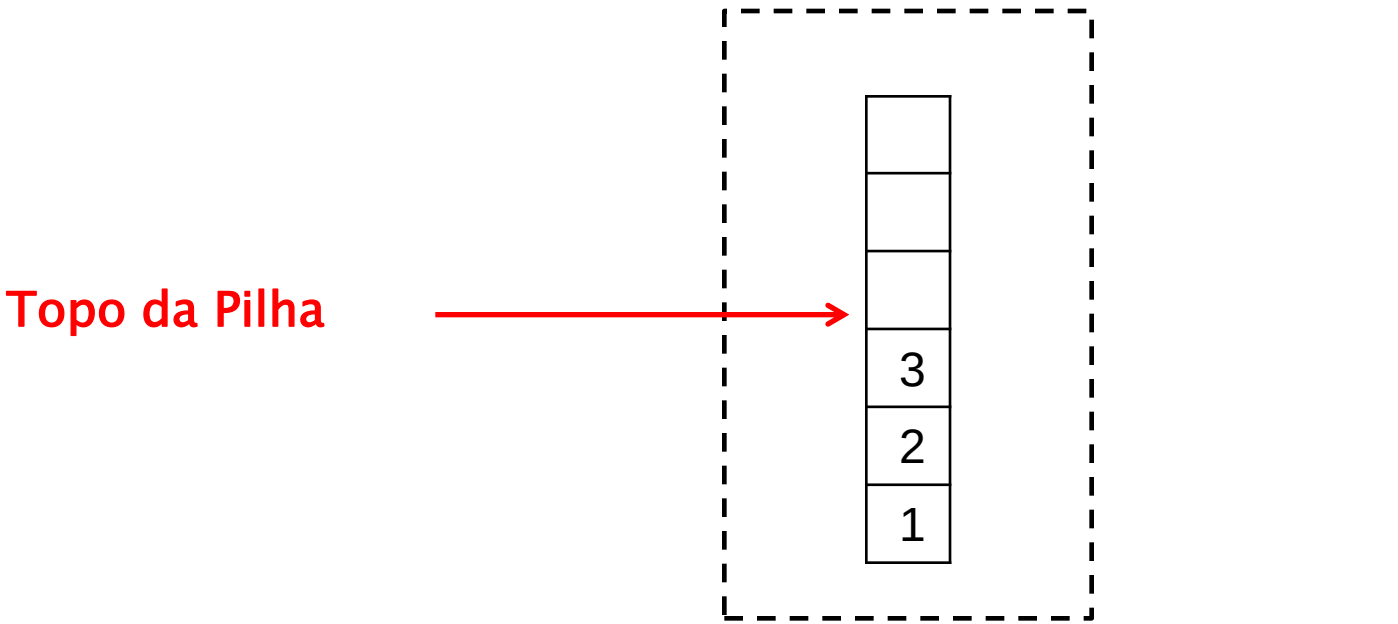
- Operação de Desempilhamento da pilha: 6,5



Desempilhamento

4

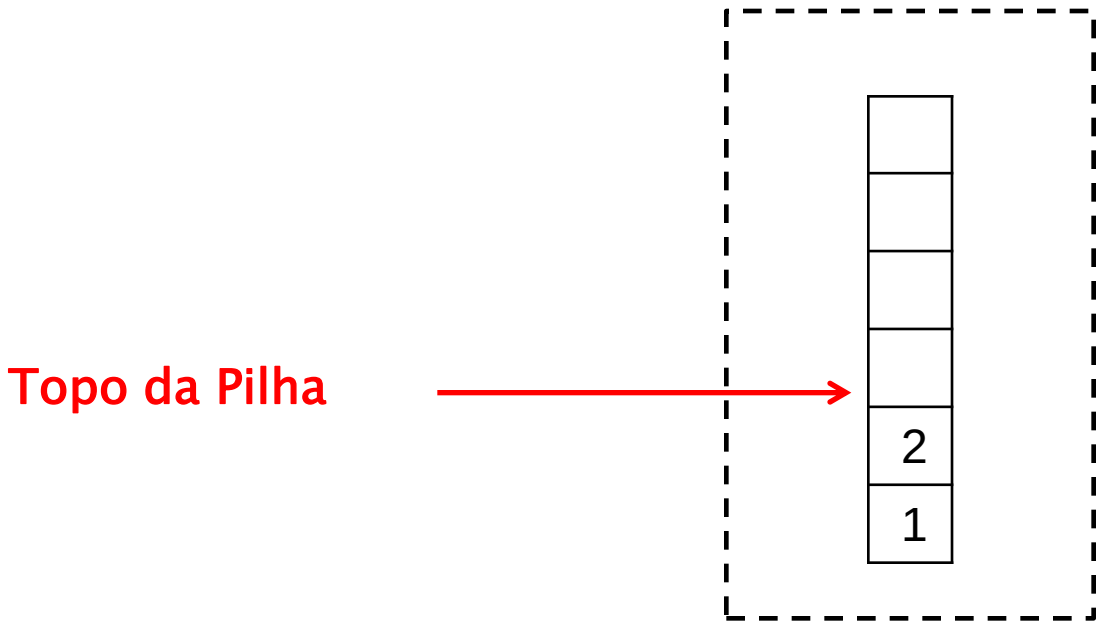
- Operação de Desempilhamento da pilha: 6,5,4



