CMP1054 – ED I

Java

Filas usando estruturas auto-referenciadas

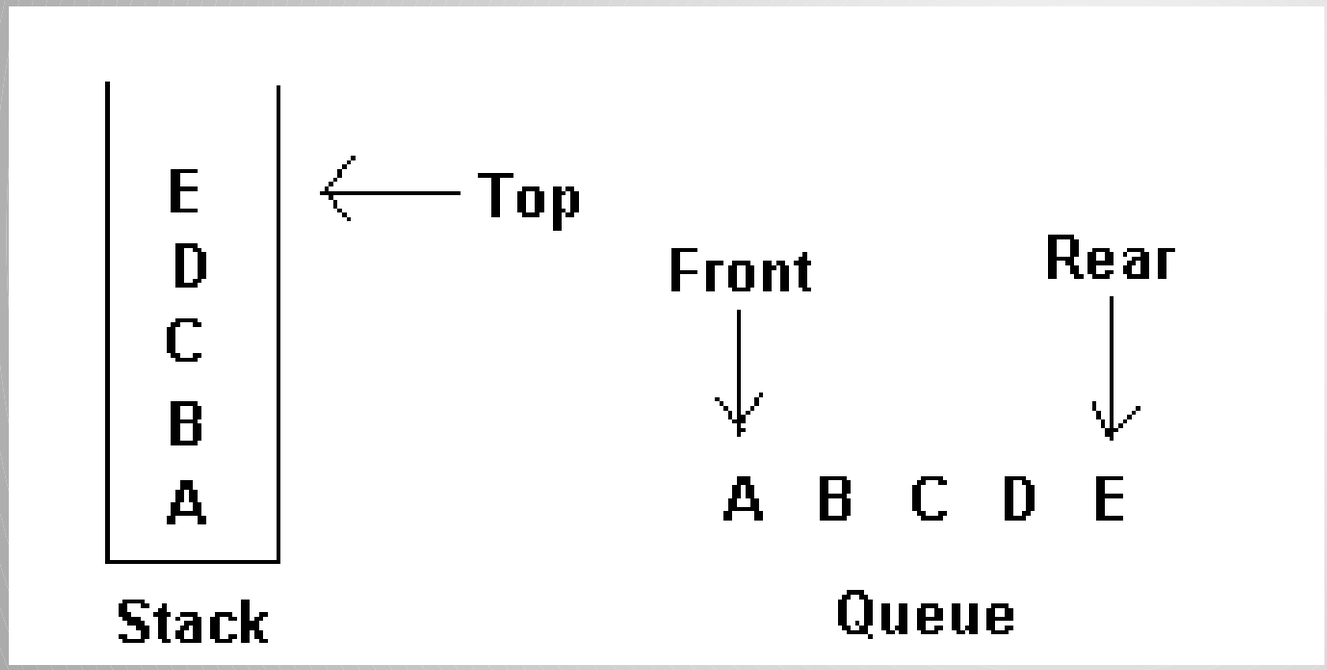
Prof. Dr. José Olimpio Ferreira

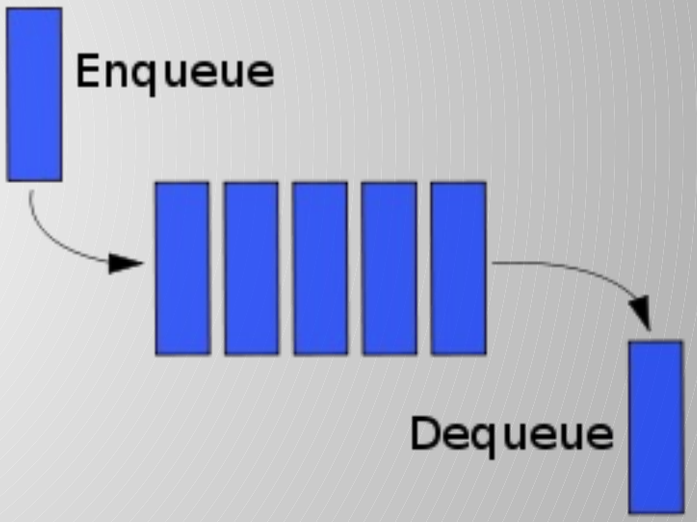
Fila

- É uma lista linear em que todas as inserções são realizadas em um extremo da lista, e todas as retiradas e, geralmente, os acessos são realizados no outro extremo da lista.
- O modelo intuitivo de uma fila é o de uma fila de espera em que as pessoas no início da fila são servidas primeiro e as pessoas que chegam entram no fim da fila.
- São chamadas listas fifo (“first-in”, “first-


designed by freepik


ZZZZ


E
D
C
B
A
Top
Stack
Front
Rear
A B C D E
Queue


Enqueue
Dequeue

Fila

- Existe uma ordem linear para filas que é a “ordem de chegada”.
- São utilizadas quando desejamos processar itens de acordo com a ordem “primeiro-que-chega, primeiro-a-ser-atendido”.
- Sistemas operacionais utilizam filas para regular a ordem na qual tarefas devem receber processamento e recursos devem ser alocados a processos.

TAD Filas

- Criar uma fila vazia.
- Enfileirar o item X no final da fila.
- Desenfileirar. Essa função retorna o item X no início da fila e o retira da fila.
- Verificar se a fila está vazia. Essa função retorna true se a fila está vazia; do contrário, retorna false.

Estrutura e operações sobre Filas Usando Estruturas Auto-referenciadas

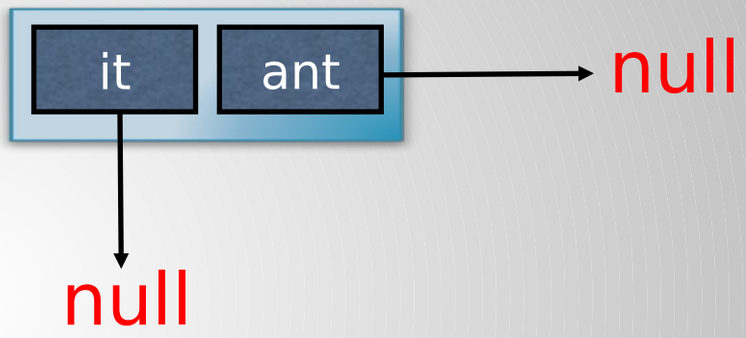
- A classe Item é responsável por manter as informações a serem enfileiradas
- A fila é implementada por meio de nós ou células.
  - classe No
  - Cada No contém um Item da fila e uma referência para outro No.
- A classe Fila
  - Contém uma referência para a frente (inicio) da fila e uma referência para a parte de trás (fim) da fila.
  - Os métodos para manipular a fila.

Objetos do tipo Item

- Atributos:
  - string nome
  - int numero
- Métodos:
  - construtor
  - getNome, getNumero
  - setNome setNumero
  - getItem

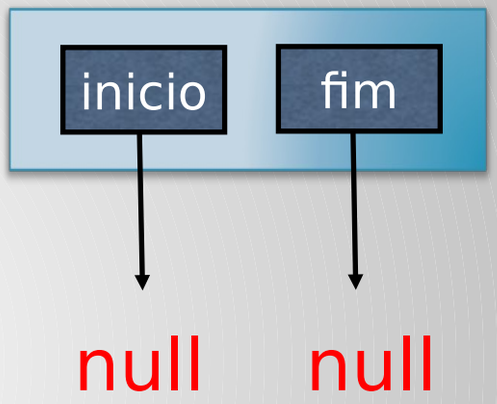
Objetos do tipo No

- Atributos:
  - Item it
  - No ant
- Métodos
  - construtor
  - setItem e setant
  - getItem e getant

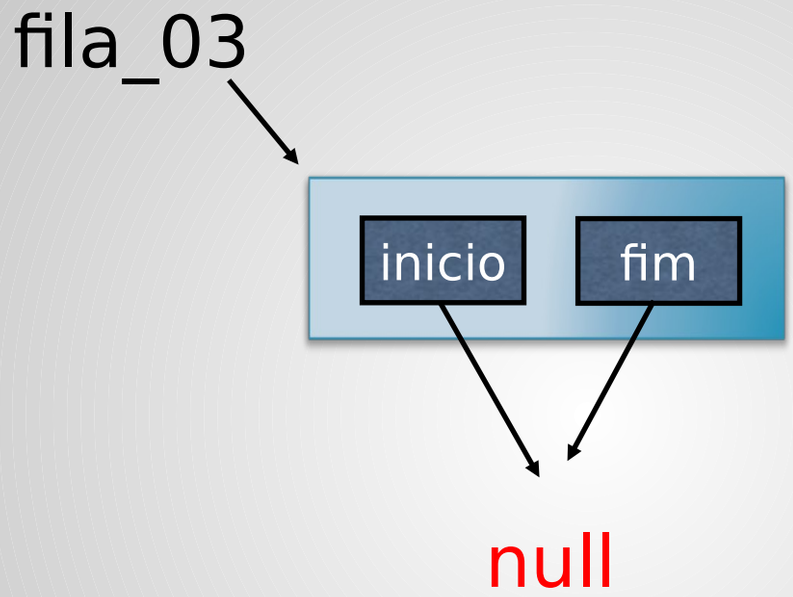


Objetos do tipo Fila

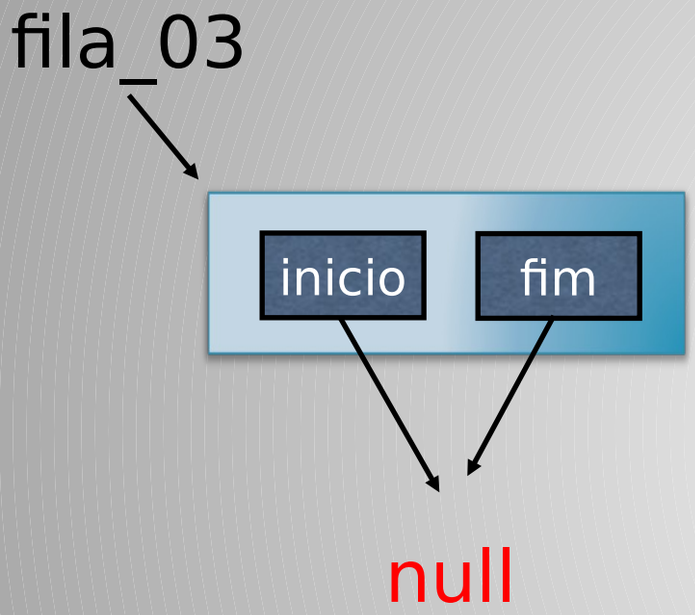
- Atributos:
  - No inicio, fim;
- Métodos:
  - construtor
  - enfileirar
  - desenfileirar
  - filaVazia

