Aula anterior

- Introdução à tipo Flexível de abstrato de dados

Tipo Abstrato de Dados (FILA)

Prof. Diego Silva Caldeira Rocha

Introdução

- As filas são um Tipo Abstrato de Dados (TAD) no qual o

primeiro elemento que entra é o primeiro a sair

  - ***F**irst **I**n, **F**irst **O**ut* (FIFO)
  - ***F**irst Came, **F**irst **S**erved* (FCFS)

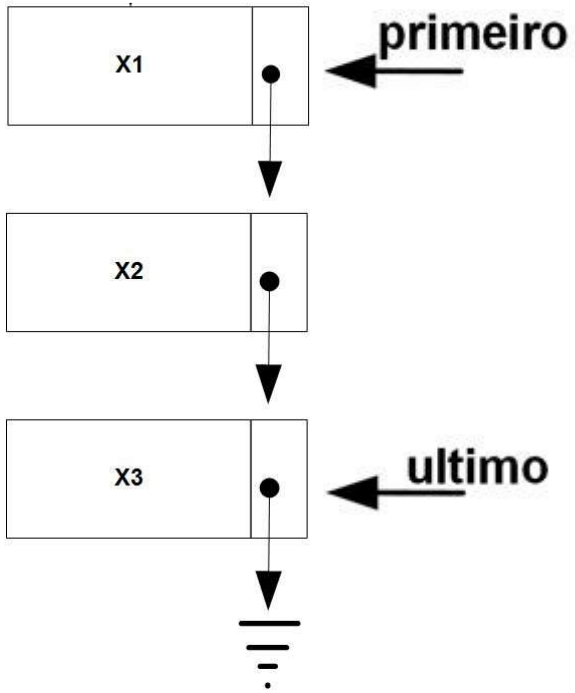
- Tem basicamente os métodos de inserir (enfileirar, *enqueue*) e remover (desenfileirar, *dequeue*)

Exemplos

Código Fonte

- PrincipalFila.java, igual ao da estrutura sequencial

- Fila.java, criará instâncias como:



Classe Fila

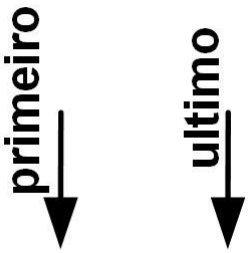
```
class Fila {
    private Celula primeiro, ultimo;
    public Fila () {
        primeiro = new Celula();
        ultimo = primeiro;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```

Classe Fila Genérica

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

Classe Fila

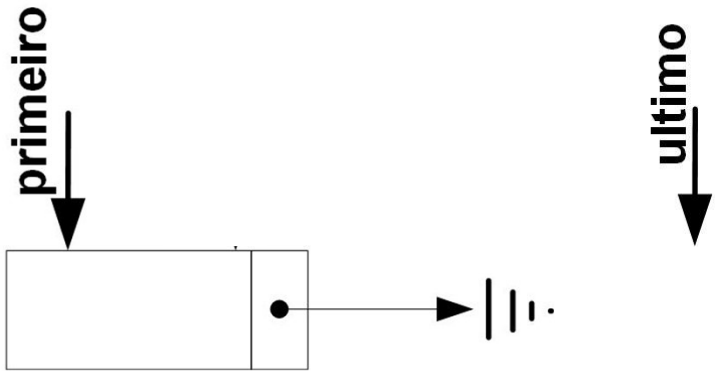
```
class Fila {
    private Celula primeiro, ultimo;
    public Fila () {
        primeiro = new Celula();
        ultimo = primeiro;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```