Desempilhamento

5

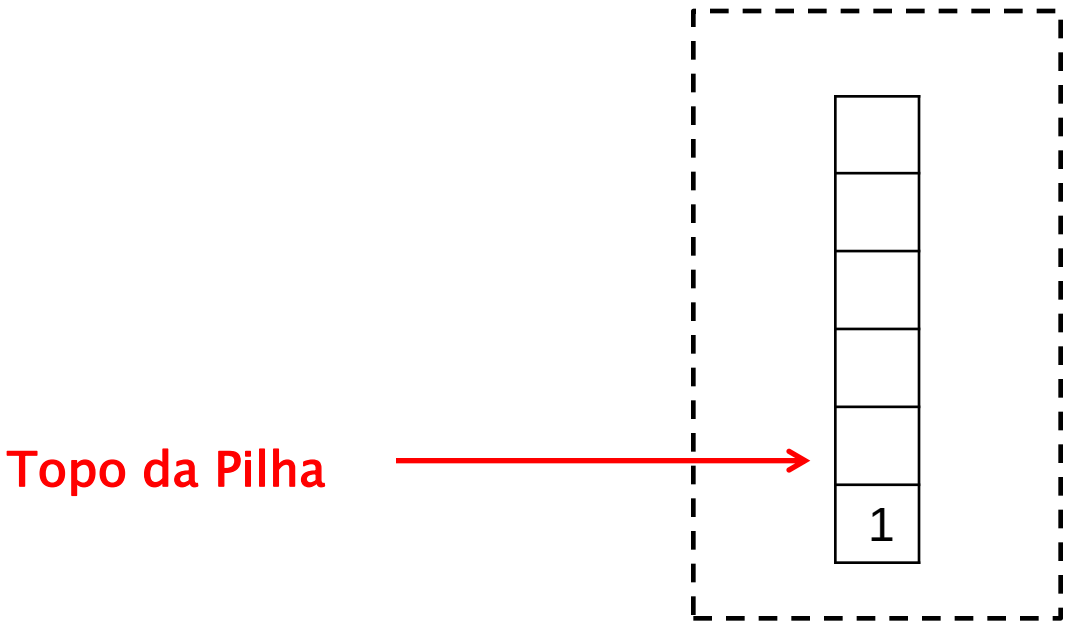
- Operação de Desempilhamento da pilha: 6,5,4, 3



Desempilhamento

6

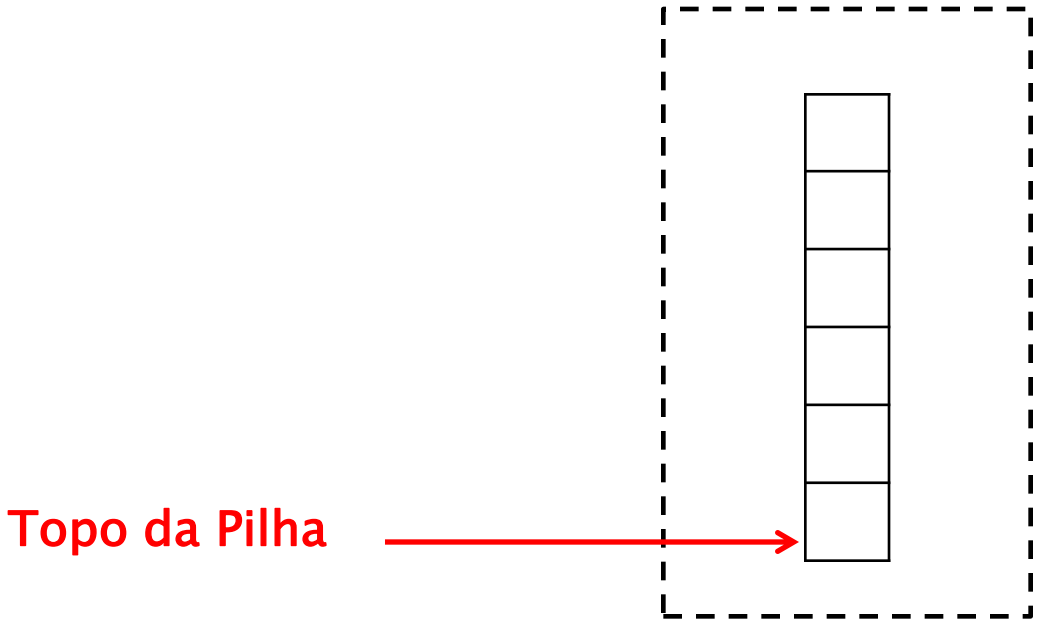
- Operação de Desempilhamento da pilha: 6,5,4, 3,2



Desempilhamento

7

- Operação de Desempilhamento da pilha: 6,5,4, 3,2,1



Aplicação - Pilha

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

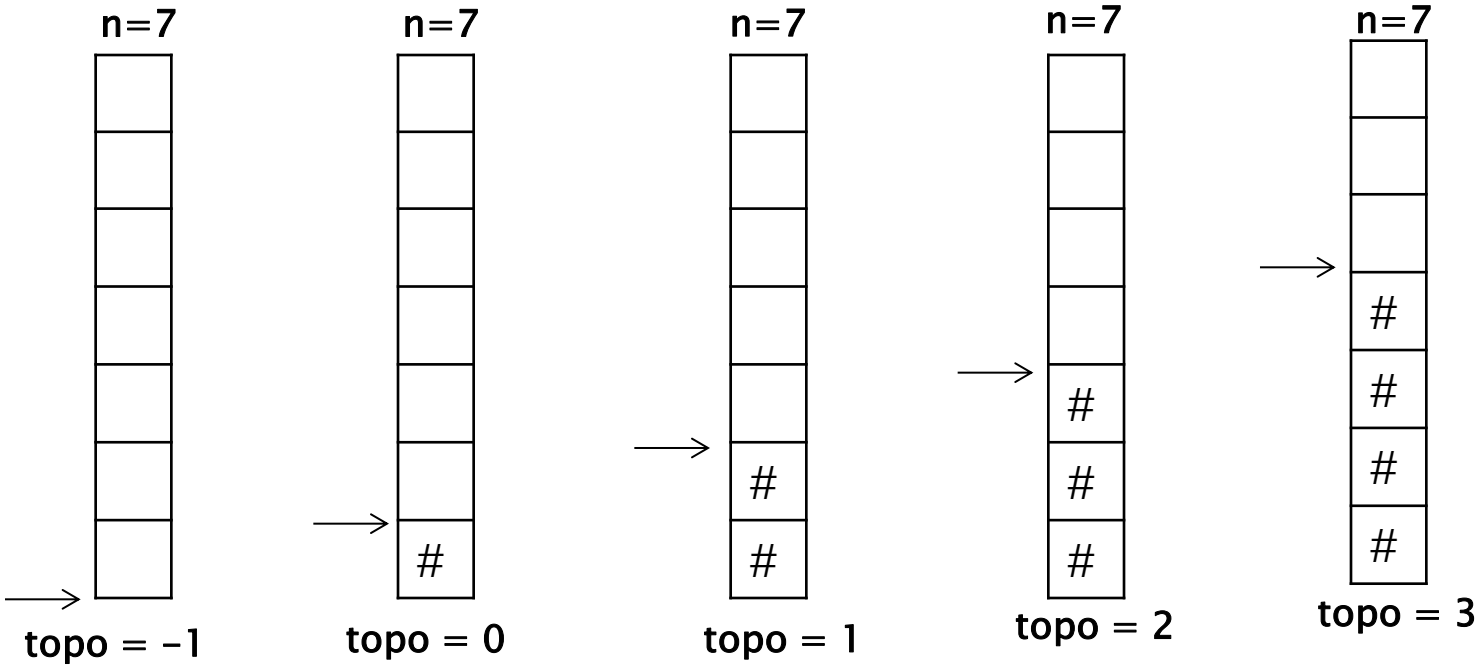
Portanto, os elementos da lista foram invertidos !

