Universidade Federal do Pará
Campus Universitário de Castanhal
Faculdade de Computação

Estruturas de Dados
Pilhas

Profª Penha Abi Harb
mpenha@ufpa.br

Pilhas

- As pilhas são estruturas baseadas no princípio LIFO (last in, first out), onde os dados que foram inseridos por último na pilha serão os primeiros a serem removidos.

Listas:Filas,Pilhas

- Lista de compras
  - Arroz
  - Feijão
  - Macarrão
  - Tomate

Muitas operações

- Fila de banco

- Pilhas de Livros

FIFO (first in, first out)

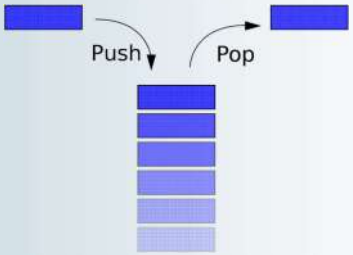
LIFO (last in, first out)

Representação de Pilhas

- Há dois tipos possíveis de representação de Pilhas lineares:
- Por Contiguidade (ESTÁTICA):
  - Garantia da precedência dos elementos pela contiguidade física na memória usando arranjos;
- Por Encadeamento (DINÂMICA)
  - Garantia da precedência dos elementos pelo seu encadeamento (contiguidade lógica) usando-se apontadores.

Pilhas

- Existem duas funções que se aplicam a todas as pilhas:
  - PUSH, que insere um dado no topo da pilha
  - POP, que remove o item no topo da pilha



Implementação de Pilhas: Contiguidade

- Inserção de um elemento no topo da pilha
  PUSH (empilhar)
- Remoção de um elemento no tipo da pilha
  POP (desempilhar)
- Ler o elemento do topo da pilha
- Verificar se a pilha está cheia
  Overflow
- Verificar se a pilha está vazia

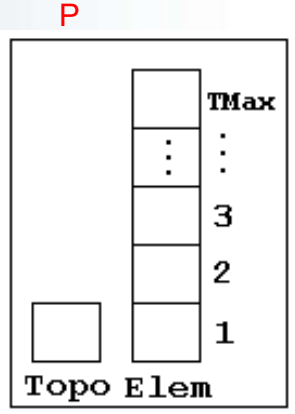
Operações com pilhas

- Inserção de um elemento no topo da pilha
  - PUSH (empilhar)
- Remoção de um elemento no topo da pilha
  - POP (desempilhar)
- Ler o elemento do topo da pilha
- Verificar se a pilha está cheia
  - Overflow
- Verificar se a pilha está vazia

Implementação de Pilhas: contiguidade

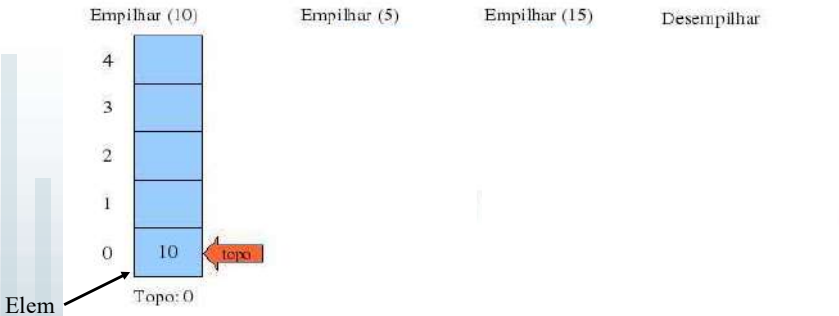
```
classe Pilha
    {
      int Topo;
      int Elem[5];
    };
Pilha P;
```

Topo guarda a ultima posição preenchida da pilha



Representação de Pilhas

Inicializando a Pilha - P.Topo = -1;



Empilhar
P.Topo = 0
P.Elem[0] = 10

Empilhar
P.Topo = 1
P.Elem[1] = 5

Empilhar
P.Topo = 2 ou
P.Topo++
P.Elem[2] = 15

P.Topo = 1 ou
P.Topo--

Implementação de Pilhas: contiguidade

```
classe Pilha
    {
      int Topo;
      int Elem[5];
    };
  Pilha P;
```

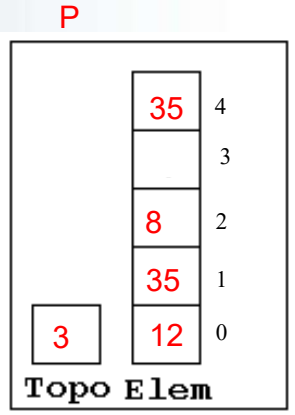
Topo guarda a ultima posição preenchida da pilha