Classe Fila

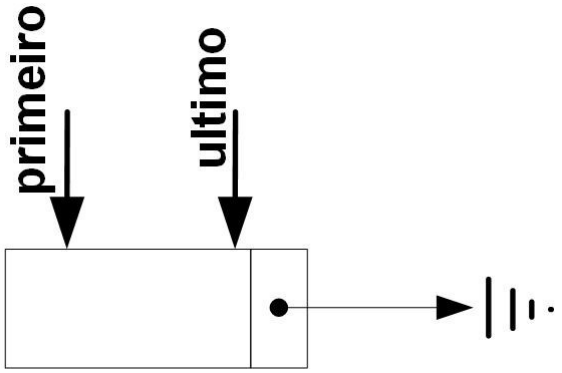
```
class Fila {
    private Celula primeiro, ultimo;
    public Fila () {
        primeiro = new Celula();
        ultimo = primeiro;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```

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



Classe Fila

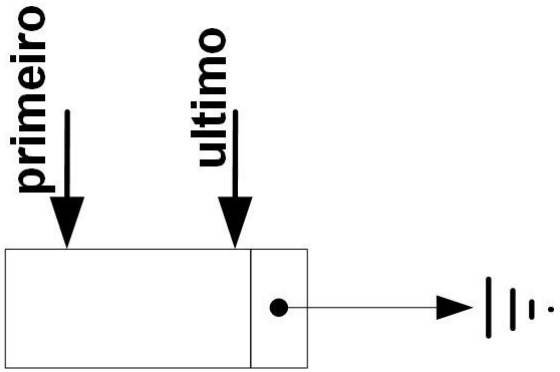
```
class Fila {
    private Celula primeiro, ultimo;
    public Fila () {
        primeiro = new Celula();
        ultimo = primeiro;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```



Inserir (ou Enfileirar ou Enqueue)

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

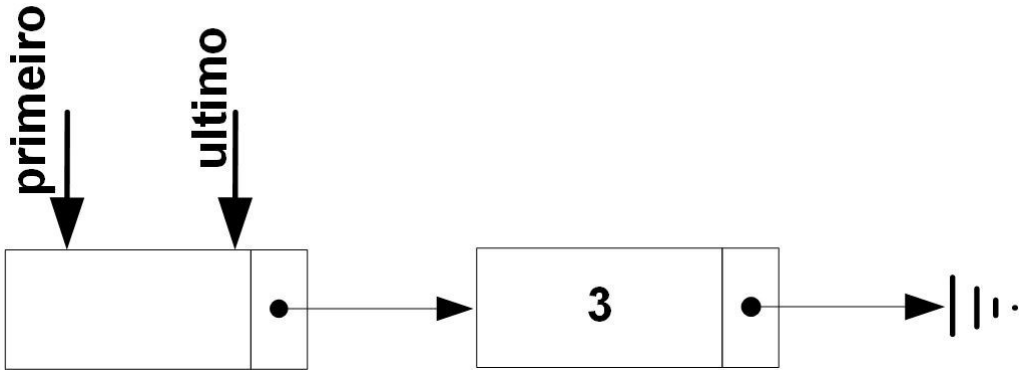
```
public void inserir(int x) {  //Inserir(3)
    ultimo.prox = new Celula(x);
    ultimo = ultimo.prox;
}
```



Inserir (ou Enfileirar ou Enqueue)

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

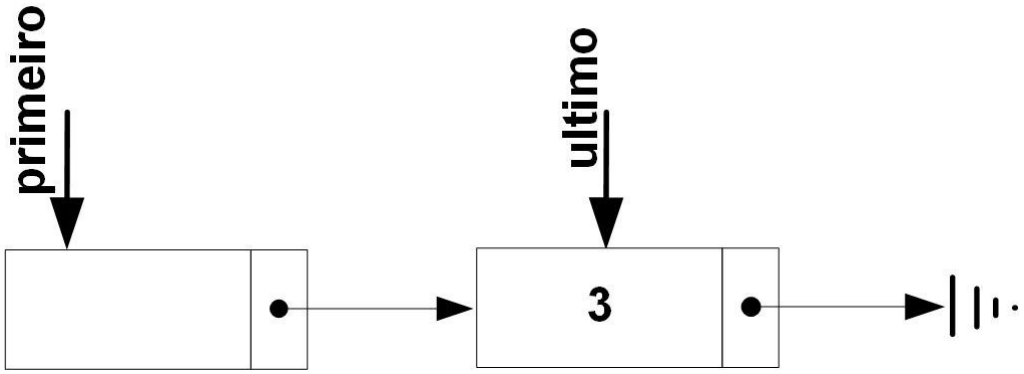
```
public void inserir(int x) {  //Inserir(3)
    ultimo.prox = new Celula(x);
    ultimo = ultimo.prox;
}
```