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.

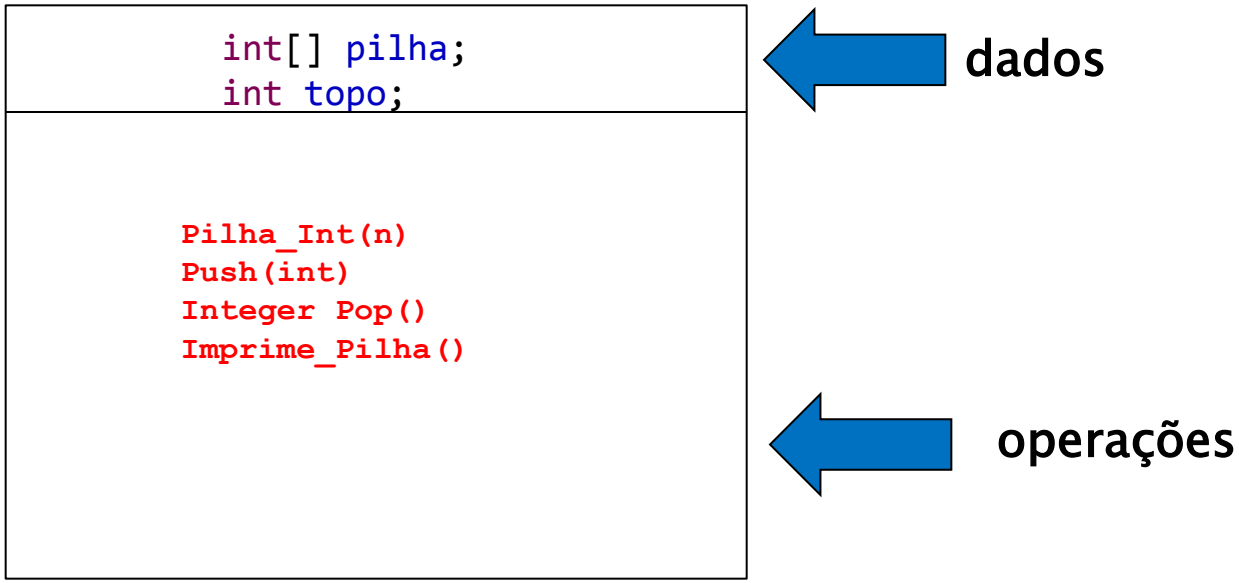
Implementação de Pilhas com arrays

- 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 uscs;

