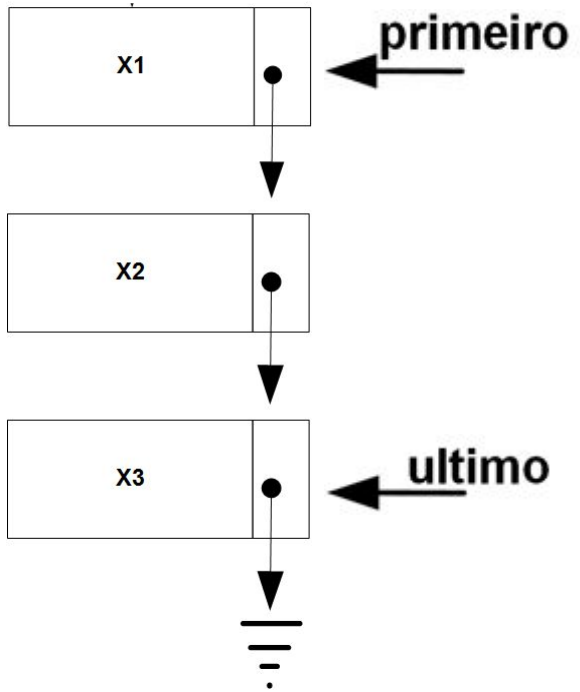
Unidade VI:
Tipos Abstratos de Dados Flexíveis - Fila

Instituto de Ciências Exatas e Informática
Departamento de Ciência da Computação

Código Fonte

- PrincipalFila.java, igual ao da estrutura sequencial

- Fila.java, criará instâncias como:

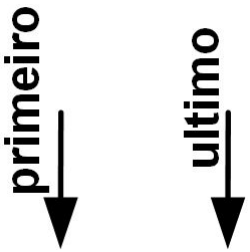


Classe Fila

```
class Fila {
    private Celula primeiro, ultimo;
    public Fila () {
        primeiro = new Celula();
        ultimo = primeiro;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```

Classe Fila

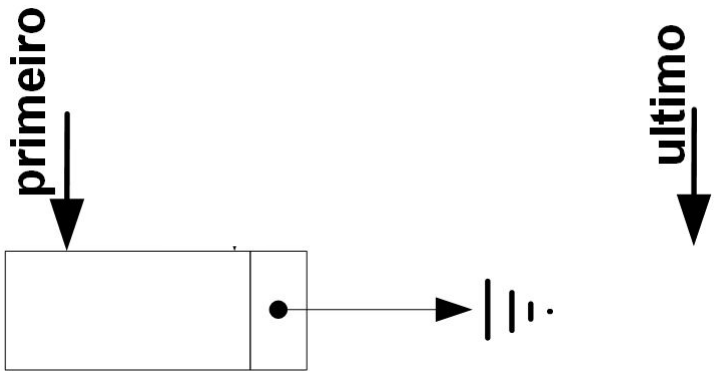
```
class Fila {
    private Celula primeiro, ultimo;
    public Fila () {
        primeiro = new Celula();
        ultimo = primeiro;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```



Classe Fila

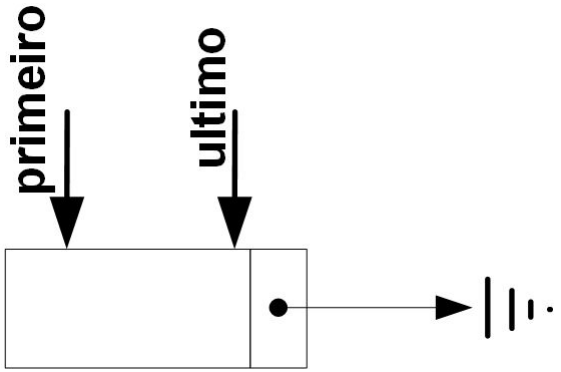
```
class Fila {
    private Celula primeiro, ultimo;
    public Fila () {
        primeiro = new Celula();
        ultimo = primeiro;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```

A primeira célula da nossa fila é o nó cabeça, célula "café com leite" cuja função é eliminar um if no inserir



Classe Fila

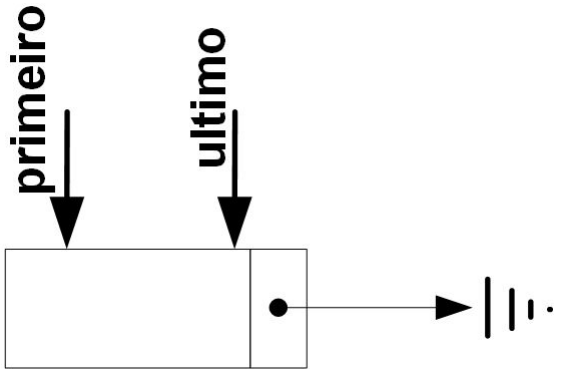
```
class Fila {
    private Celula primeiro, ultimo;
    public Fila () {
        primeiro = new Celula();
        ultimo = primeiro;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```



Inserir (ou Enfileirar)

```
class Fila {
    private Celula primeiro, ultimo;
    public Fila () {
        primeiro = new Celula();
        ultimo = primeiro;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```

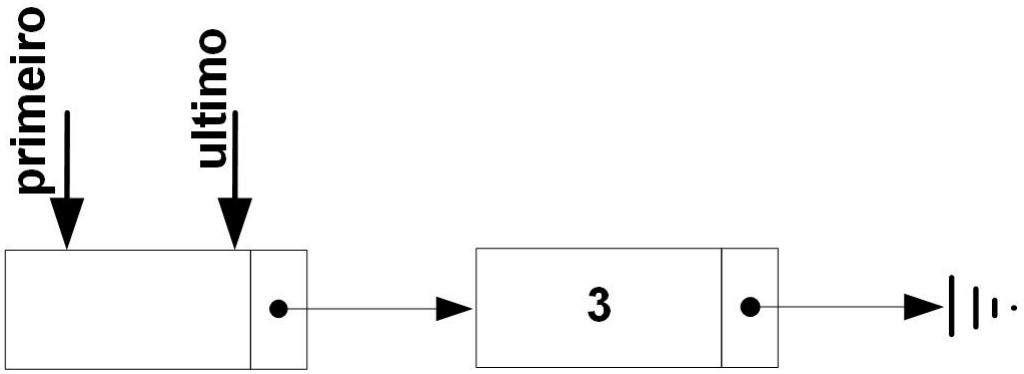
```
public void inserir(int x) {  //Inserir(3)
    ultimo.prox = new Celula(x);
    ultimo = ultimo.prox;
}
```



Inserir (ou Enfileirar)