P.Topo = 0
P.Elem[0] = 12

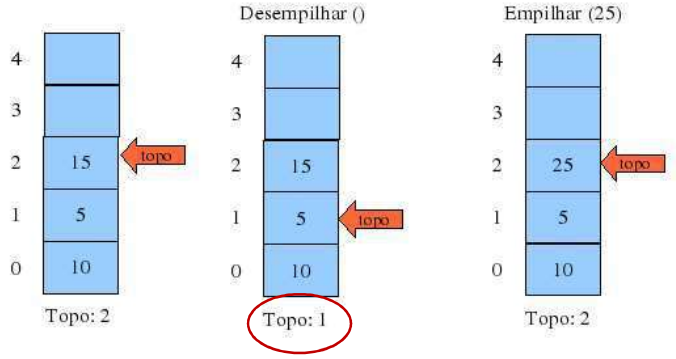
P.Elem[2] = 8
P.Topo = 2

P.Topo = 1
P.Elem[1] = 35

P.Elem[4] = 35
P.Topo = 3



Representação de Pilhas



Na realidade a remoção de um elemento da pilha é realizada apenas alterando-se a informação da posição do topo.

Implementação de Pilhas: contiguidade

```
classe Pilha
    {
      int Topo;
      int Elem[5];
    };
  Pilha P;
```

Topo guarda a ultima posição preenchida da pilha

P.Topo = 0
P.Elem[0] = 12

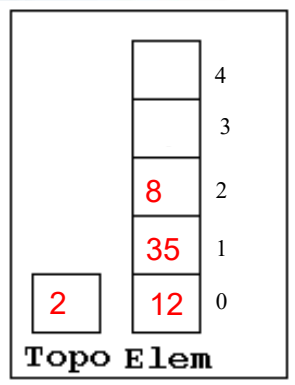
P.Topo = 1
P.Elem[1] = 35

P.Elem[2] = 8
P.Topo = 2

Desempilhar
P.Topo--;

Verificar Elemento

```
System.out.println(
        P.Elem[P.Topo]);
```

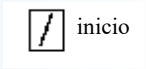


Implementação de Pilhas: Encadeadas

Encadeadas: Operações

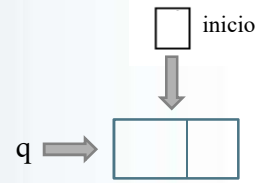
- Inicialização

```
Pilha P1 = new Pilha();
P1.inicio = null;
```



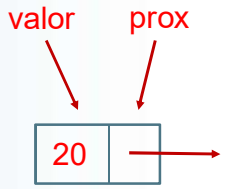
- Empilhamento do valor V

```
void push( int V)
{
      noPilha q = new noPilha;
      q.valor = V
      q.proximo = inicio;
      inicio = q;
}
```



Implementação de Pilhas: Encadeadas

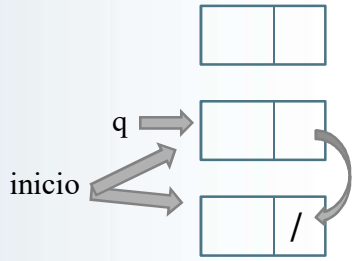
```
public class noPilha {
   public int valor;
   public noPilha proximo;
}
```



Encadeadas: Operações

- Empilhamento do valor V

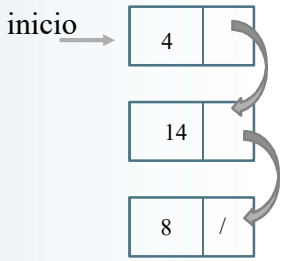
```
void push( int V)
{
      noPilha q = new noPilha;
      q.valor = V
      q.proximo = inicio;
      inicio = q;
}
```



Encadeadas: Operações

- Desempilhando

```
void pop(){
   if (inicio == NULL)
     System.out.println("Pilha vazia");
   else {
     inicio = inicio.prox;
    }
}
```



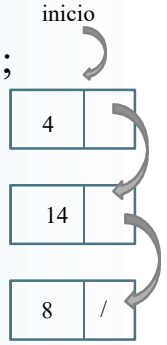
Filas

- As filas são estruturas baseadas no princípio FIFO (first in, first out), em que os elementos que foram inseridos no início são os primeiros a serem removidos.
- Uma fila possui duas funções básicas:
  - ENQUEUE, que adiciona um elemento ao final da fila,
  - DEQUEUE, que remove o elemento no início da fila.