Inserir (ou Enfileirar ou Enqueue)

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

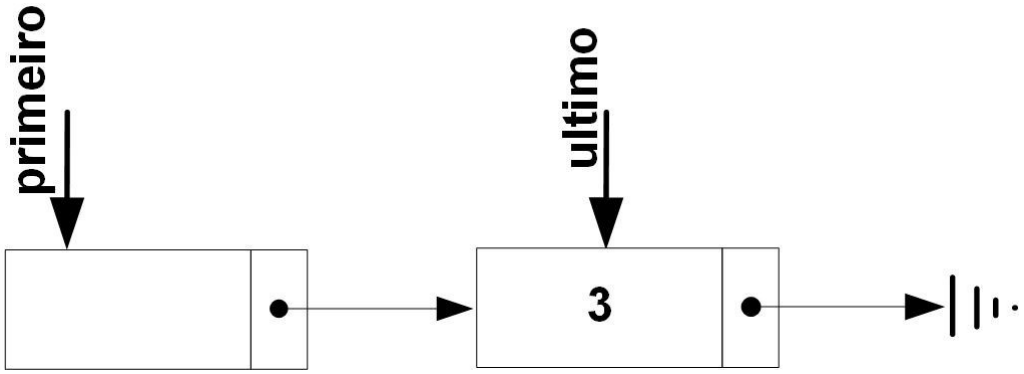
```
public void inserir(int x) {  //Inserir(3)
    ultimo.prox = new Celula(x);
    ultimo = ultimo.prox;
}
```



Inserir (ou Enfileirar ou Enqueue)

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

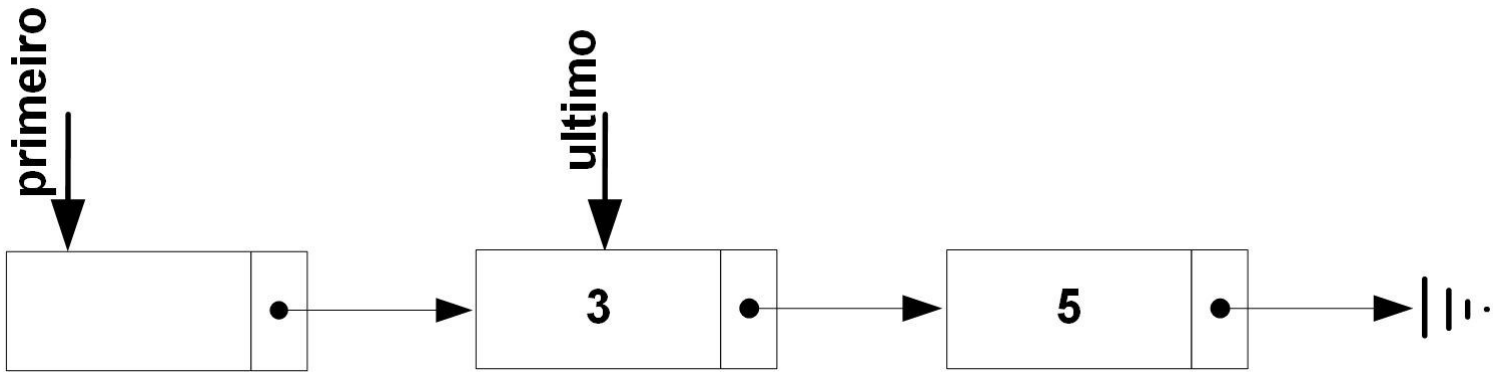
```
public void inserir(int x) {  //Inserir(5)
    ultimo.prox = new Celula(x);
    ultimo = ultimo.prox;
}
```



Inserir (ou Enfileirar ou Enqueue)