```
class Fila {
    private Celula primeiro, ultimo;
    public Fila () {
        primeiro = new Celula();
        ultimo = primeiro;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```

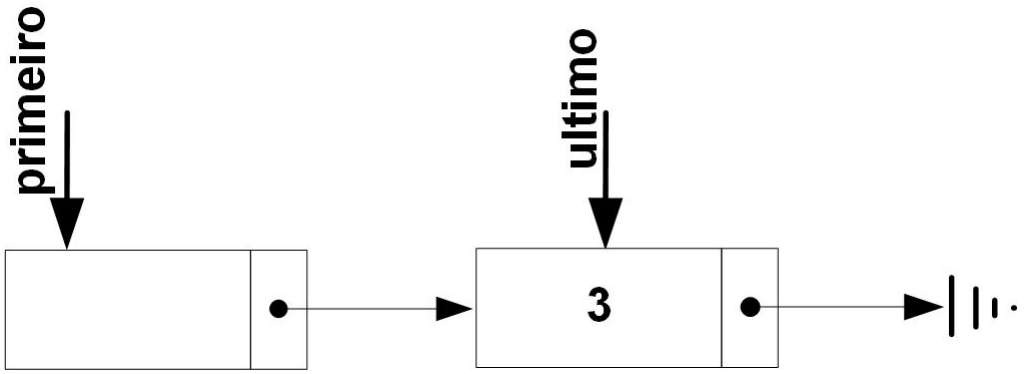
```
public void inserir(int x) {  //Inserir(3)
    ultimo.prox = new Celula(x);
    ultimo = ultimo.prox;
}
```



Inserir (ou Enfileirar)

```
class Fila {
    private Celula primeiro, ultimo;
    public Fila () {
        primeiro = new Celula();
        ultimo = primeiro;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```

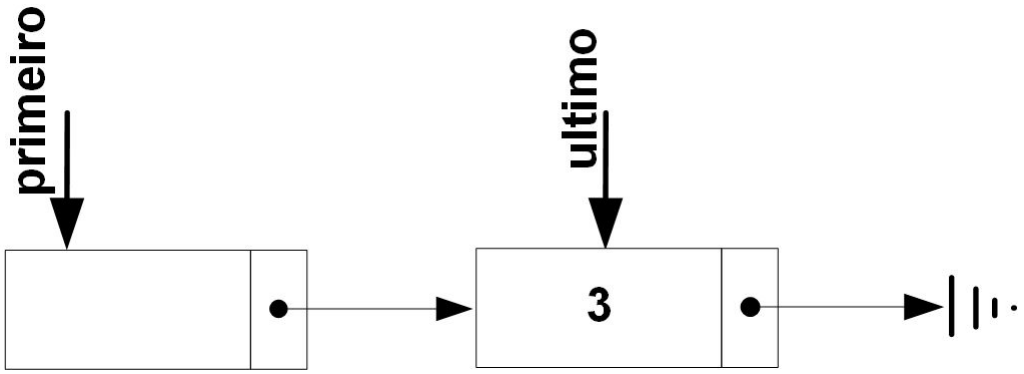
```
public void inserir(int x) {  //Inserir(3)
    ultimo.prox = new Celula(x);
    ultimo = ultimo.prox;
}
```



Inserir (ou Enfileirar)

```
class Fila {
    private Celula primeiro, ultimo;
    public Fila () {
        primeiro = new Celula();
        ultimo = primeiro;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```

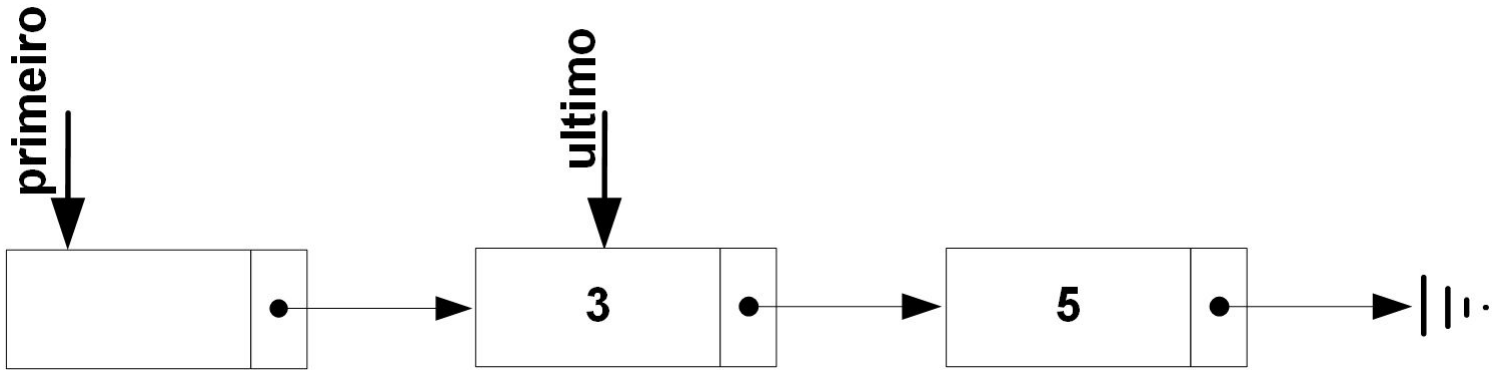
```
public void inserir(int x) {  //Inserir(5)
    ultimo.prox = new Celula(x);
    ultimo = ultimo.prox;
}
```