Encadeadas: Operações

- Obtendo o Elemento do Topo

```
void ElementoTopo() {
 if (inicio == null){
   System.out.println("Pilha Vazia!");
}else{
   System.out.println(inicio.valor);
}
```



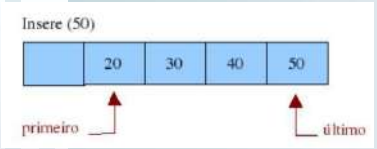
Filas

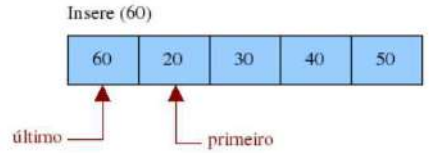
Representação de Filas

- Há dois tipos possíveis de representação de Filas:
- Por Contiguidade (ESTÁTICA):
  - Garantia da precedência dos elementos pela contiguidade física na memória usando arranjos;
- Por Encadeamento (DINÂMICA)
  - Garantia da precedência dos elementos pelo seu encadeamento (contiguidade lógica) usando-se apontadores.

Filas Contiguas

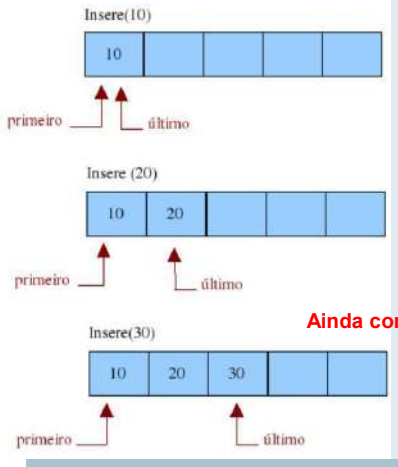
- Para evitar problemas de não ser capaz de inserir mais elementos na fila, mesmo quando ela não está cheia, as referências primeiro e último circundam até o inicio do vetor, resultando numa fila circular.

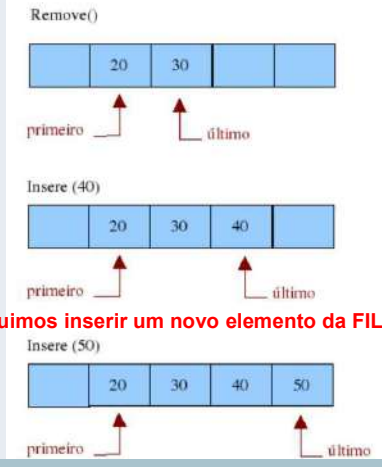




Filas Contiguas

- Supondo uma fila com capacidade para 5 elementos.





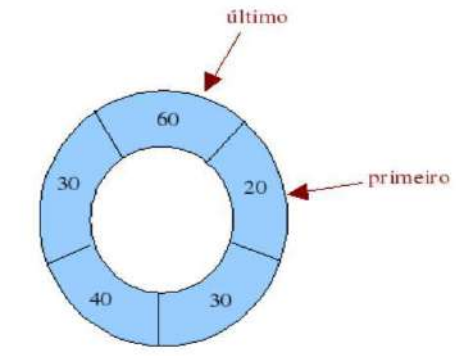
Ainda conseguimos inserir um novo elemento da FILA??

A remoção de um elemento da fila é realizada apenas alterando-se a informação da posição do primeiro.

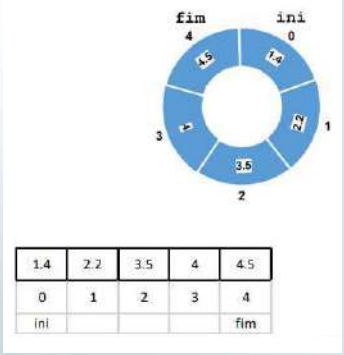
Filas Contiguas

- Desta forma a fila simula uma representação circular:

```
classe FILA
  {
    int primeiro;
    int ultimo;
    int Elem[TMax];
  };
```