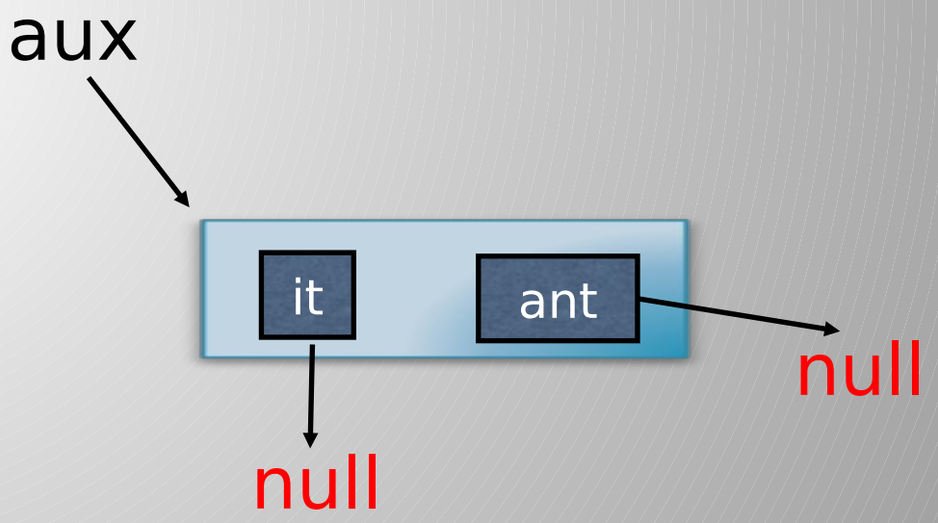


Fila(): Constrói fila vazia

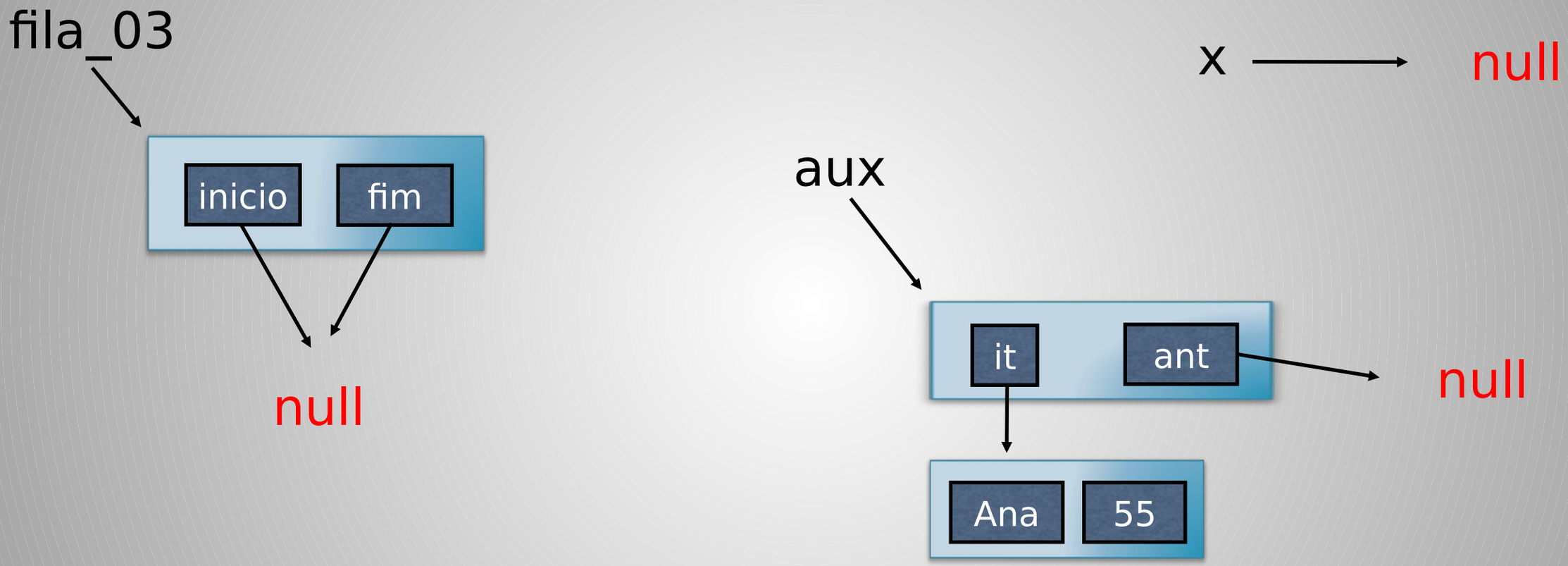


enfileirar(Item x)