Inserir (ou Enfileirar)

```
class Fila {
    private Celula primeiro, ultimo;
    public Fila () {
        primeiro = new Celula();
        ultimo = primeiro;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```

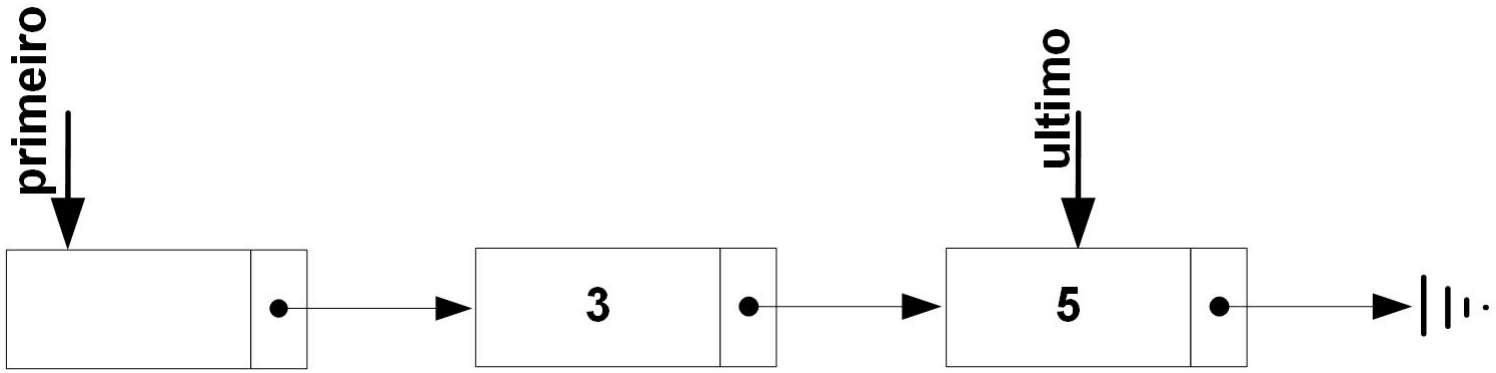
```
public void inserir(int x) {  //Inserir(5)
    ultimo.prox = new Celula(x);
    ultimo = ultimo.prox;
}
```



Inserir (ou Enfileirar)

```
class Fila {
    private Celula primeiro, ultimo;
    public Fila () {
        primeiro = new Celula();
        ultimo = primeiro;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```

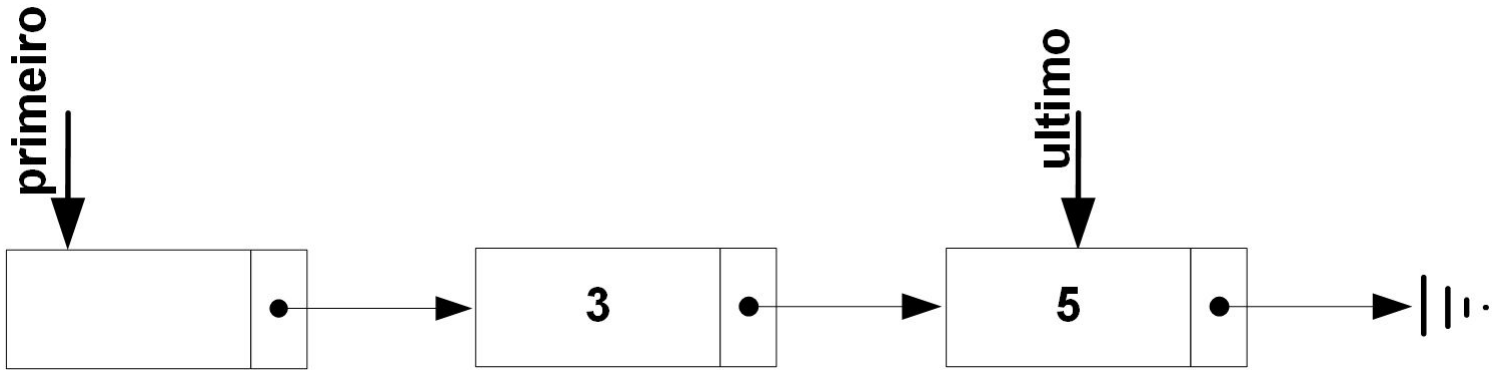
```
public void inserir(int x) {  //Inserir(5)
    ultimo.prox = new Celula(x);
    ultimo = ultimo.prox;
}
```



Inserir (ou Enfileirar)

```
class Fila {
        private Celula primeiro, ultimo;
        public Fila () {
                primeiro = new Celula();
                ultimo = primeiro;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```

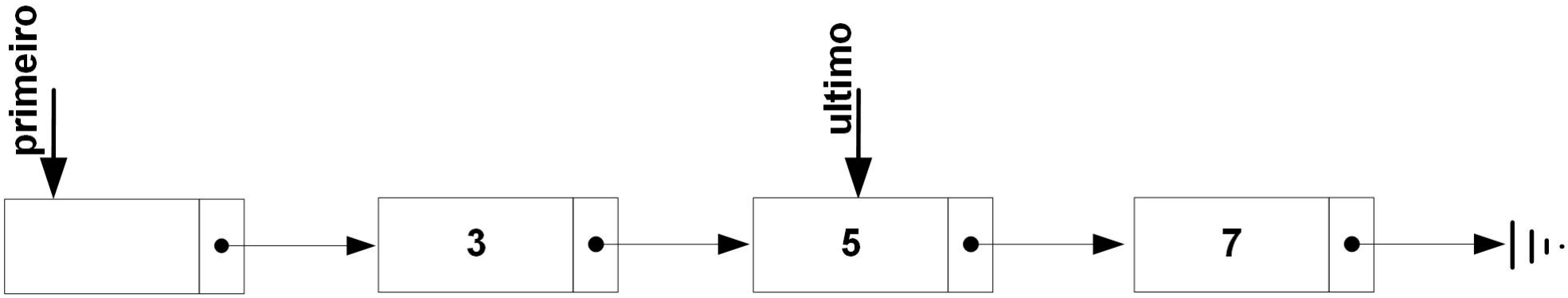
```
public void inserir(int x) {  //Inserir(7)
      ultimo.prox = new Celula(x);
      ultimo = ultimo.prox;
}
```



Inserir (ou Enfileirar)