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!
```

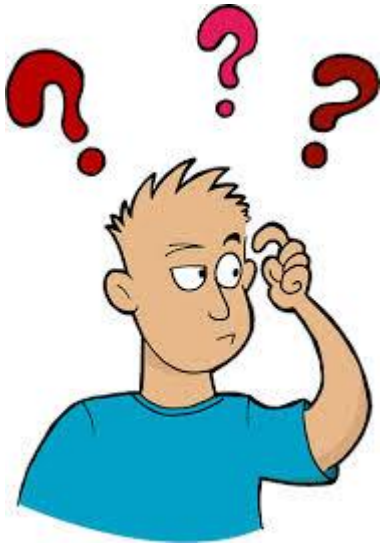
Fila

- Estrutura de Dados que implementa uma lista **FIFO** (First In, First Out).

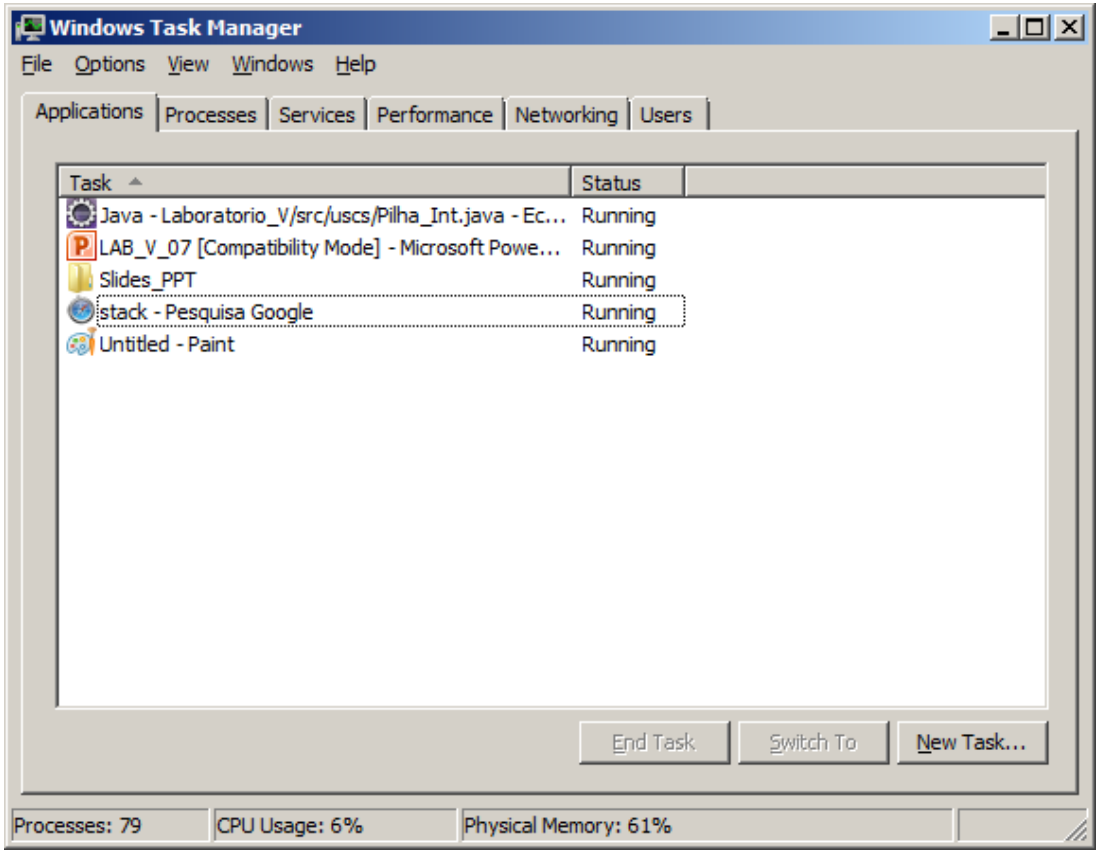
Fila

- 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 ?

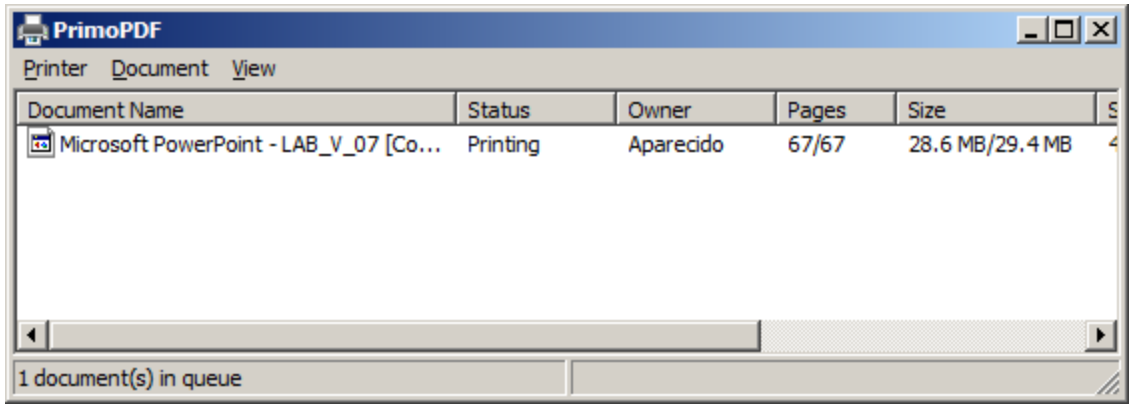


Fila – Exemplo de Aplicação



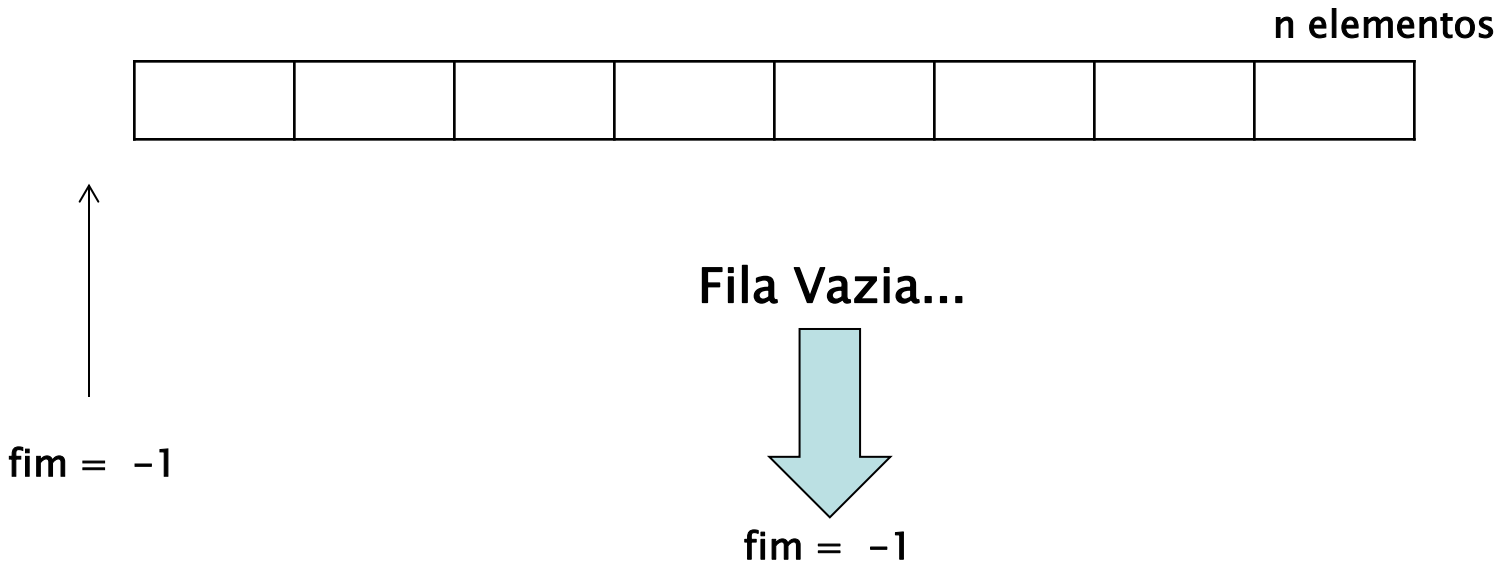
- Fila de Processos

Fila – Exemplo de Aplicação