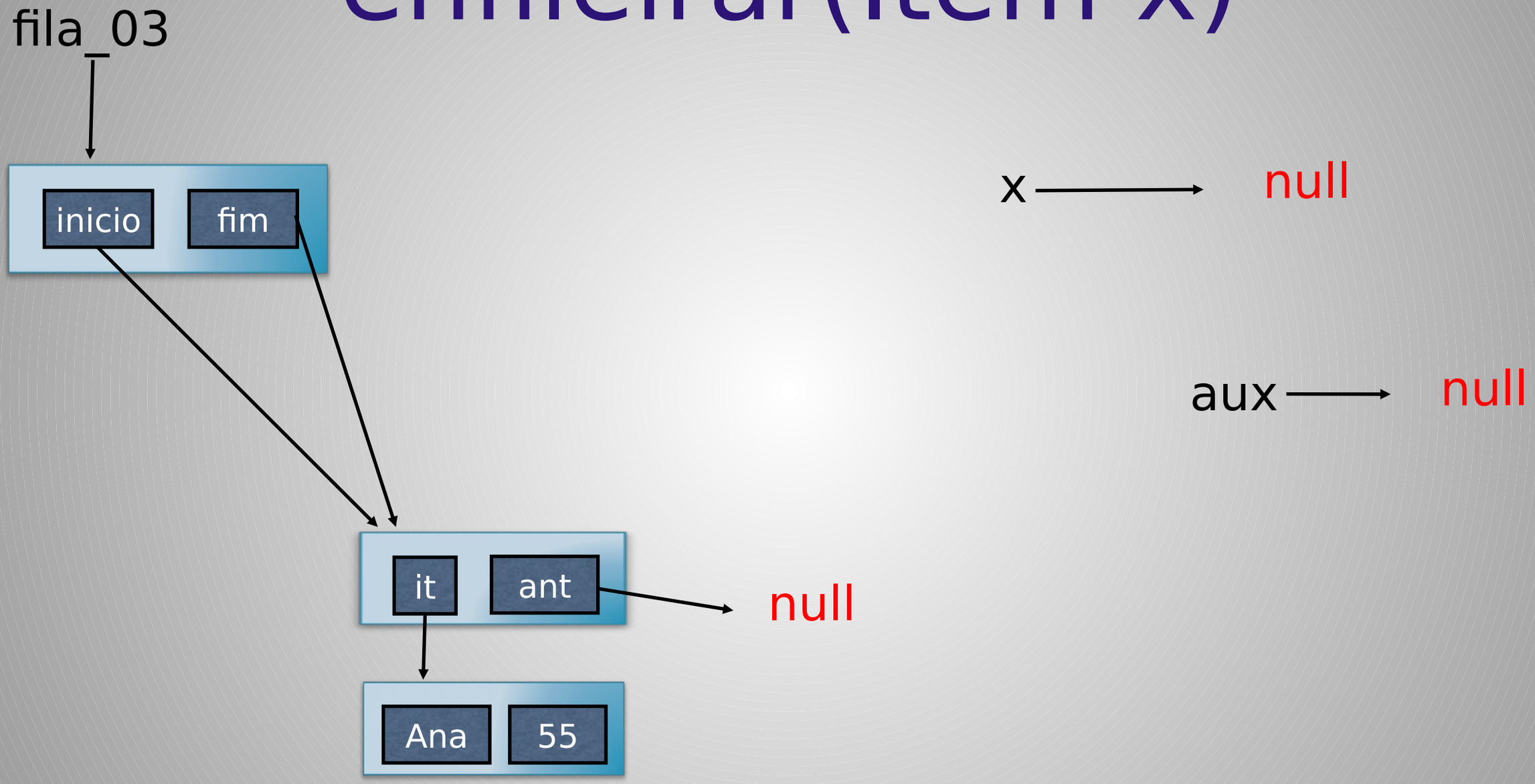




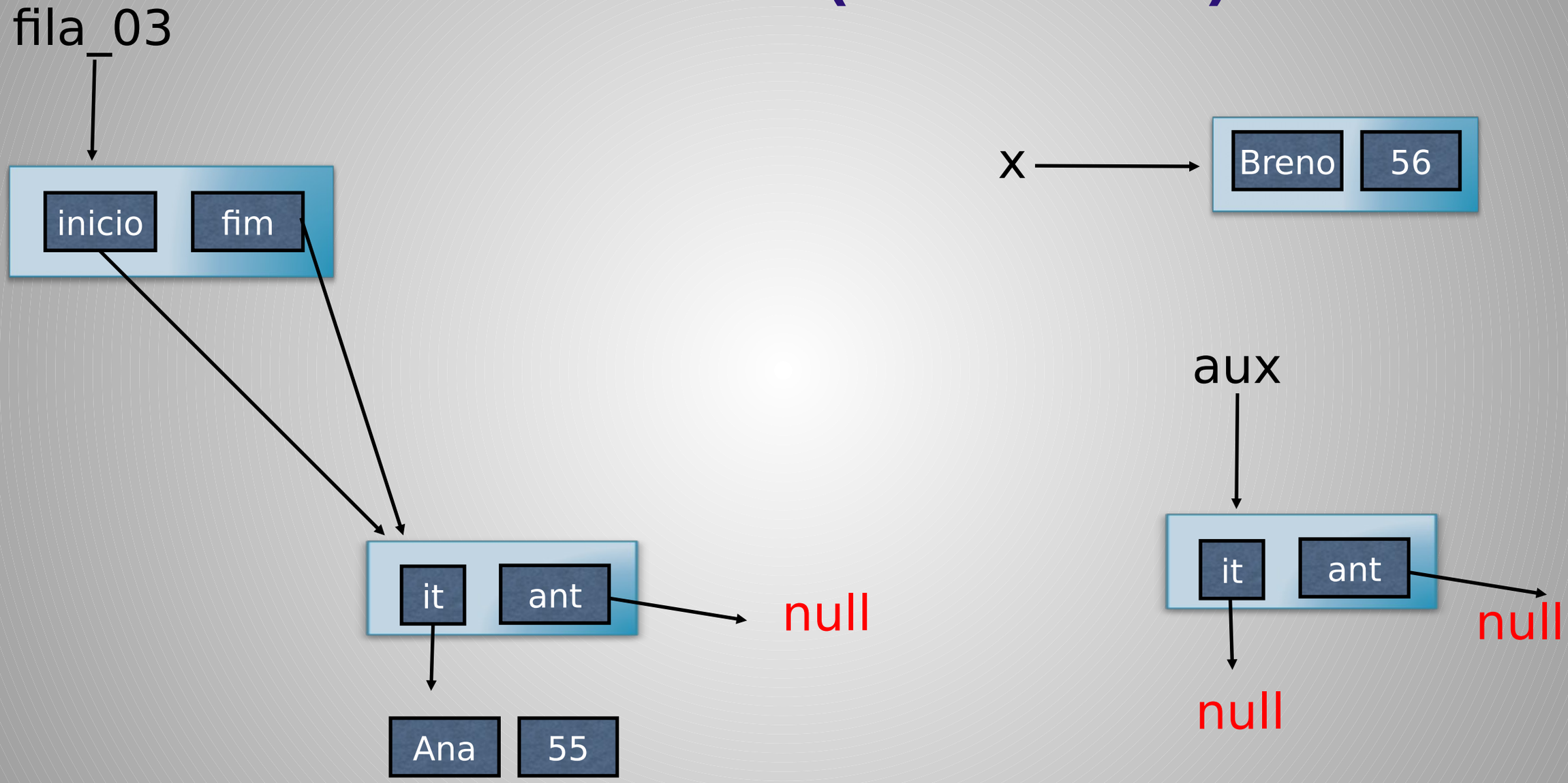
enfileirar(Item x)



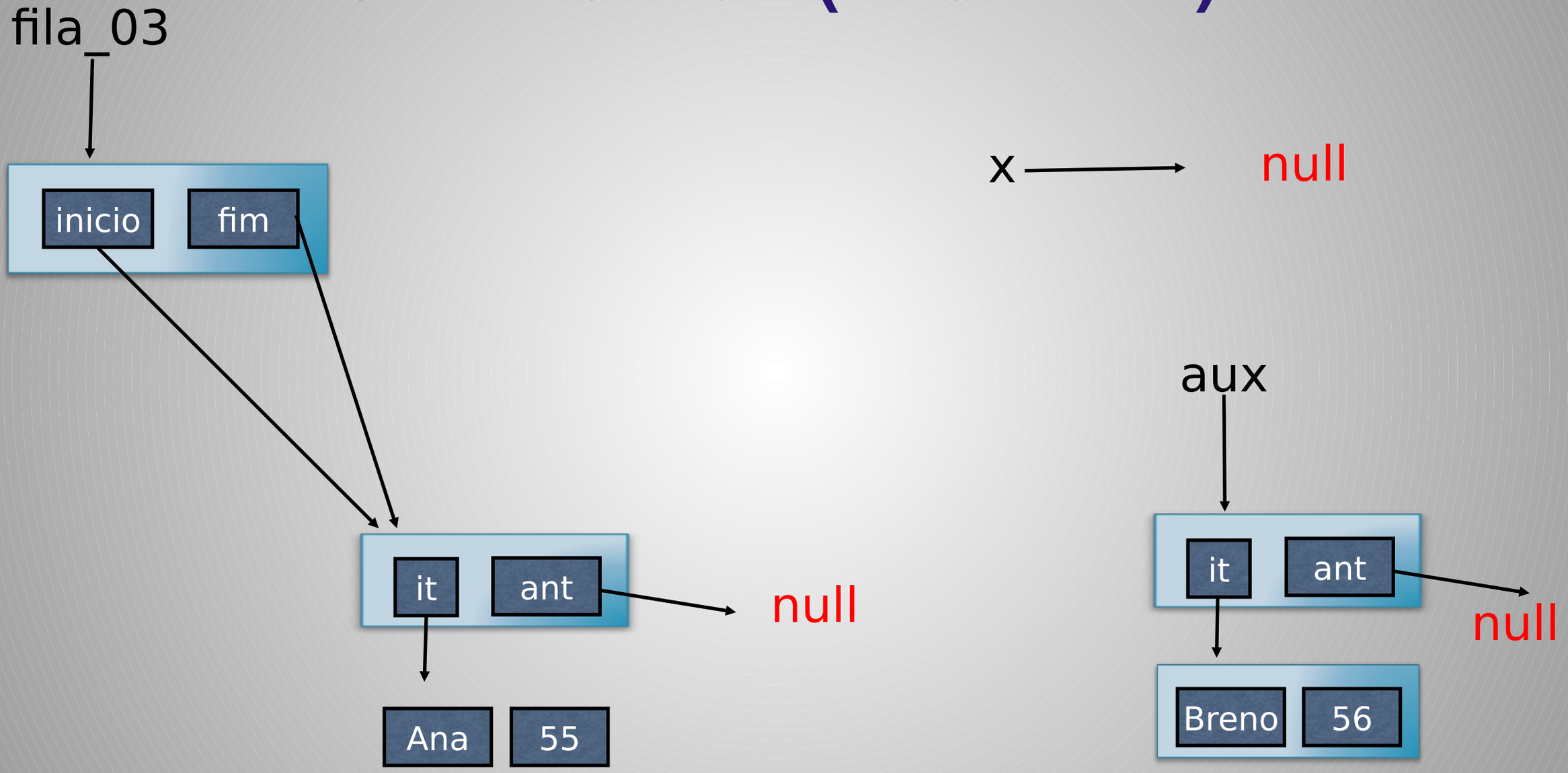
enfileirar(Item x)



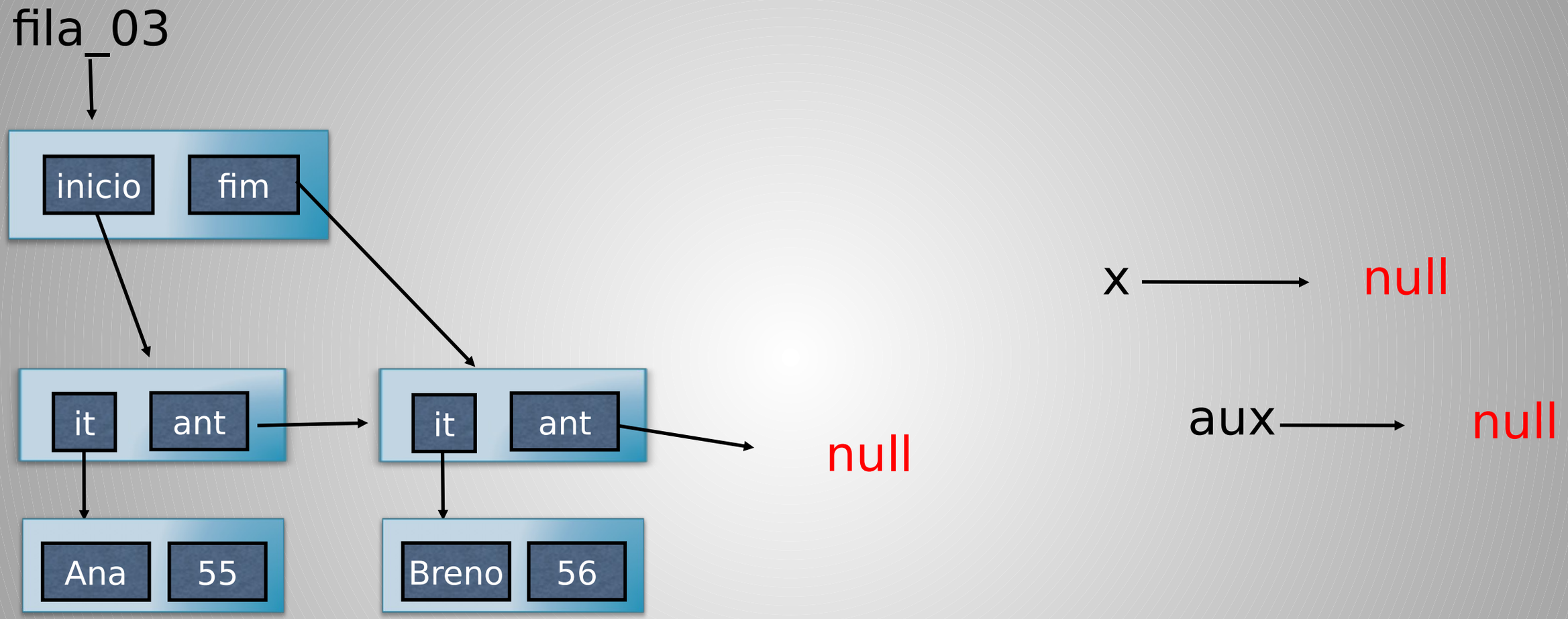
enfileirar(Item x)