```
class Fila {
    private Celula primeiro, ultimo;
    public Fila () {
        primeiro = new Celula();
        ultimo = primeiro;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```

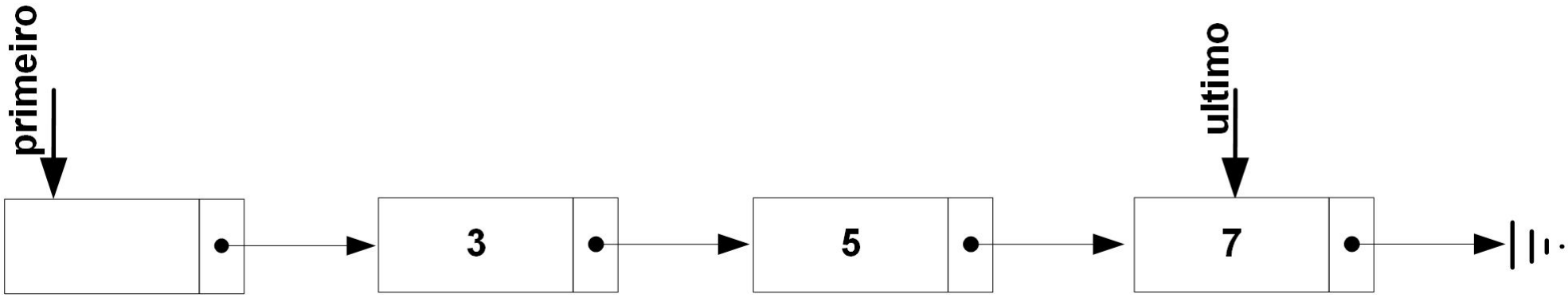
```
public void inserir(int x) {  //Inserir(7)
    ultimo.prox = new Celula(x);
    ultimo = ultimo.prox;
}
```



Inserir (ou Enfileirar)

```
class Fila {
    private Celula primeiro, ultimo;
    public Fila () {
        primeiro = new Celula();
        ultimo = primeiro;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```

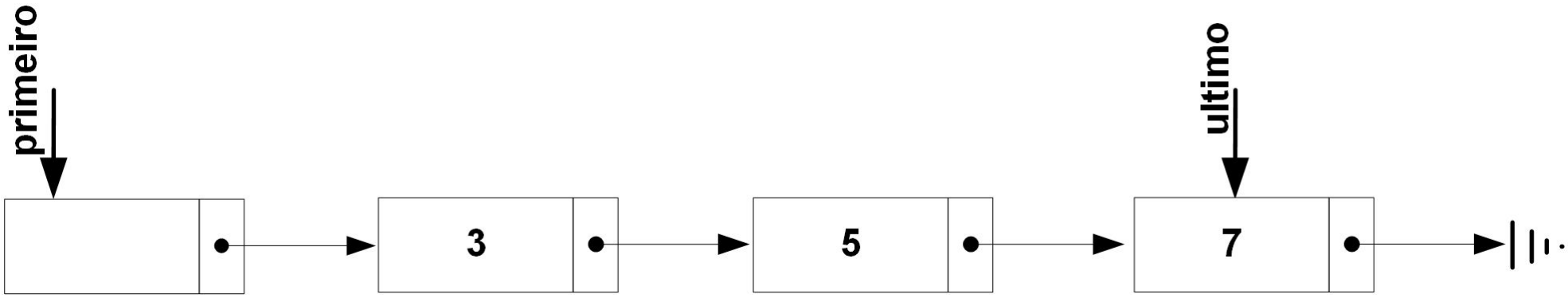
```
public void inserir(int x) {  //Inserir(7)
    ultimo.prox = new Celula(x);
    ultimo = ultimo.prox;
}
```



Remover (ou Desenfileirar)

```
class Fila {
    private Celula primeiro, ultimo;
    public Fila () {
        primeiro = new Celula();
        ultimo = primeiro;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```

```
public int remover() throws Exception{
  if (primeiro == ultimo)
    throw new Exception("Erro!");
  Celula tmp = primeiro;
  primeiro = primeiro.prox;
  int elemento = primeiro.elemento;
  tmp.prox = null;
  tmp = null;
  return elemento;
}
```

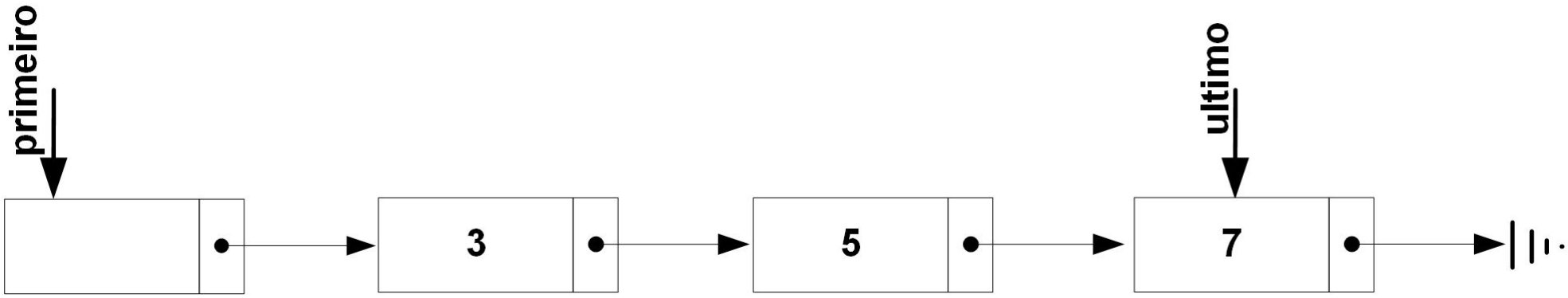


Remover (ou Desenfileirar)

```
class Fila {
    private Celula primeiro, ultimo;
    public Fila () {
        primeiro = new Celula();
        ultimo = primeiro;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```