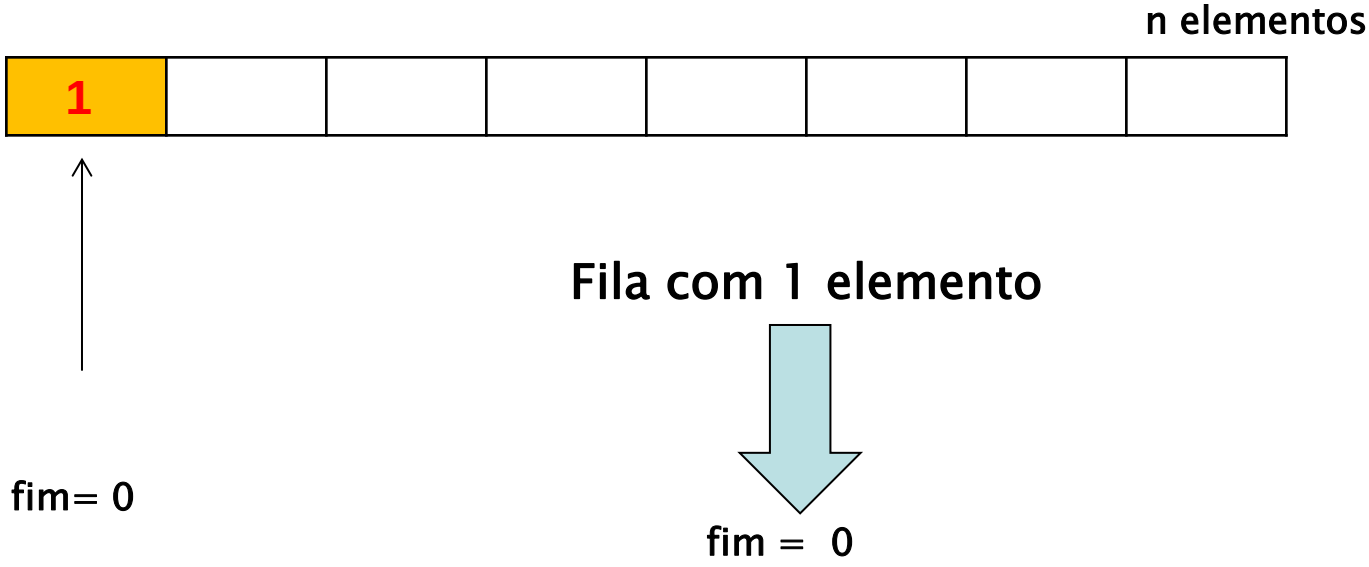


- Fila de Impressão

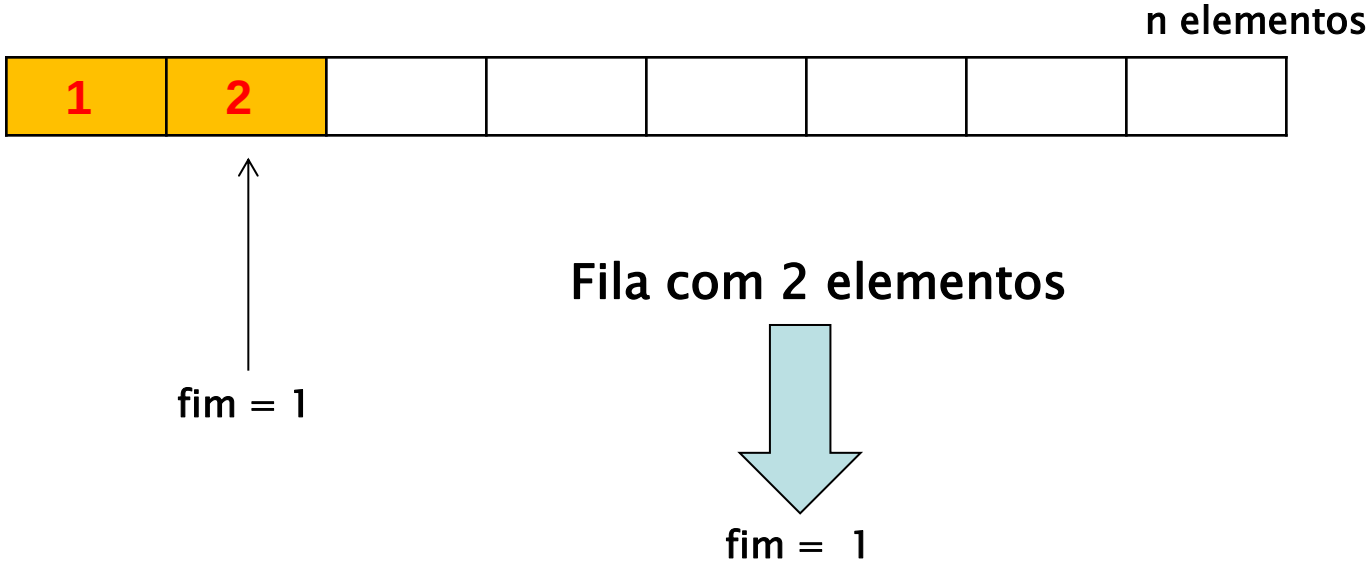
Implementação de Filas com Arrays



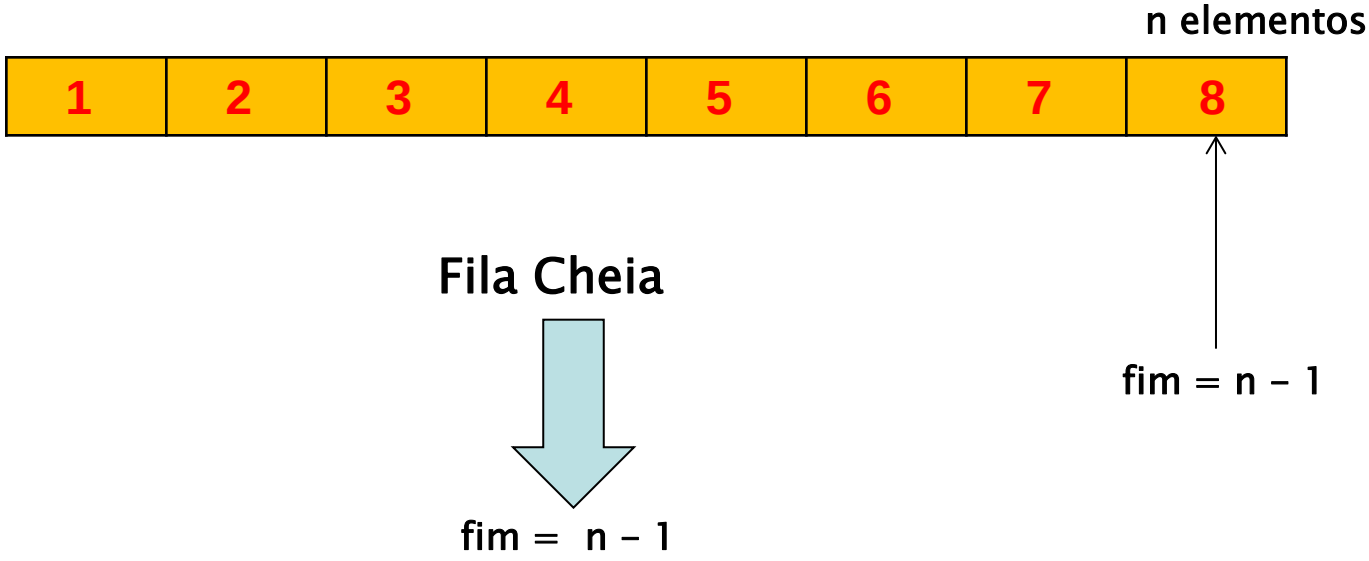
Inserção na Fila



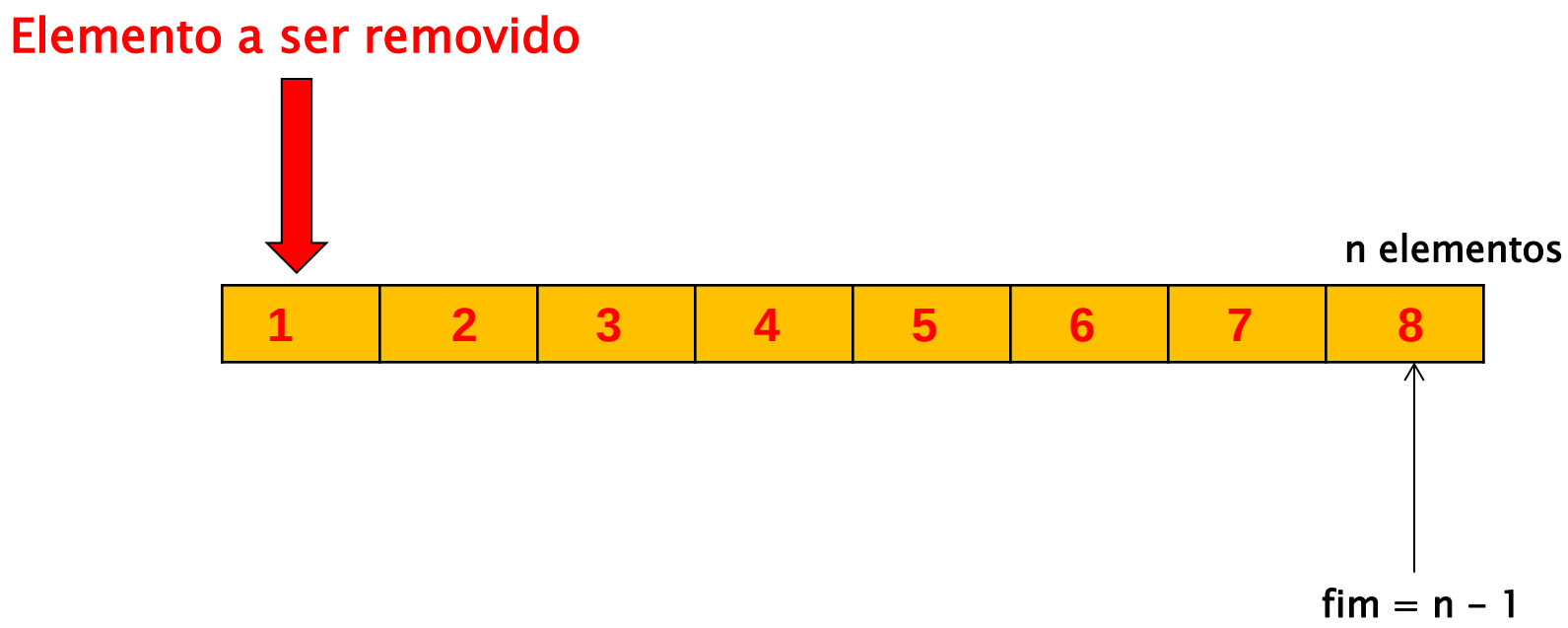
Inserção na Fila



Inserção na Fila

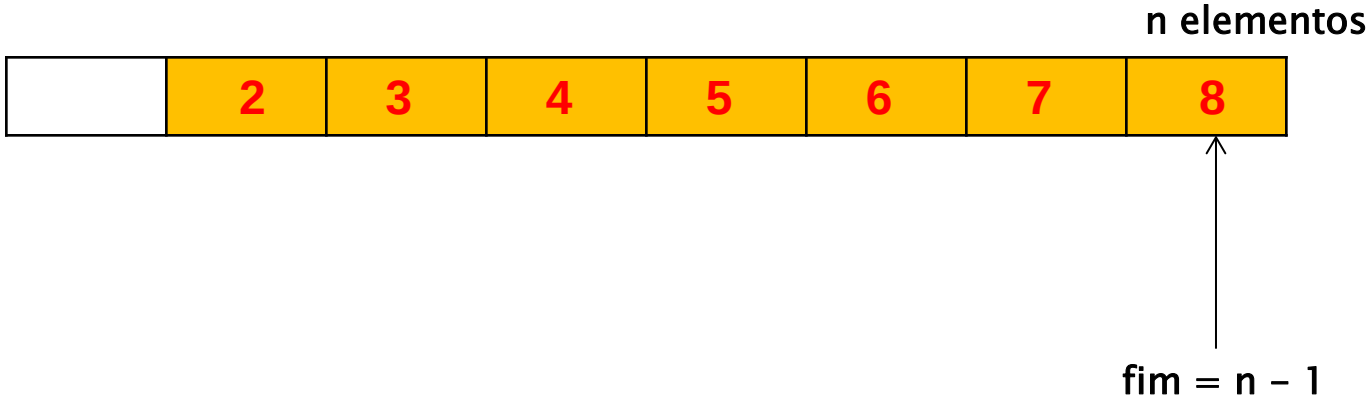


Remoção da Fila



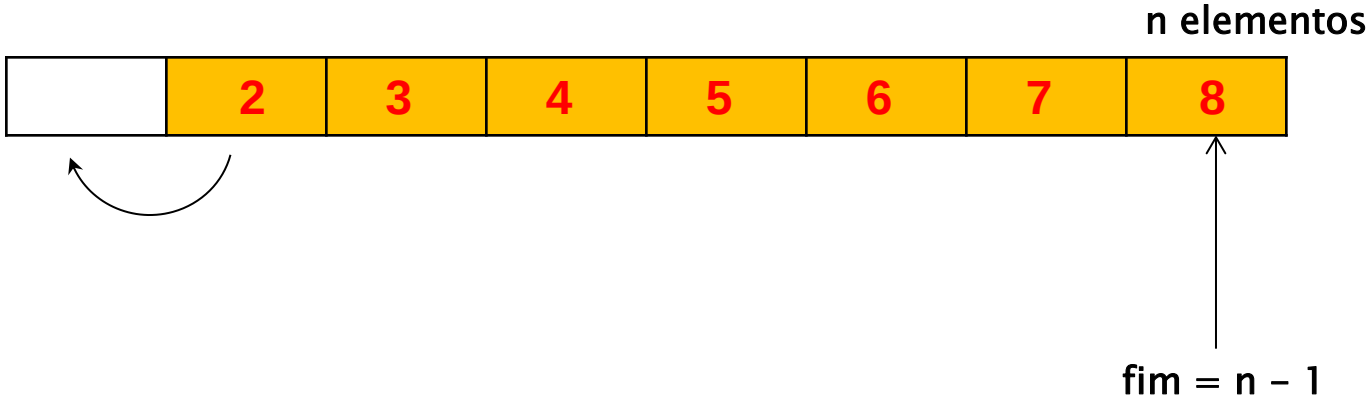
Remoção da Fila

A fila anda !!!



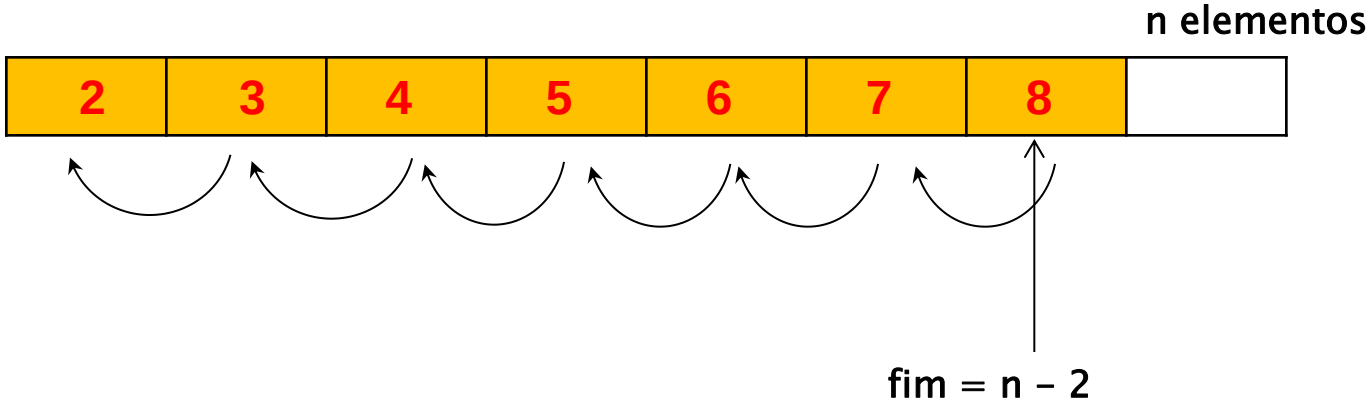