Filas Circulares - vetor

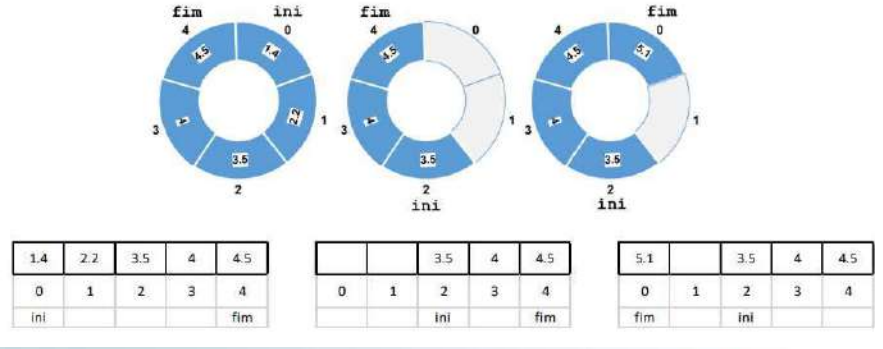


Conseguimos inserir??

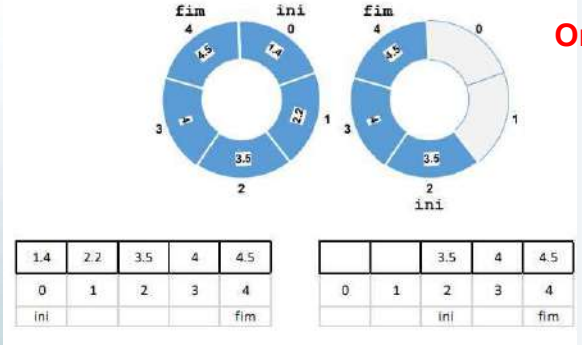
Conseguimos remover??

Qual valor será removido?

Filas Circulares - vetor



Filas Circulares - vetor



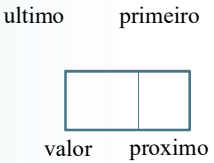
Conseguimos inserir??

Conseguimos remover??

Onde será inserido?

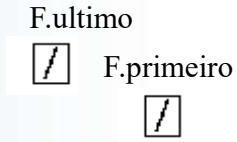
Implementação de Filas: Encadeadas

- **public class noFila {**
- // FILAS ENCADEADAS
- **public int valor;**
- **public noFila proximo;**
- // <u>Ponteiros de controle</u>
- **public noFila primeiro;**
- **public noFila ultimo;**
- }



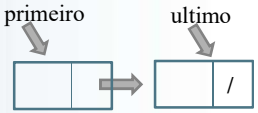
Encadeadas: Operações

```
//inicialização
public class Fila {
        //metodo main
       Fila F = new Fila();
       F.primeiro = null;
       F.ultimo = null;
```



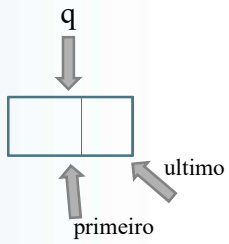
Encadeadas: Operações

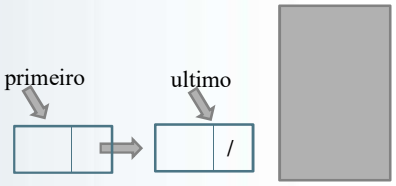
```
//remover elementos na FILA
public void remover( ){

   if (primeiro == null){ // fila vazia
       System.out.println("fila vazia");
   }else{
        primeiro = primeiro.proximo;
       if (primeiro == null) //um elemento apenas
           ultimo = null;
   }
}
```



Encadeadas: Operações

```
// inserir elementos na FILA
public void inserir(int V){
    noFila q = new noFila();
    q.valor = V;
    q.proximo = null;
    if (primeiro == null){ // fila vazia
        primeiro = q;
        ultimo = q;
    }else{
        ultimo.proximo = q;
        ultimo = q;
    }
}
```





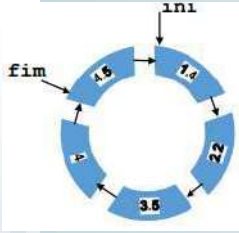
Encadeadas: Operações

- Consulta

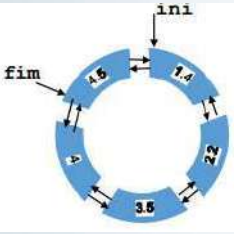
```
//Ler o elemento da Fila
public void elemento (){
// verificar se está vazia antes
     System.out.println(primeiro.valor);
}
```

Podemos também imprimir toda a fila

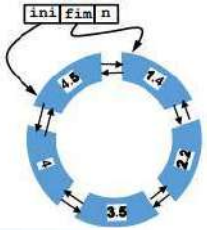
Filas Circulares - encadeadas



Simplesmente encadeada



Duplamente encadeada



Duplamente encadeada com tamanho n