```
public int remover() throws Exception{
  if (primeiro == ultimo)
    throw new Exception("Erro!");
  Celula tmp = primeiro;
  primeiro = primeiro.prox;
  int elemento = primeiro.elemento;
  tmp.prox = null;
  tmp = null;
  return elemento;
}
```

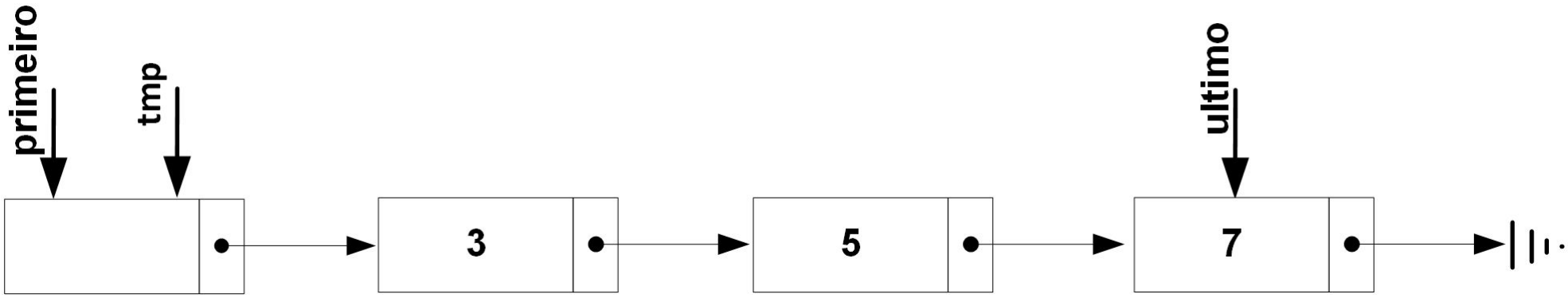
false



Remover (ou Desenfileirar)

```
class Fila {
      private Celula primeiro, ultimo;
      public Fila () {
            primeiro = new Celula();
            ultimo = primeiro;
      }
      public void inserir(int x) { ... }
      public int remover() { ... }
      public void mostrar() { ... }
}
```

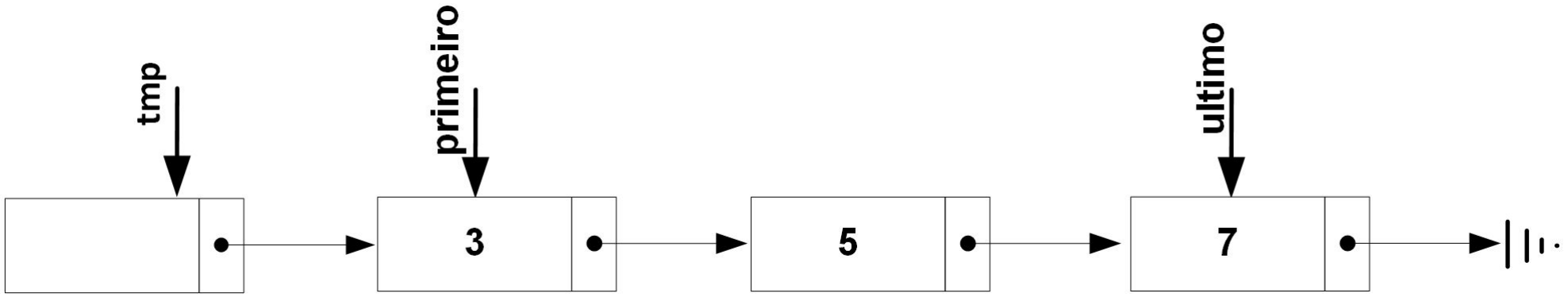
```
public int remover() throws Exception{
   if (primeiro == ultimo)
      throw new Exception("Erro!");
   Celula tmp = primeiro;
   primeiro = primeiro.prox;
   int elemento = primeiro.elemento;
   tmp.prox = null;
   tmp = null;
   return elemento;
}
```



Remover (ou Desenfileirar)

```
class Fila {
      private Celula primeiro, ultimo;
      public Fila () {
            primeiro = new Celula();
            ultimo = primeiro;
      }
      public void inserir(int x) { ... }
      public int remover() { ... }
      public void mostrar() { ... }
}
```

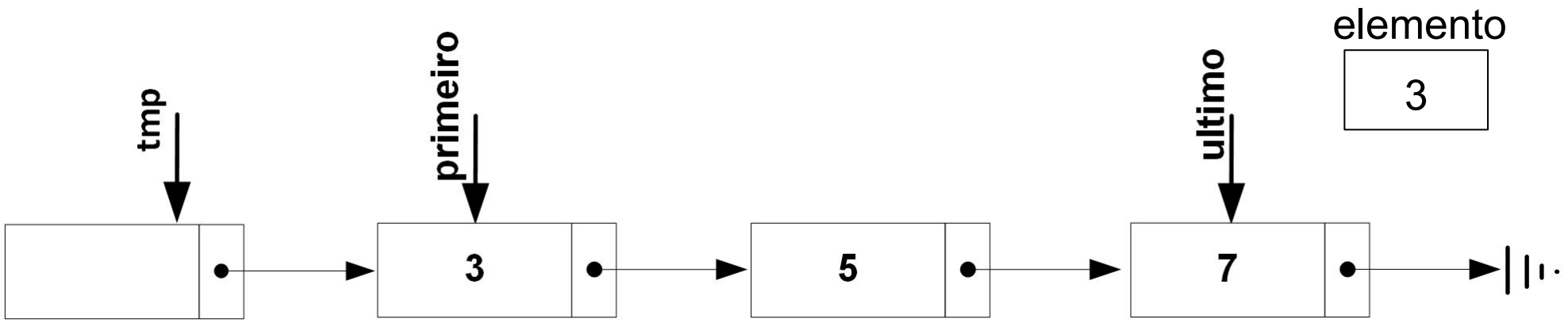
```
public int remover() throws Exception{
   if (primeiro == ultimo)
      throw new Exception("Erro!”);
   Celula tmp = primeiro;
   primeiro = primeiro.prox;
   int elemento = primeiro.elemento;
   tmp.prox = null;
   tmp = null;
   return elemento;
}
```



Remover (ou Desenfileirar)