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

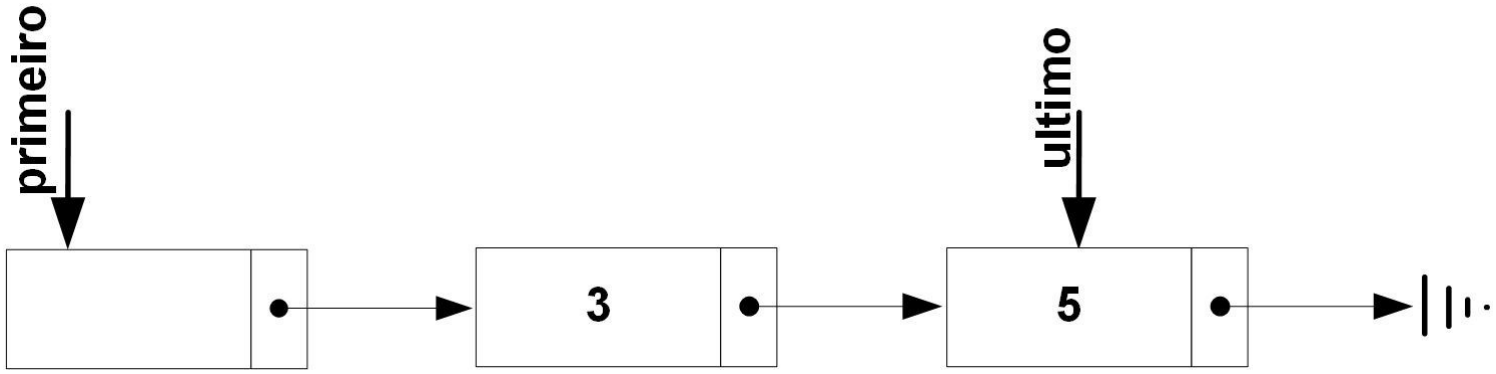
```
public void inserir(int x) {  //Inserir(5)
    ultimo.prox = new Celula(x);
    ultimo = ultimo.prox;
}
```



Inserir (ou Enfileirar ou Enqueue)

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

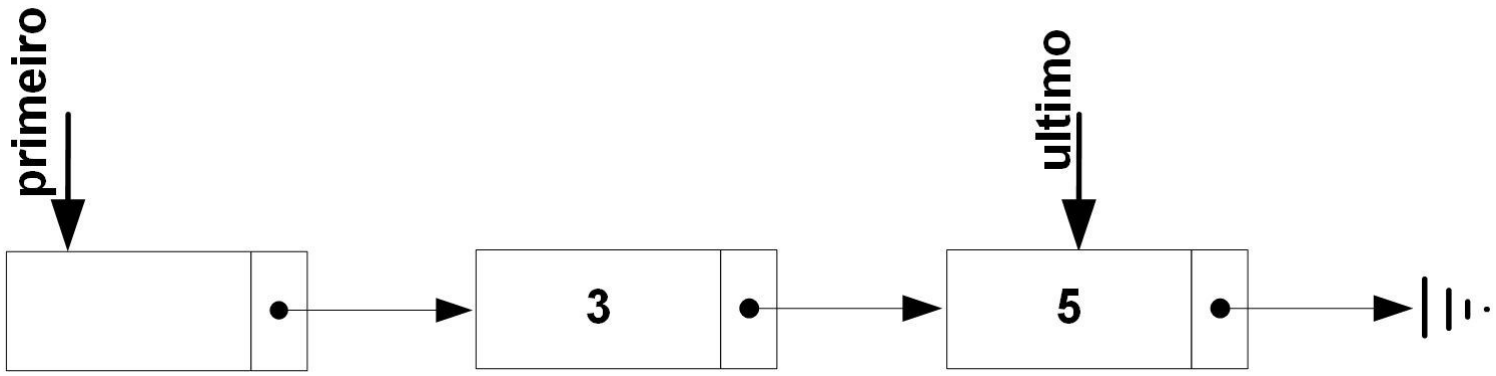
```
public void inserir(int x) {  //Inserir(5)
    ultimo.prox = new Celula(x);
    ultimo = ultimo.prox;
}
```



Inserir (ou Enfileirar ou Enqueue)