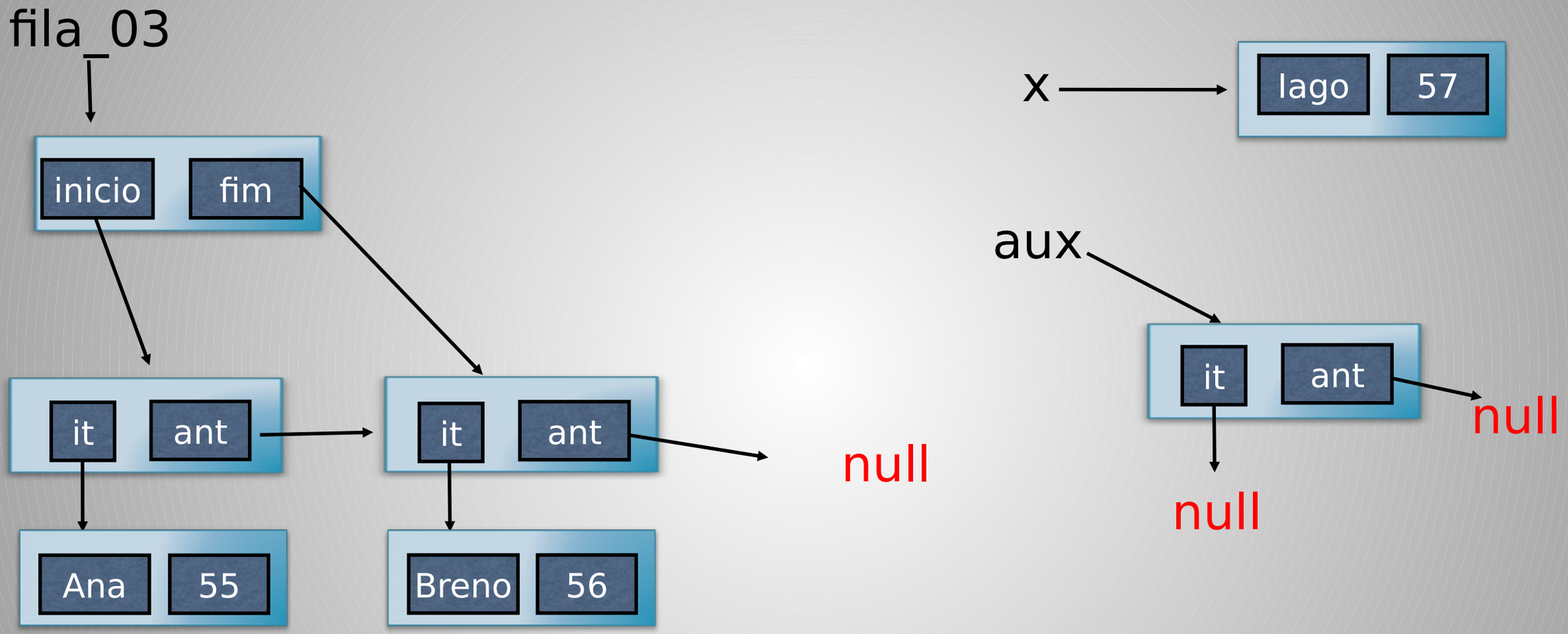
enfileirar(Item x)



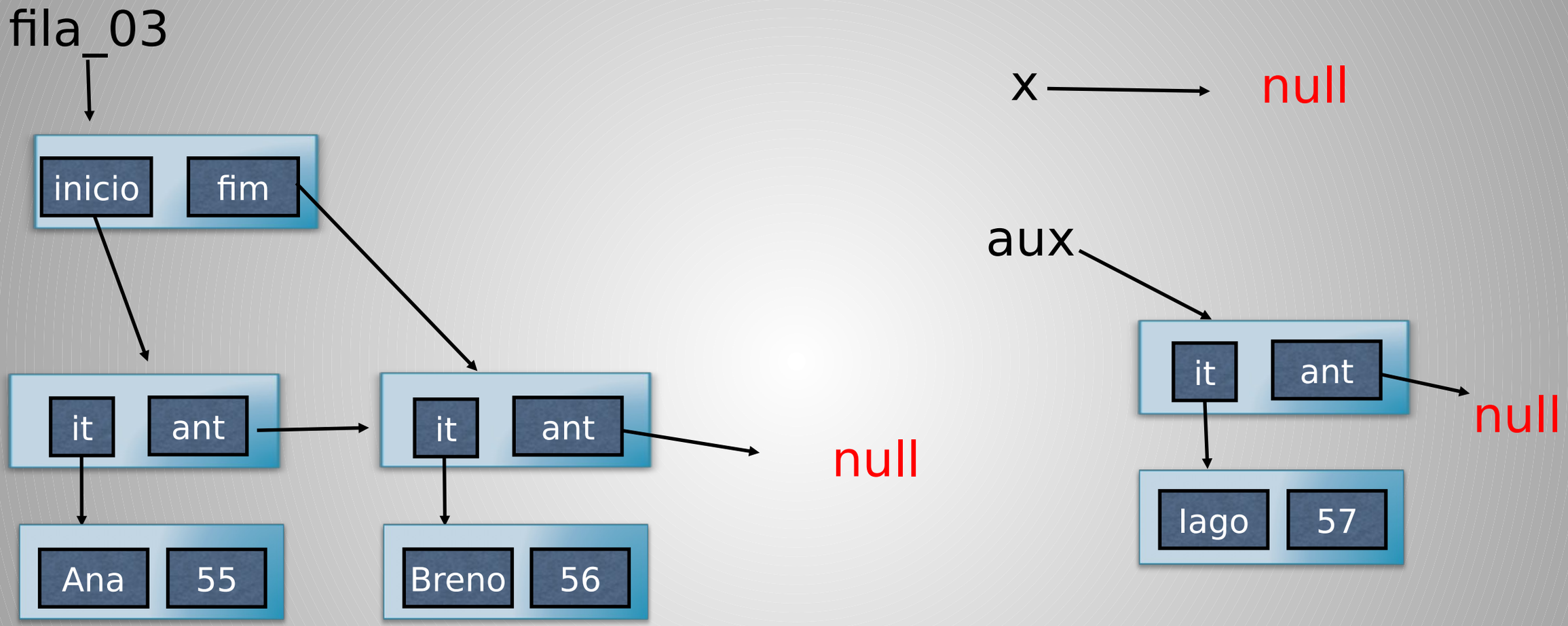
enfileirar(Item x)



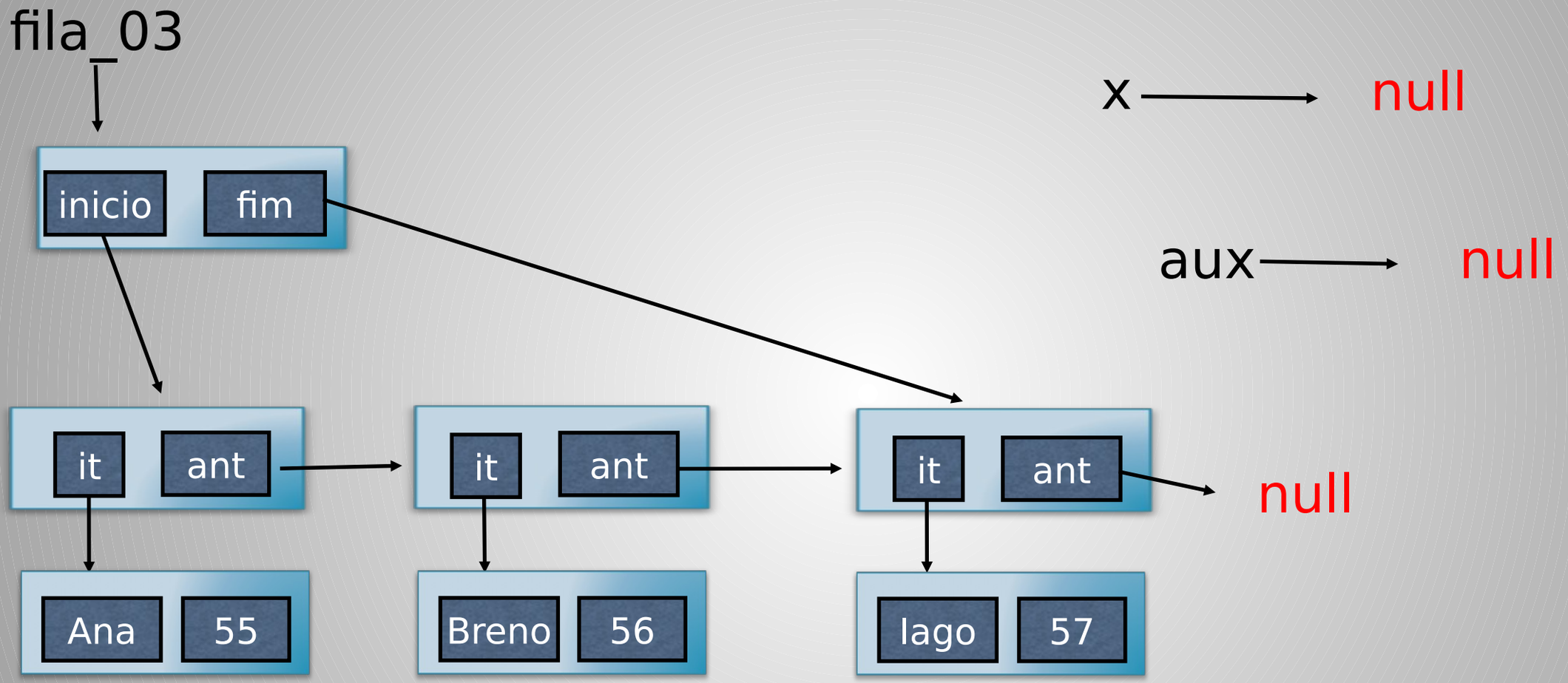
enfileirar(Item x)