```
class Fila {
    private Celula primeiro, ultimo;
    public Fila () {
        primeiro = new Celula();
        ultimo = primeiro;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```

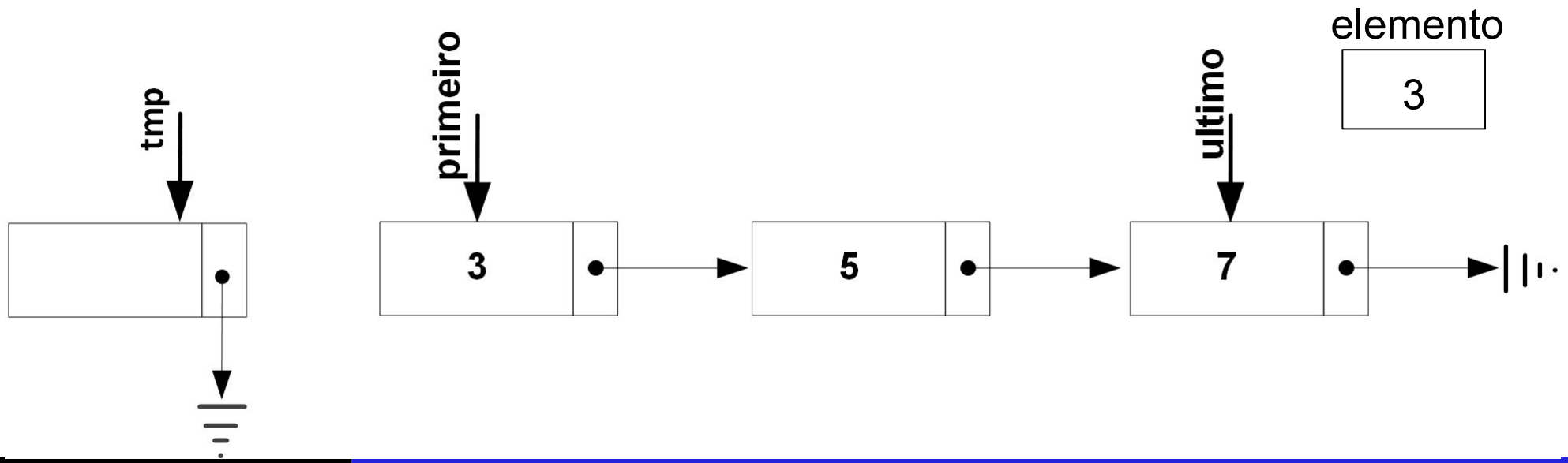
```
public int remover() throws Exception{
  if (primeiro == ultimo)
    throw new Exception("Erro!");
  Celula tmp = primeiro;
  primeiro = primeiro.prox;
  int elemento = primeiro.elemento;
  tmp.prox = null;
  tmp = null;
  return elemento;
}
```



Remover (ou Desenfileirar)

```
class Fila {
      private Celula primeiro, ultimo;
      public Fila () {
            primeiro = new Celula();
            ultimo = primeiro;
      }
      public void inserir(int x) { ... }
      public int remover() { ... }
      public void mostrar() { ... }
}
```

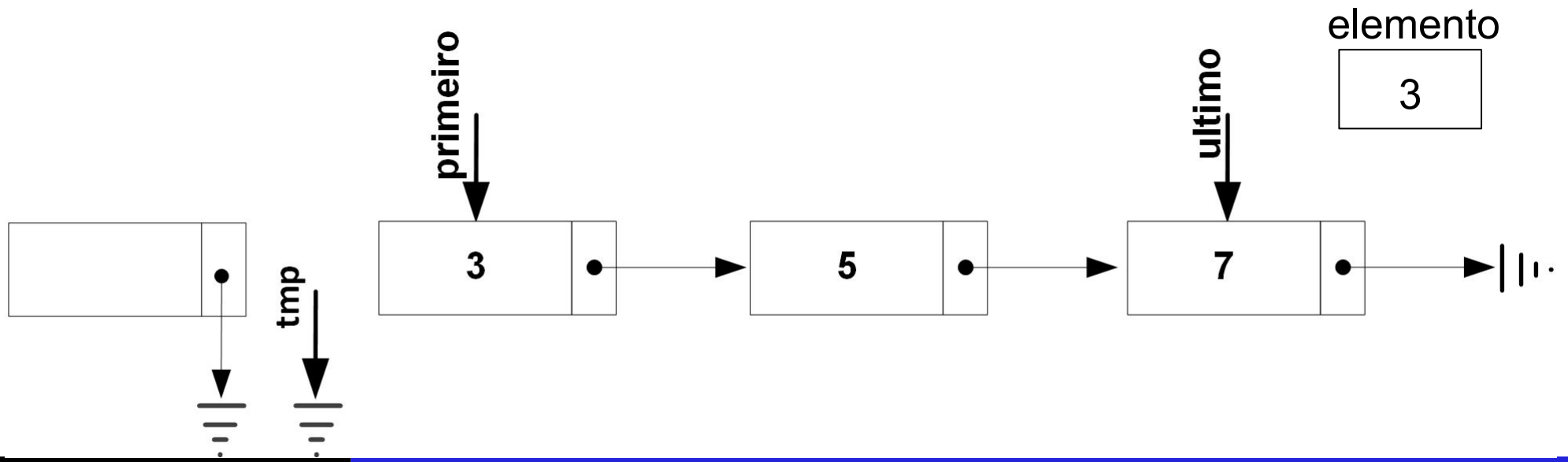
```
public int remover() throws Exception{
   if (primeiro == ultimo)
      throw new Exception("Erro!");
   Celula tmp = primeiro;
   primeiro = primeiro.prox;
   int elemento = primeiro.elemento;
   tmp.prox = null;
   tmp = null;
   return elemento;
}
```



Remover (ou Desenfileirar)

```
class Fila {
      private Celula primeiro, ultimo;
      public Fila () {
            primeiro = new Celula();
            ultimo = primeiro;
      }
      public void inserir(int x) { ... }
      public int remover() { ... }
      public void mostrar() { ... }
}
```