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

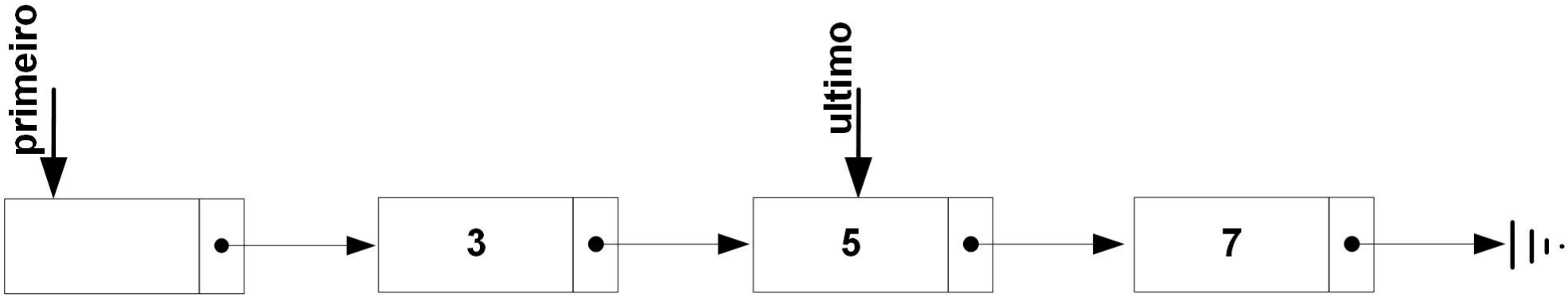
```
public void inserir(int x) {  //Inserir(7)
    ultimo.prox = new Celula(x);
    ultimo = ultimo.prox;
}
```



Inserir (ou Enfileirar ou Enqueue)

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

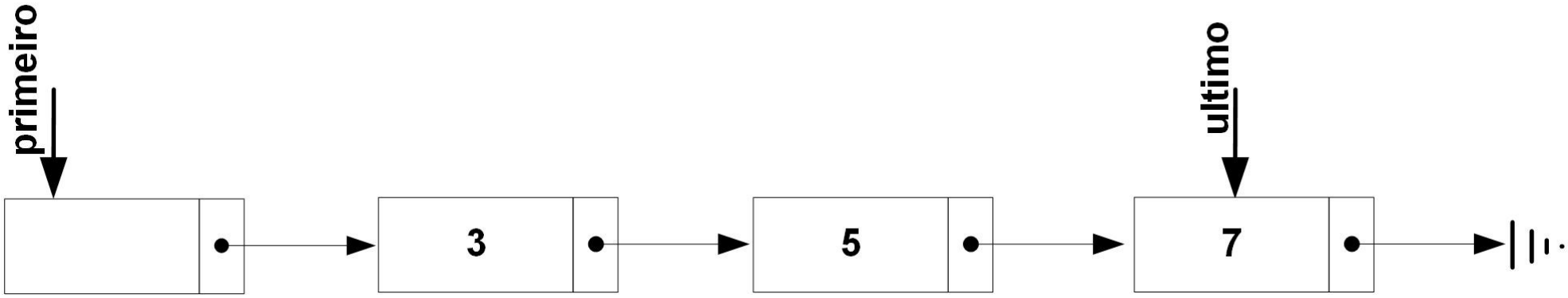
```
public void inserir(int x) {  //Inserir(7)
    ultimo.prox = new Celula(x);
    ultimo = ultimo.prox;
}
```



Inserir (ou Enfileirar ou Enqueue)