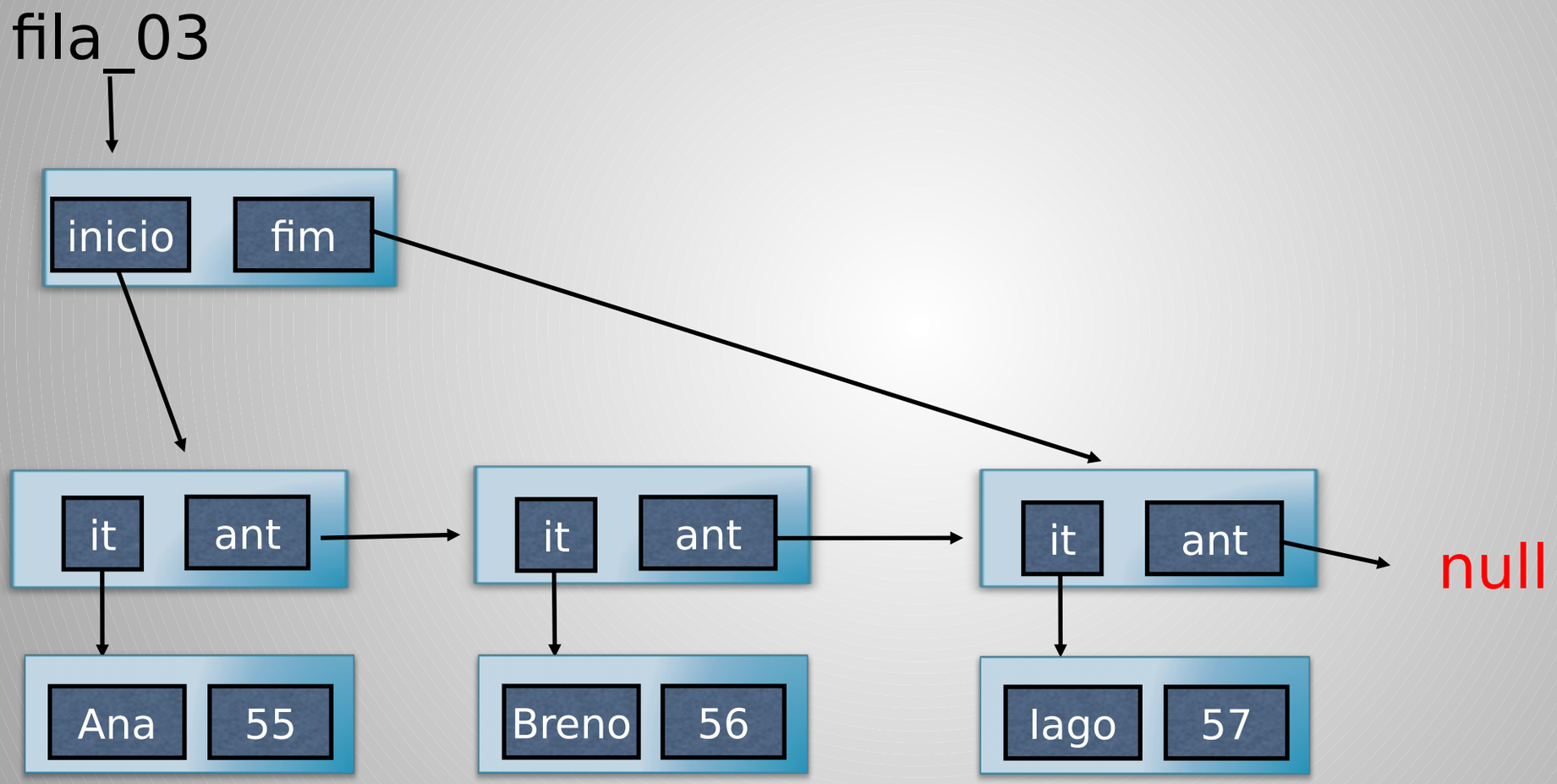
enfileirar(Item x)



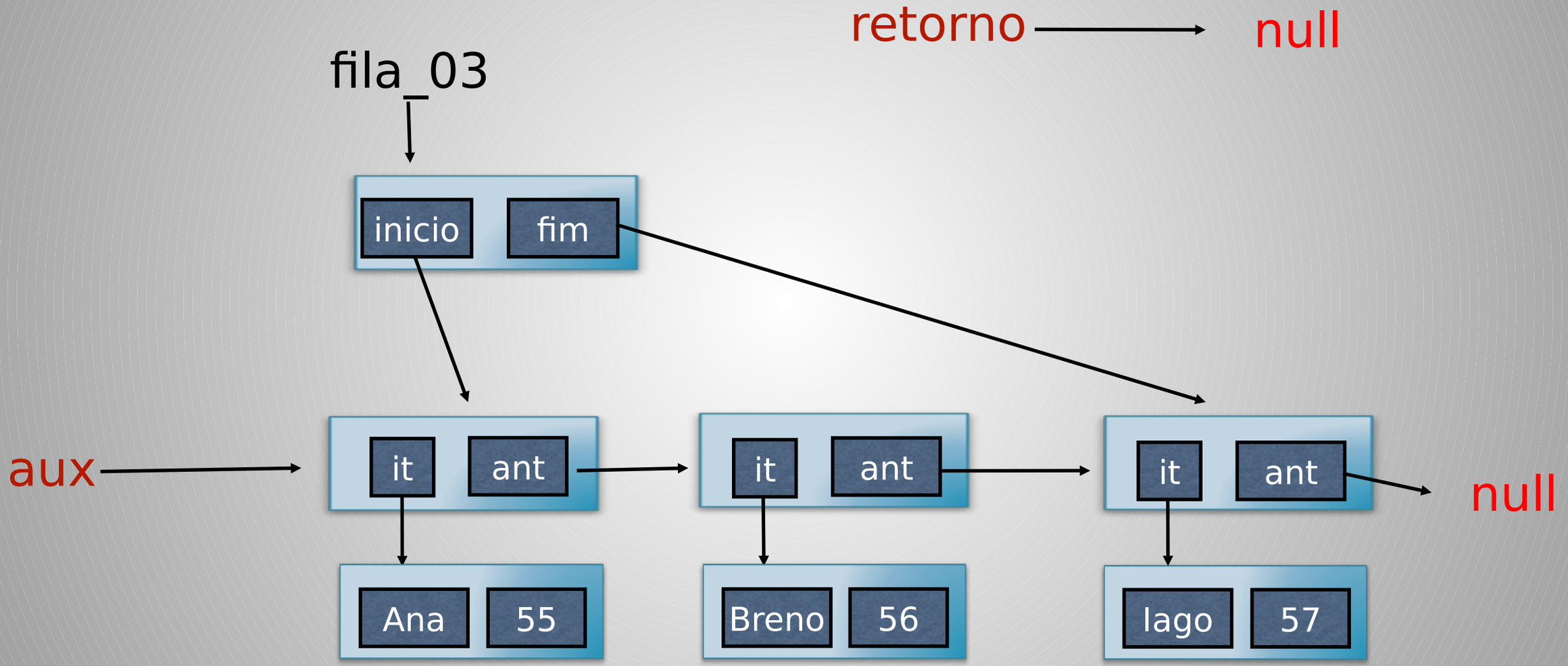
enfileirar(Item x)