Remoção da Fila

A fila anda !!!



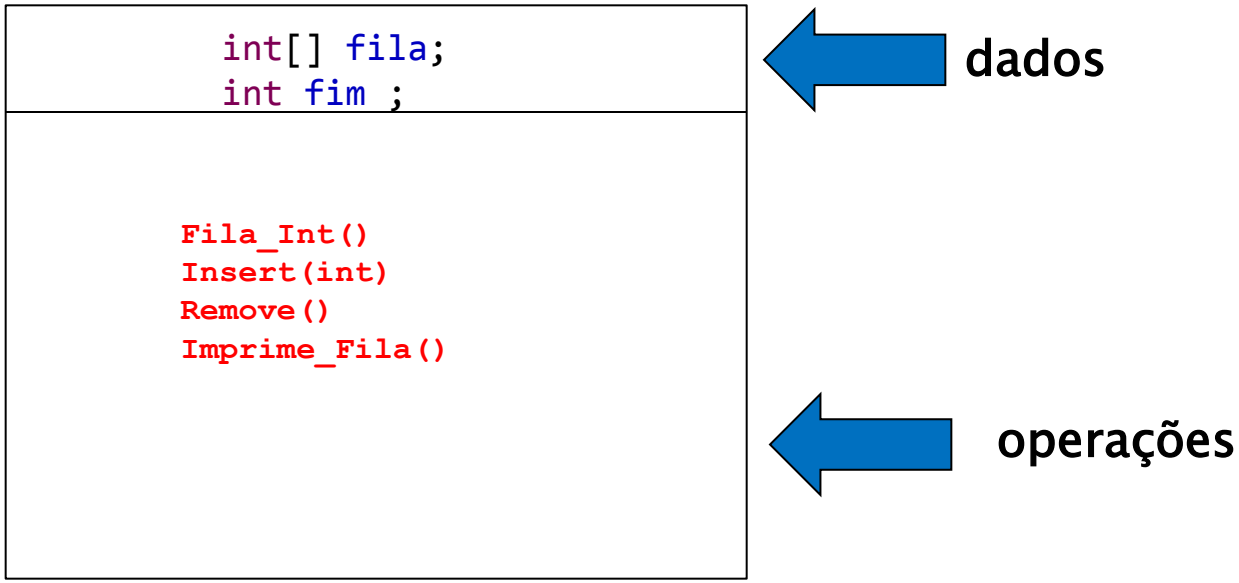
Remoção da Fila

A fila anda !!!



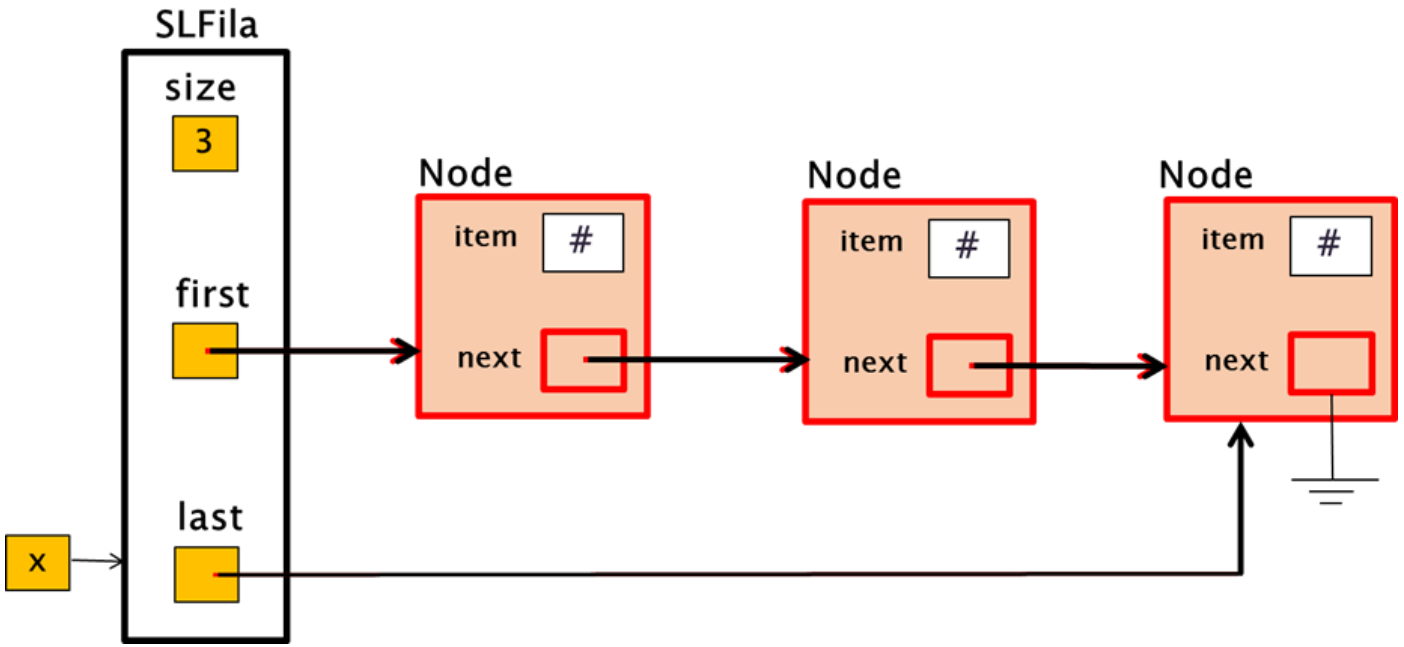
Implementação de Filas com arrays

Tipo Abstrato de Dados: Fila_Int

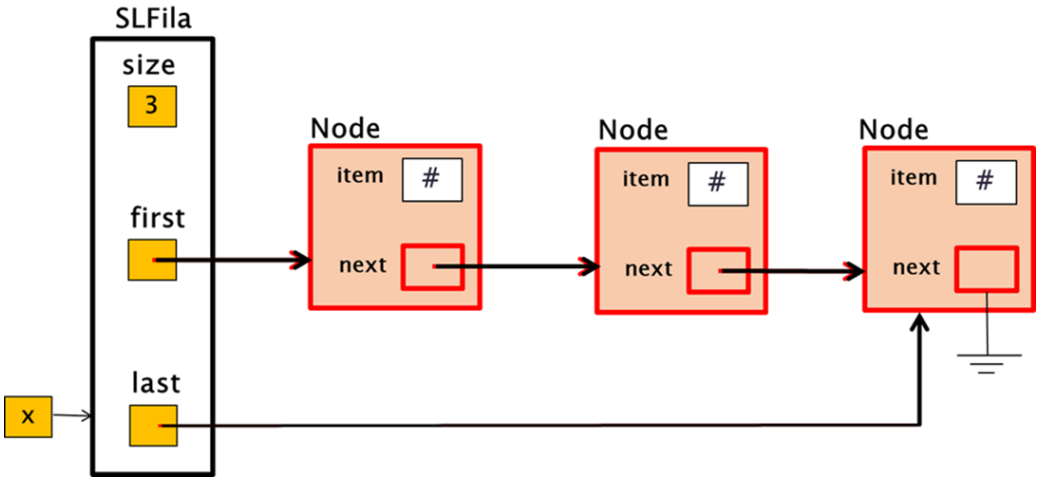


Implementação de Filas com Listas Ligadas

Tipo Abstrato de Dados: SLFila



Tipo Abstrato de Dados: SLFila



- Integer **size**; //membro de dados com o tamanho da fila
- Node **first**; // membro com referência ao primeiro elemento da fila
- Node **last**; // membro com referência ao último elemento da fila
- **SLFila_Int()** //método construtor que inicializa uma fila vazia
- void **Enqueue**(Integer) // método Enqueue para acrescentar elemento na fila
- Integer **Dequeue**() // método Dequeue para retirar elemento da fila
- void **Imprime_Fila**() // método para imprimir os dados da fila

Tipo Abstrato de Dados: SLFila