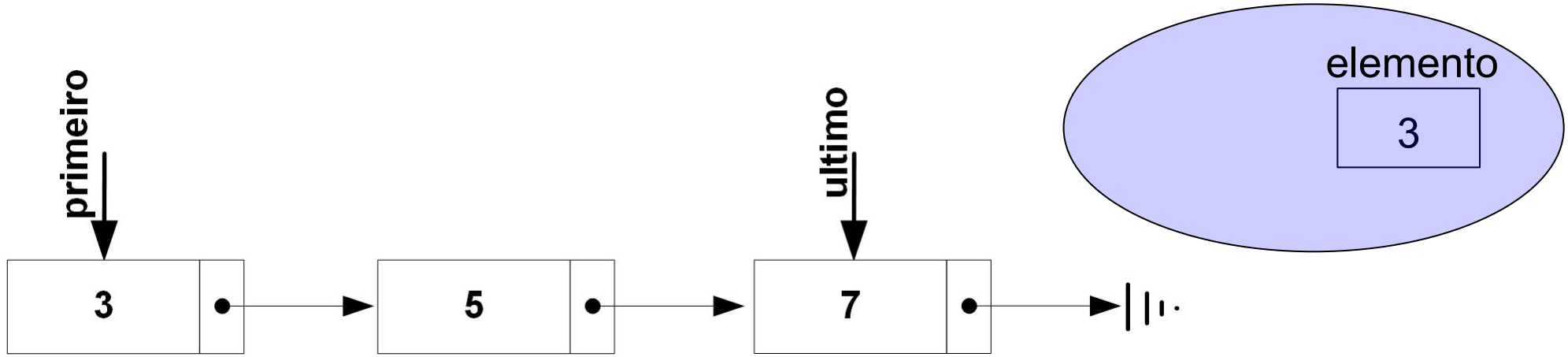
```
public int remover() throws Exception{
   if (primeiro == ultimo)
      throw new Exception("Erro!");
   Celula tmp = primeiro;
   primeiro = primeiro.prox;
   int elemento = primeiro.elemento;
   tmp.prox = null;
   tmp = null;
   return elemento;
}
```



Remover (ou Desenfileirar)

```
class Fila {
      private Celula primeiro, ultimo;
      public Fila () {
            primeiro = new Celula();
            ultimo = primeiro;
      }
      public void inserir(int x) { ... }
      public int remover() { ... }
      public void mostrar() { ... }
}
```

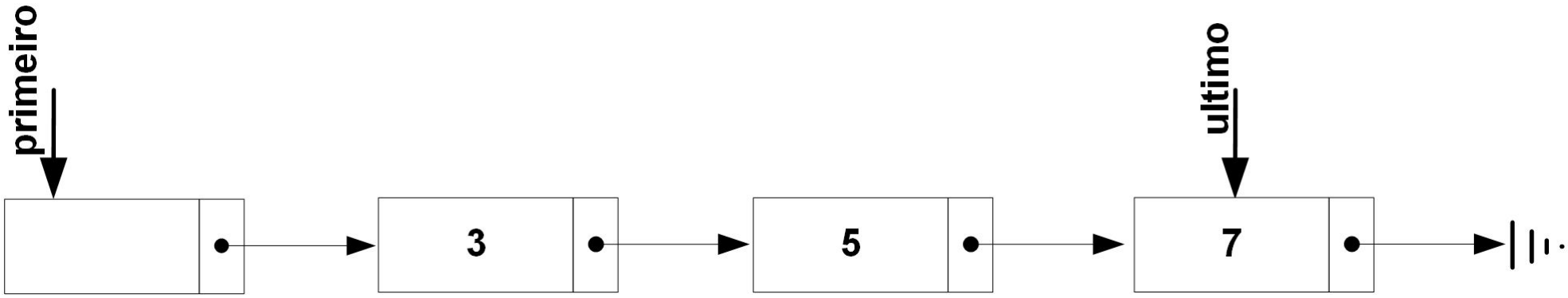
```
public int remover() throws Exception{
   if (primeiro == ultimo)
      throw new Exception("Erro!");
   Celula tmp = primeiro;
   primeiro = primeiro.prox;
   int elemento = primeiro.elemento;
   tmp.prox = null;
   tmp = null;
   return elemento;
}
```



Exercício Resolvido (1)

- Nosso método *remover* remove fisicamente o nó cabeça e faz com que a célula do três seja a cabeça. Como o alteramos para que ele remova fisicamente a célula do três ? **(FAZER AGORA)**

Dica: Execute seu método na fila abaixo

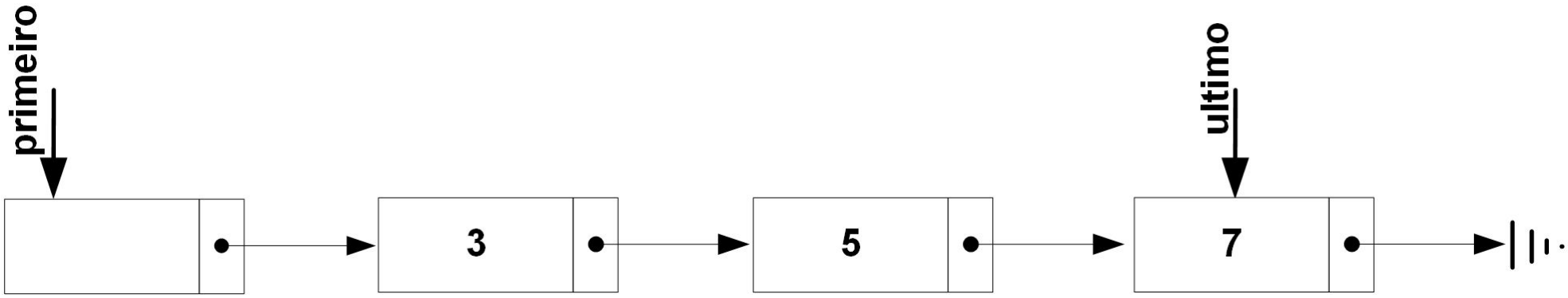


Exercício Resolvido (1)