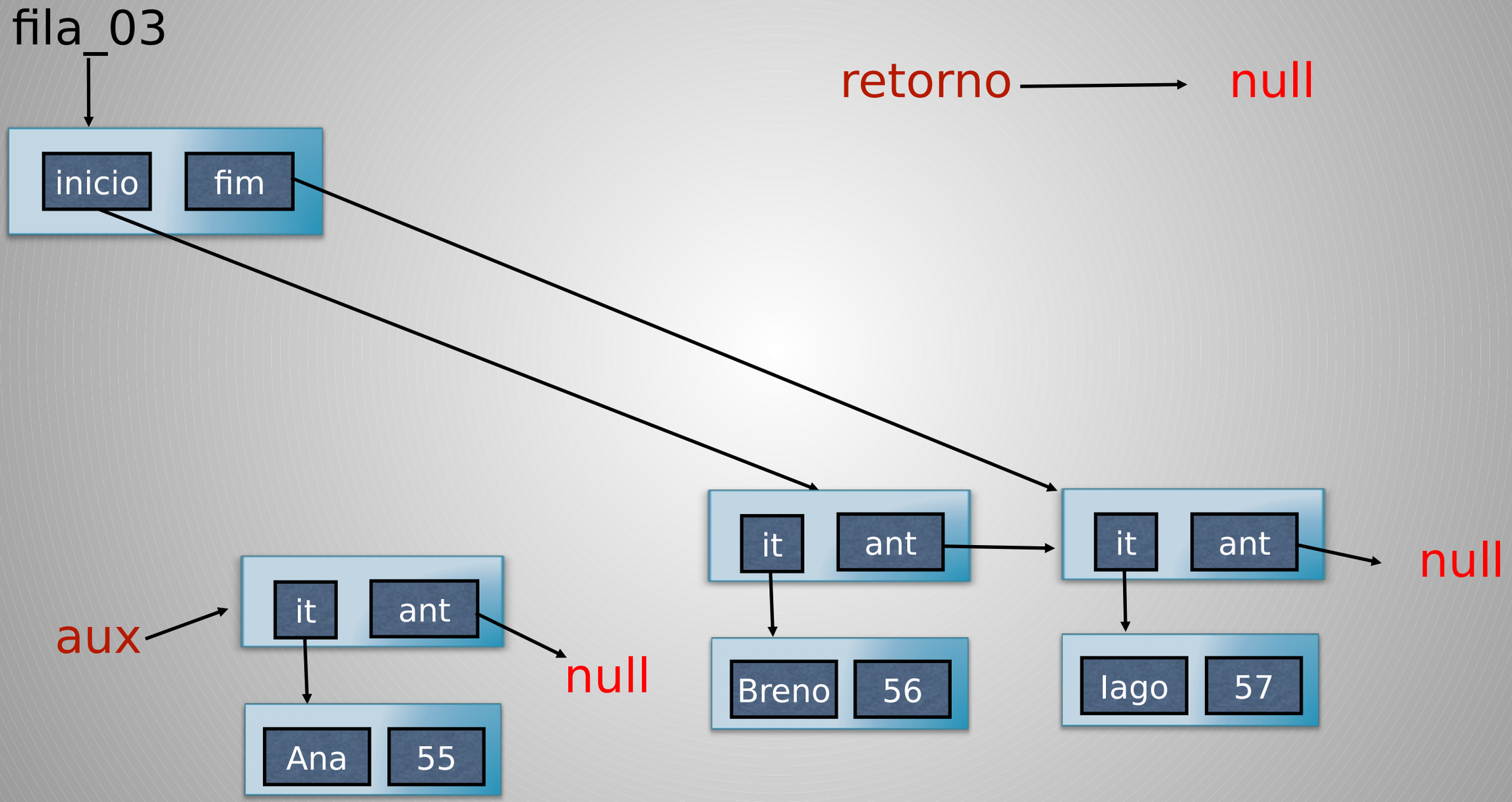
Fila após 3 chegadas



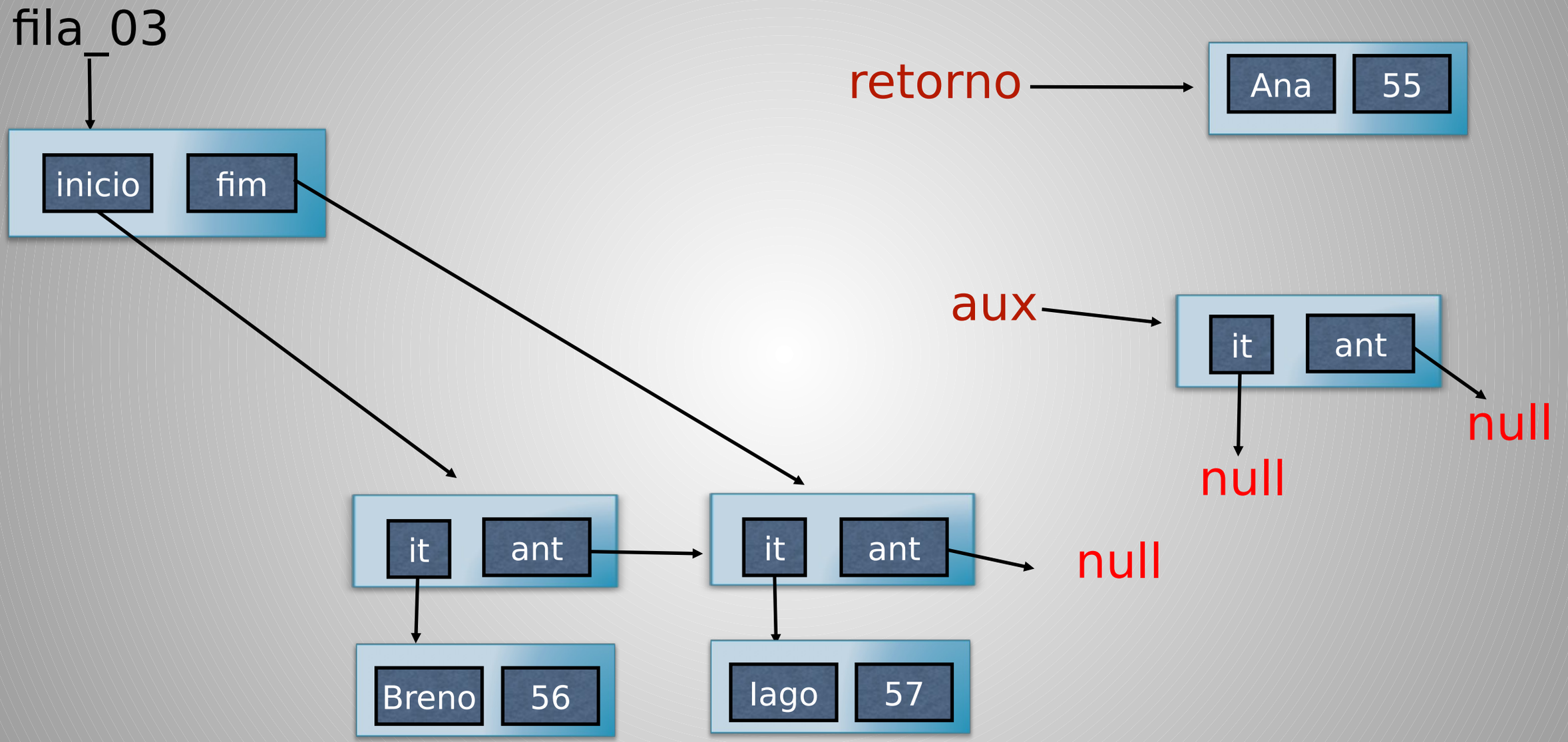
Item desenfileirar()



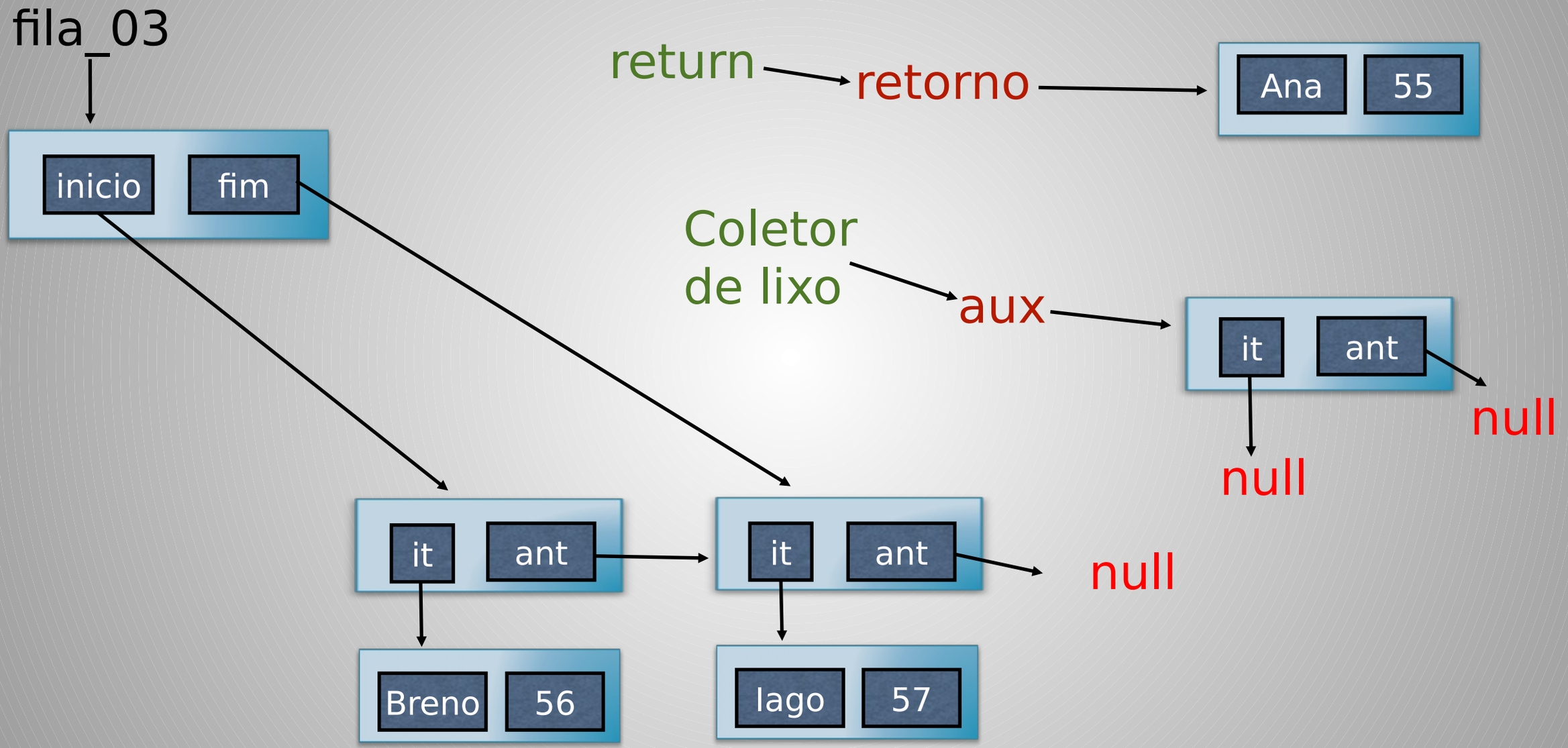
Item desenfileirar()



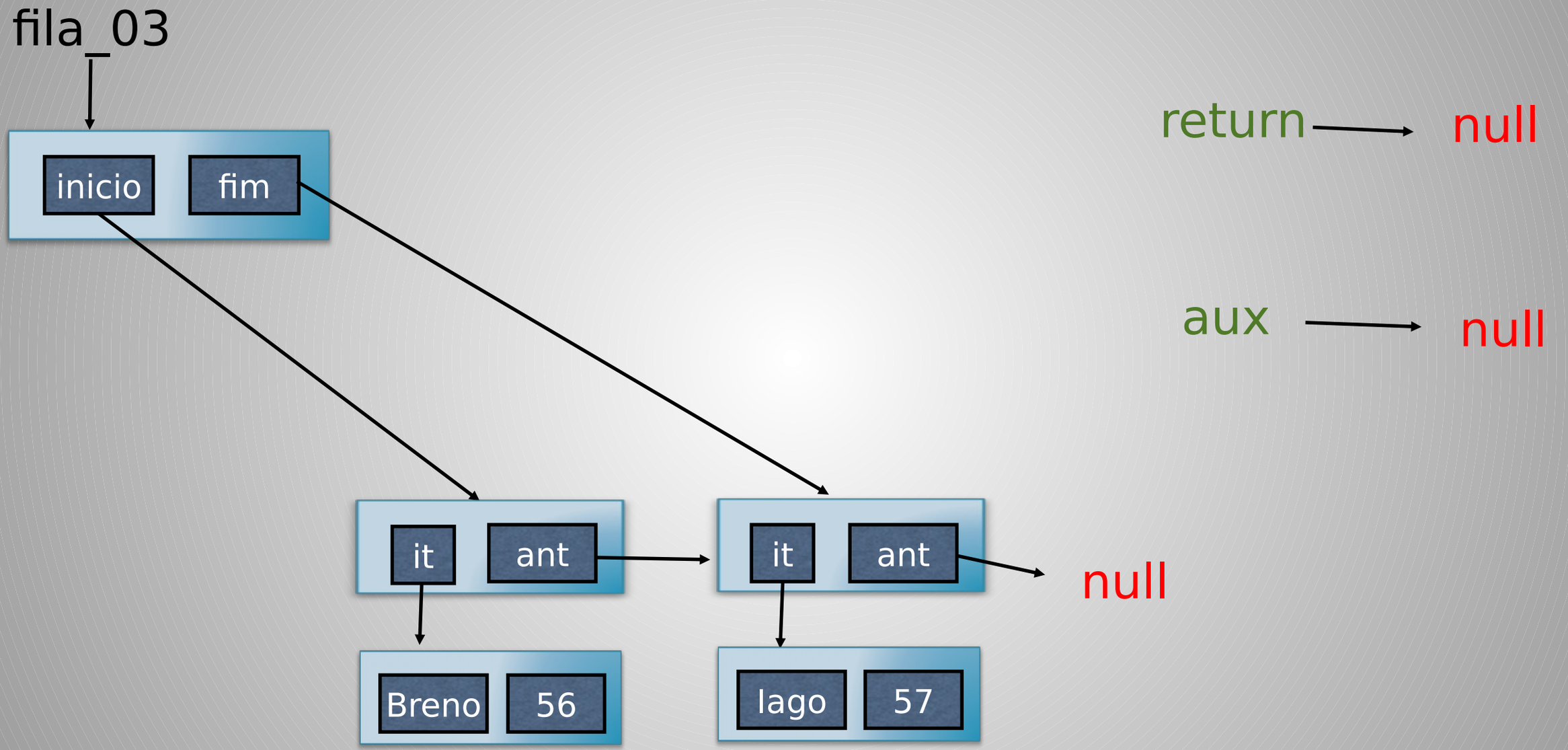
Item desenfileirar()