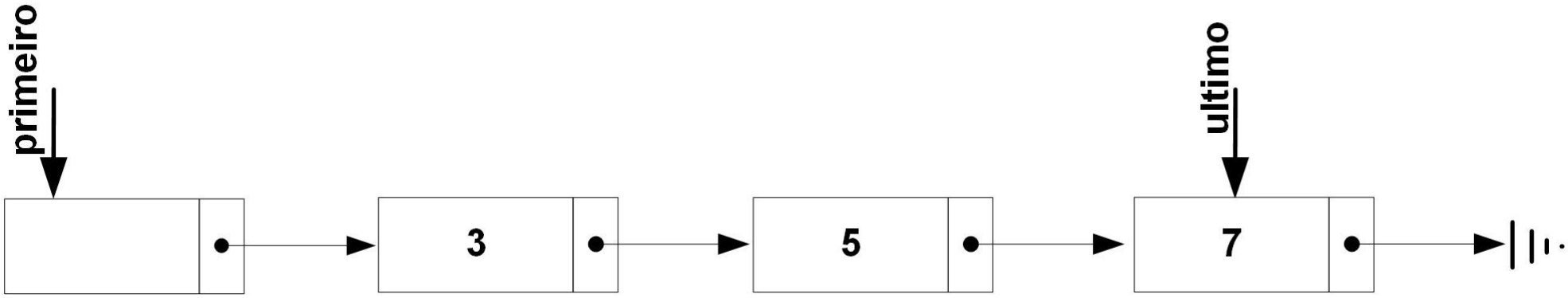
```
class Fila {
    private Celula primeiro, ultimo;
    public Fila () {
        primeiro = new Celula();
        ultimo = primeiro;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```

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



Remover (ou Desenfileirar ou Dequeue)

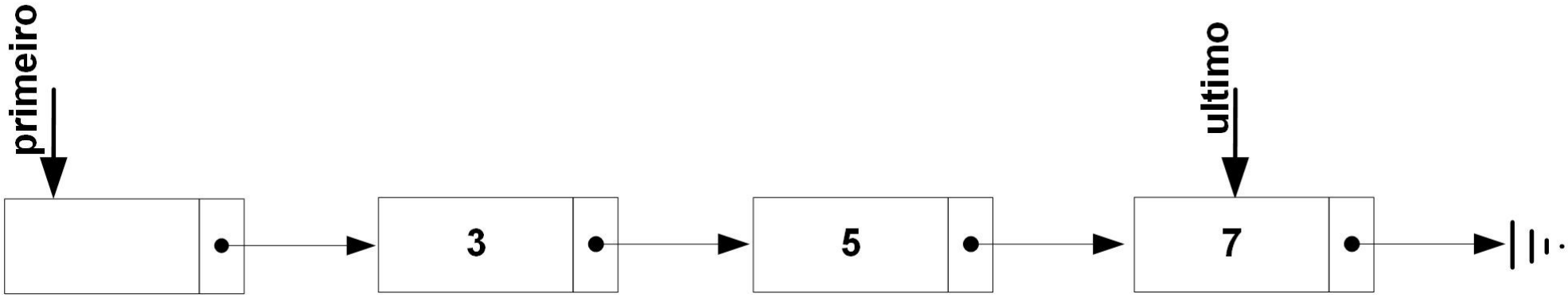
```
class Fila {
    private Celula primeiro, ultimo;
    public Fila () {
        primeiro = new Celula();
        ultimo = primeiro;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```



Remover (ou Desenfileirar ou Dequeue)

```
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;
}
```

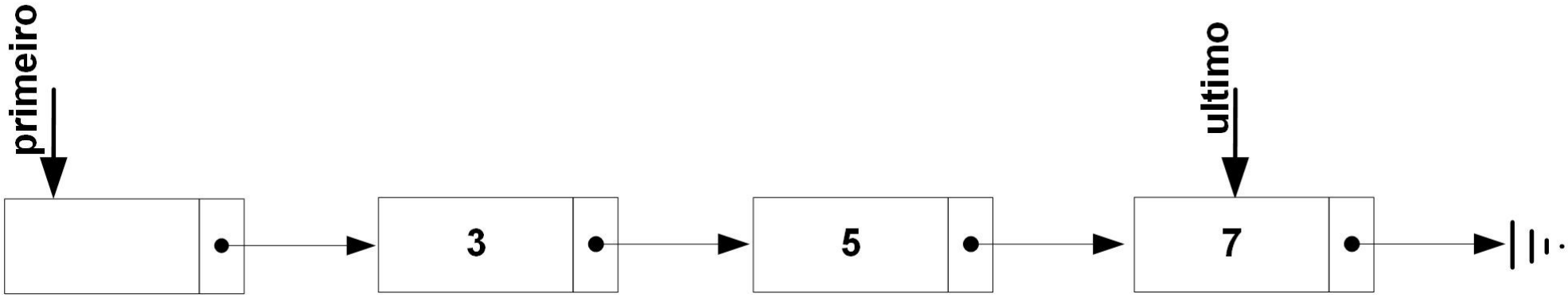


Remover (ou Desenfileirar ou Dequeue)

```
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;
}
```