- Nosso método *remover* remove fisicamente
célula do três seja a cabeça. Como o altera
fisicamente a célula do três ? **(FAZER AGO**

```
public int remover() throws Exception{
   if (primeiro == ultimo)
      throw new Exception("Erro!");
   Celula tmp = primeiro.prox;
   primeiro.prox = primeiro.prox.prox;
   int elemento = tmp.elemento;
   tmp.prox = null;
   tmp = null;
   return elemento;
}
```

Dica: Execute seu método na fila abaixo



Exercício (1)

```
class Fila {
    private Celula primeiro, ultimo;
    public Fila () {
        primeiro = new Celula();
        ultimo = primeiro;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```

```
public void mostrar() {
   // Exercício: Implemente este método
}
```

Exercício Resolvido (2)

- Seja nossa Fila, faça um método que retorne o maior elemento contido na mesma **(FAZER AGORA)**

Exercício Resolvido (2)

- Seja nossa Fila, faça um método que retorne o maior elemento contido na mesma **(FAZER AGORA)**

```
int maior() throws Exception {
int maior = - 1;
if (primeiro == ultimo) {
      throw new Exception("Erro!”);
} else {
            maior = primeiro.prox.elemento;
            Celula i = primeiro.prox.prox;
            while (i != null){
                  if (i.elemento > maior) {
                        maior = i.elemento;
                  }
                  i = i.prox;
      }     }
      return maior;
}
```

Exercício Resolvido (3)

- Seja nossa Fila, faça um método para retornar o terceiro elemento supondo que o mesmo existe **(FAZER AGORA)**

Exercício Resolvido (3)

- Seja nossa Fila, faça um método para retornar o terceiro elemento supondo que o mesmo existe **(FAZER AGORA)**

```
int retornarTerceiroElemento(){
    return (primeiro.prox.prox.prox.elemento);
}
```

Exercício Resolvido (4)

- Seja nossa Fila, faça um método que soma o conteúdo dos elementos contidos na mesma

Exercício Resolvido (4)

- Seja nossa Fila, faça um método que soma o conteúdo dos elementos contidos na mesma

```
int somar() {
int resp = 0;
    for (Celula i = primeiro.prox; i != null; i = i.prox) {
        resp += i.elemento;
    }
    return resp;
}
```

Exercício Resolvido (5)

- Seja nossa Fila, faça um método que inverta a ordem dos seus elementos

Exercício Resolvido (5)

- Seja nossa Fila, faça um método que inverta a ordem dos seus elementos

```
void inverter () {
    Celula fim = ultimo;
    while (primeiro != fim){
        Celula nova = new Celula (primeiro.prox.elemento);
        nova.prox = fim.prox;
        fim.prox = nova;
        Celula tmp = primeiro.prox;
        primeiro.prox = tmp.prox;
        nova = tmp = tmp.prox = null;
        if (ultimo == fim) {ultimo = ultimo.prox;    }
    }
    fim = null;
}
```

Exercício Resolvido (6)

- Seja nossa Fila, faça um método recursivo para contar o número de elementos pares AND múltiplos de cinco contidos na fila

Exercício Resolvido (6)

- Seja nossa Fila, faça um método recursivo para contar o número de elementos pares AND múltiplos de cinco contidos na fila

```
int contar() { return contar(primeiro.prox);    }
int contar(Celula i){
      int resp;
      if (i == null){

      resp = 0;
      } else if (i.elemento % 2 == 0 && i.elemento % 5 == 0){
            resp = 1 + contar(i.prox);
      } else {
            resp = contar(i.prox);
      }
      return resp;
}
```

Exercício (2)

- Seja nossa Fila, mostre graficamente a execução do código abaixo supondo que a fila contém 5, 10, 15, 20 e 25, respectivamente

```
        void metodoDoidao () {
        Celula fim = ultimo;
              while (primeiro != fim) {
                    ultimo.prox = new Celula (primeiro.prox.elemento);
                    Celula tmp = primeiro;
                    primeiro = primeiro.prox;
                    tmp = tmp.prox = null;
                    ultimo = ultimo.prox;
              }
              fim = null;
  }
```

Exercício Resolvido (7)

- Implemente o método *Celula toFila(Celula topo)* que recebe o endereço de memória da primeira posição de uma pilha flexível e retorna o endereço de memória do nó cabeça de uma fila flexível contendo os elementos da pilha na ordem em que os mesmos foram inseridos na pilha. Seu método deve percorrer a pilha e inserir cada elemento da mesma na nova fila a ser retornada. A pilha não pode ser destruída.

Exercício Resolvido (7)

- Implemente o método *Celula toFila(Celula topo)* que recebe o endereço de memória
ereço de
memória
da pilha
na order
o deve
percorre
ser
retornad

```
Celula toFila(Celula topo){
    Celula primeiro = new Celula();
    Celula ultimo = primeiro;
    ultimo = toFila(topo, ultimo);
    return primeiro;
}
Celula toFila(Celula i, Celula ultimo){
    if (i != null){
        ultimo = toFila(i.prox, ultimo);
        ultimo.prox = new Celula(i.elemento);
        ultimo = ultimo.prox;
    }
    return ultimo;
}
```

Exercício (3)

- Implemente a fila flexível sem nó cabeça

Exercício (4)

- Implemente a pilha flexível com nó cabeça

Exercício Resolvido (8)

- Implemente uma fila sem a existência do “ponteiro” último

Exercício Resolvido (8)

- Implemente uma fila sem a existência do “ponteiro” último

```
public class Fila {
   private Celula primeiro;
   public Fila() {
      primeiro = new Celula();
   }
   public void void inserir(int x) {
      Celula i;
      for (i = primeiro; i.prox != null; i = i.prox);
      i.prox = new Celula(x);
      i = null;
   }
   public int remover() throws Exception {
      if (primeiro == ultimo) throw new Exception("Erro ao remover!");
      Celula tmp = primeiro;
      primeiro = primeiro.prox;
      int resp = primeiro.elemento;
      tmp.prox = null;
      tmp = null;
      return resp;
} }
```