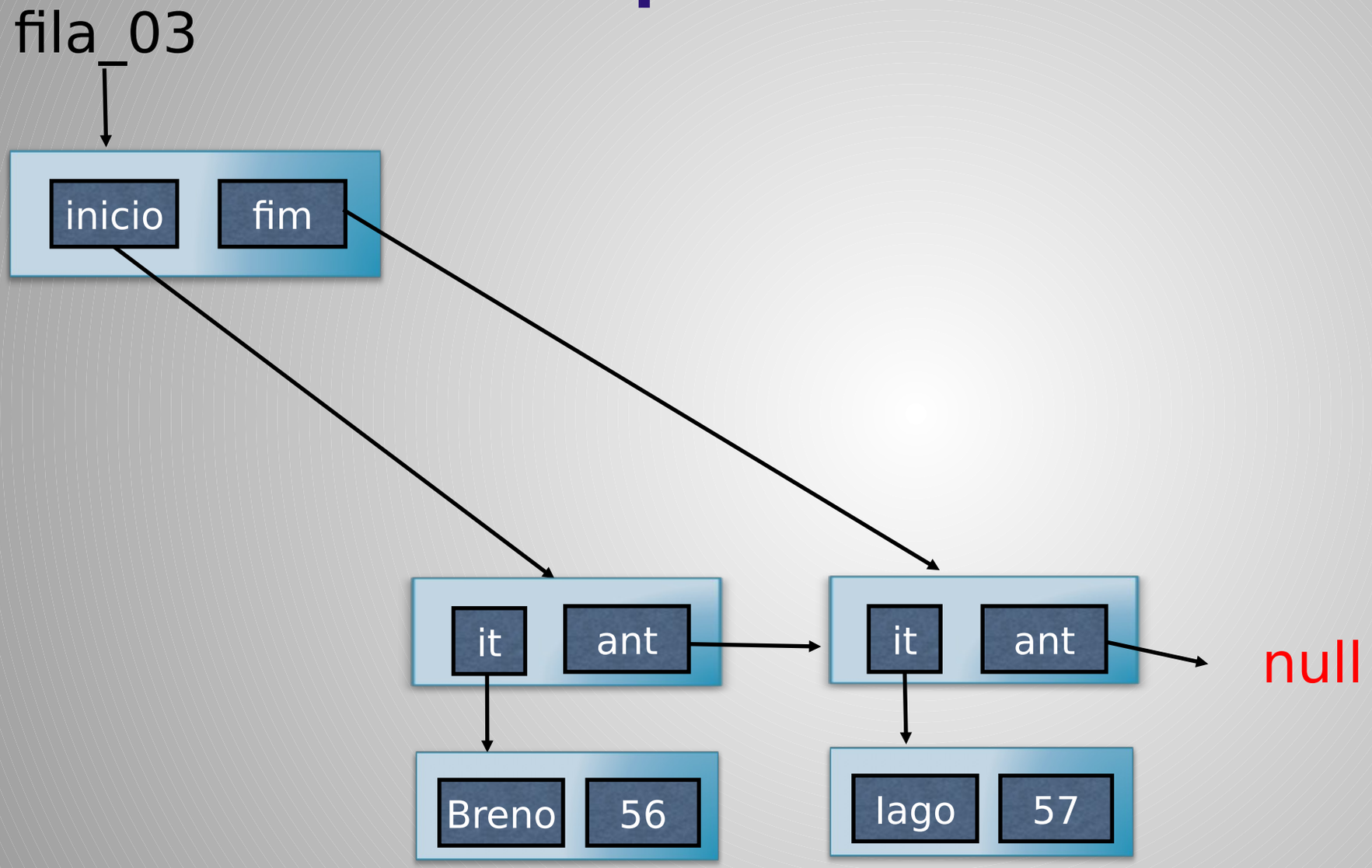
Item desenfileirar()



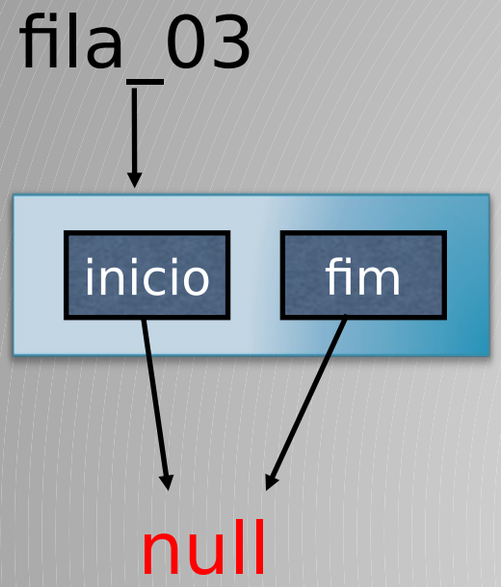
Item desenfileirar()



Fila após uma saída



Fila após 3 saídas



- A fila voutou a ficar vazia.

Classe Cliente

- nome: String
- cpf: String
- fone: String
+ getCliente(): String
+ Cliente(String, String, String)

Classe Base

- leia: Scanner, static
+ main(): void, static
+ obtemLCliente(): Cliente, static

Classe Fila

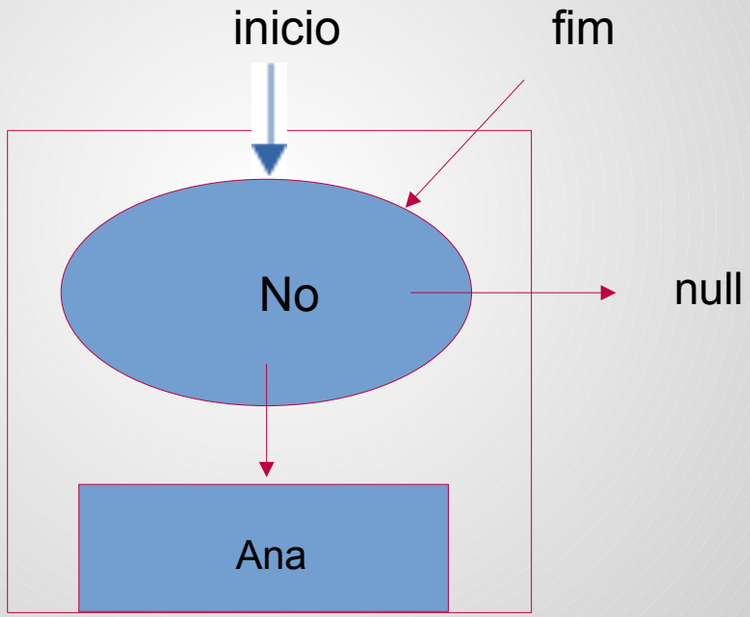
Classe No

- dados: Cliente
- ant: No
+ No(Cliente)

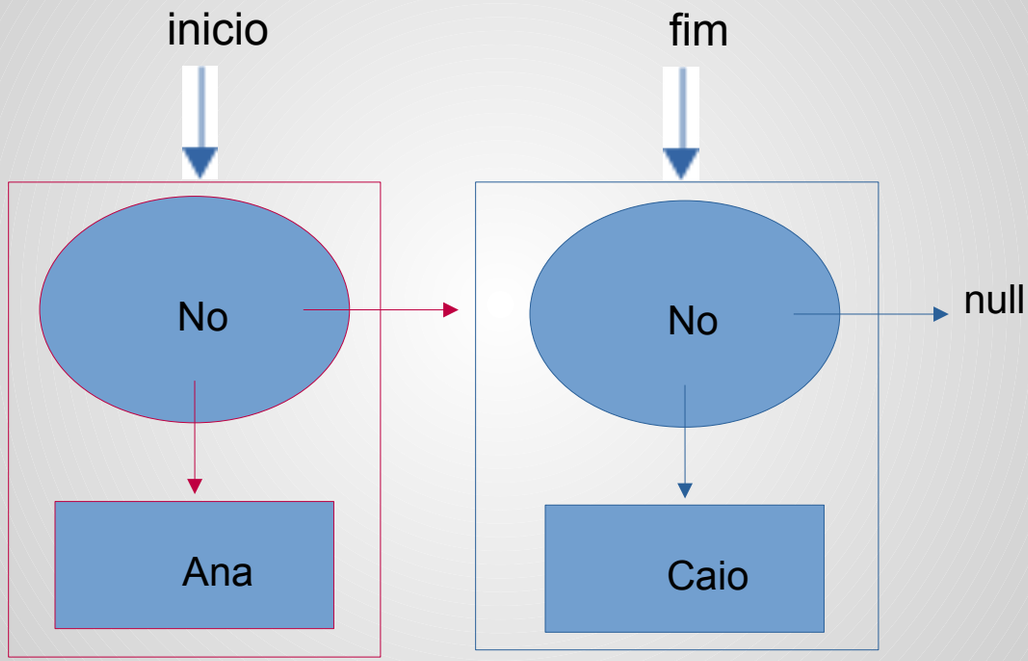
- inicio: No
- fim: No
- qde: int
+ enfileirar(Cliente): void
+ desenfileirar(): Cliente
+ vazia(): lógico
+ getQde(): int