```
package uscs;

public class SLFila_Int {

	public Integer size;
	public Node first;
	public Node last;

	public SLFila_Int(){

		this.first = null;
		this.last = null;
		this.size = 0;
	}
```

Tipo Abstrato de Dados: SLFila

```
public void Enqueue(Integer n) {

        Node novo_node = new Node(n);

        if (this.size == 0) {

                this.first = novo_node;
                this.last = novo_node;
                this.size = this.size + 1;
        }

        else {
                        this.last.next = novo_node;
                        this.last = novo_node;
                        this.size = this.size + 1;
        }
}
```

Tipo Abstrato de Dados: SLFila

```
public Integer  Dequeue() {

            if (this.size == 0)  {

                  System.out.println("Erro no Dequeue... Fila vazia...");
                  return null;
            }
            else {
                        if (this.size == 1) {
                              Integer dado_retornado = this.first.item;
                              this.first = null;
                              this.last = null;
                              this.size = 0;
                              return dado_retornado;

                        }
                        else {
                              Integer dado_retornado = this.first.item;
                              this.first = this.first.next;
                              this.size = this.size - 1;
                              return dado_retornado;
                        }
            }
}
```

Tipo Abstrato de Dados: SLFila

```
public void Imprime_Fila() {

            if (this.size == 0)
                  System.out.println("\nNão há dados para imprimir!  Fila
vazia...");
            else {
                        Node Node_trab = this.first;
                        System.out.println("\nDados na Fila: ");
                        System.out.println("----------------");
                        while (Node_trab != null ) {
                              System.out.print( Node_trab.item + "  ");
                              Node_trab = Node_trab.next;
                        }
                        System.out.println("");
            }
      }
```