Fila vazia

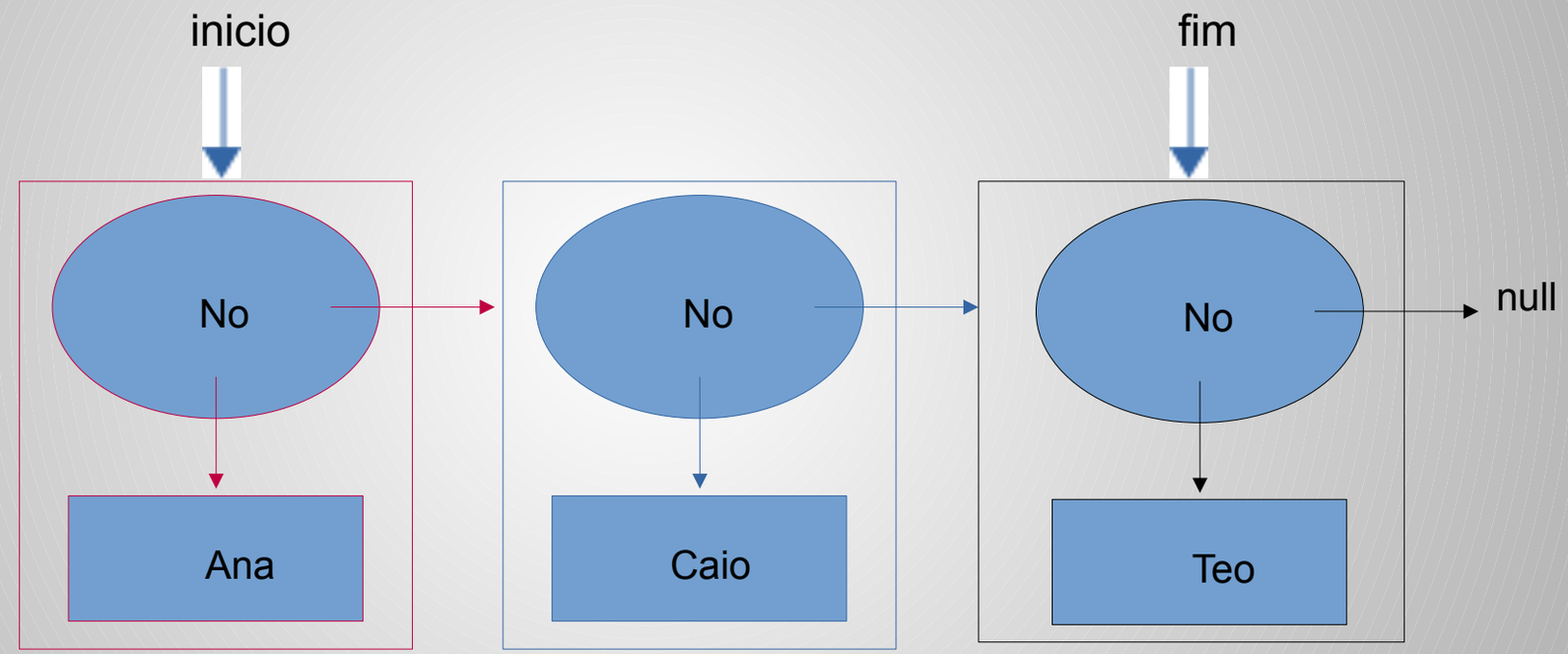
Fila com 1 Cliente



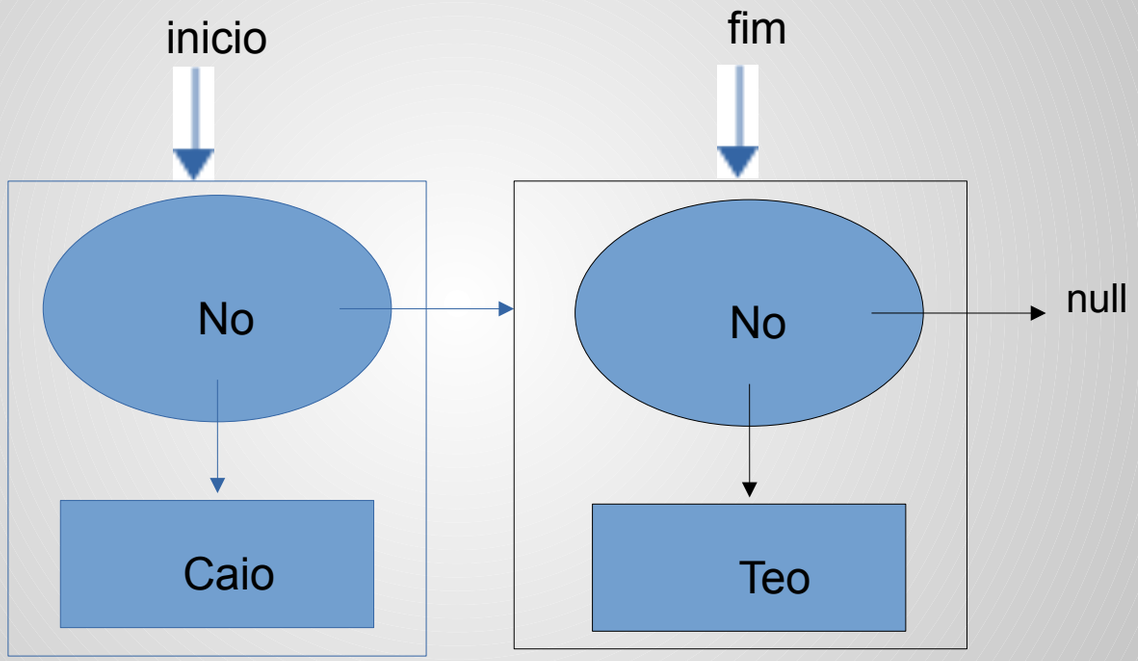
Fila com 2 Clientes



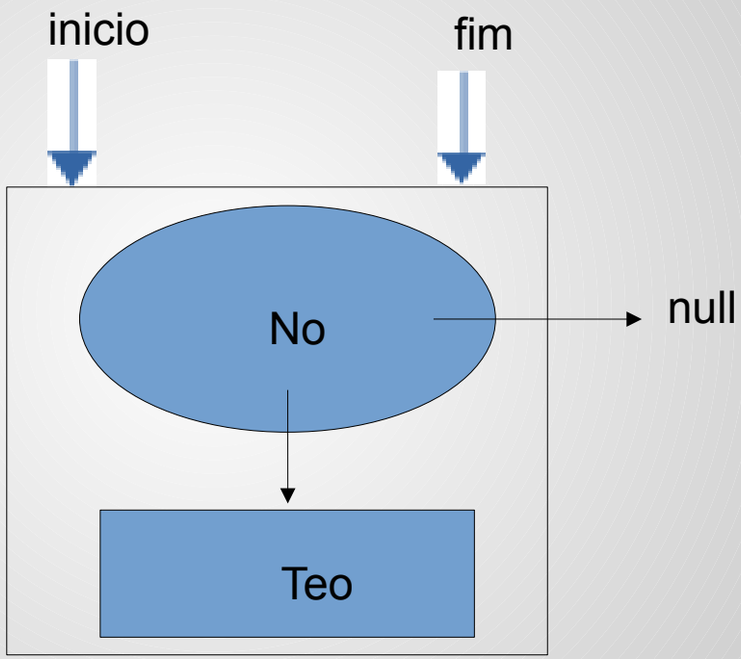
Fila com 3 Clientes



Fila com 2 Clientes



Fila com 1 Cliente



Fila vazia

```
public class Cliente {
  private String nome, fone, cpf;
  public Cliente(String n, String f, String c) {
    nome = n;
    fone = f;
    cpf = c;
  }
  public String getCliente() {
    String aux = nome + "\n" + fone + "\n" + cpf +
"\n";
    return aux;
  }
  }
```

```
public class Fila {
   private class No{
      private Cliente dados;
      private No ant;
      No(Cliente aux) {
         dados = aux;
         ant = null;
      }
   }
   private No inicio, fim;
   private int qde;
   Fila(){
      inicio = fim = null;
      qde = 0;
   }
   public int getQde() {
              return qde;
   }
   public boolean vazia() {
              return inicio == null;
       }
```

```
public void enfileirar(Cliente aux) {
   No novo = new No(aux);
   if(vazia()) {
     inicio = fim = novo;
   }
   else {
      fim.ant = novo;
      fim = novo;
   }
   qde++;
}
public Cliente desenfileirar() {
   if(vazia()) return null;
   No obj = inicio;
   inicio = obj.ant;
   Cliente aux = obj.dados;
   qde--;
   return aux;
}
}
```

```
import java.util.Scanner;
public class Main {
static private Scanner leia = new Scanner(System.in);
    static private Cliente obtemCliente(){
        String n, f, c;
        leia.skip("\\R");
        System.out.println("Nome: ");
        n = leia.nextLine();
        System.out.println("Telefone: ");
        f = leia.next();
        System.out.println("CPF: ");
        c = leia.next();
        Cliente novo = new Cliente(n, f, c);
        return novo;
    }
    public static void main(String[] args) {
        Fila obj = new Fila();
        int op = 0;
        Cliente aux = null;
```

```
do{
            System.out.println("Digite: ");
            System.out.println("1 - para enfileirar.");
            System.out.println("2 - para desenfileirar.");
            System.out.println("3 - para encerrar.");
            op = leia.nextInt();
            switch(op){
                case 1: // enfileirar
                    aux = obtemCliente();
                    obj.enfileirar(aux);
                    System.out.println("Pilha com " + obj.getQde() + "
clientes.");
                    aux = null;
                    break;
                case 2: // desenfileirar
                    aux = obj.desenfileirar();
                    if(aux == null) System.out.println("Fila vazia.");
                    else {
                        System.out.println(aux.getCliente());
                        System.out.println("Pilha com " + obj.getQde() + "
clientes.");
                        aux = null;
                    }
```

```
                    case 3: // encerrar
                        System.out.println("Programa
encerrando.");
                        break;
                    default:
                        System.out.println("Opção inválida.");
                }
            }while(op!=3);
        }
}
```