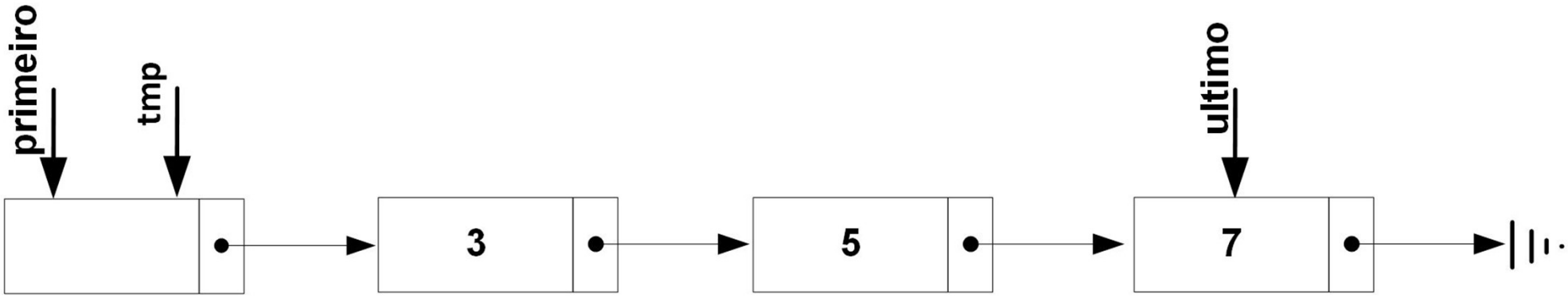
false



Remover (ou Desenfileirar ou Dequeue)

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

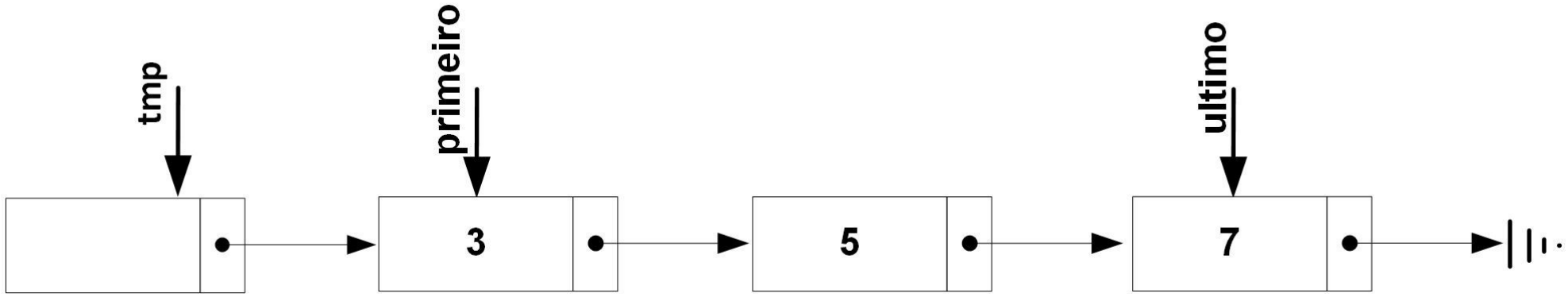
```
public int remover() throws Exception{
  if (primeiro == ultimo)
    throw new Exception("Erro!");
  Celula tmp = primeiro;
  primeiro = primeiro.prox;
  int elemento = primeiro.elemento;
  tmp.prox = null;
  tmp = null;
  return elemento;
}
```



Remover (ou Desenfileirar ou Dequeue)

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

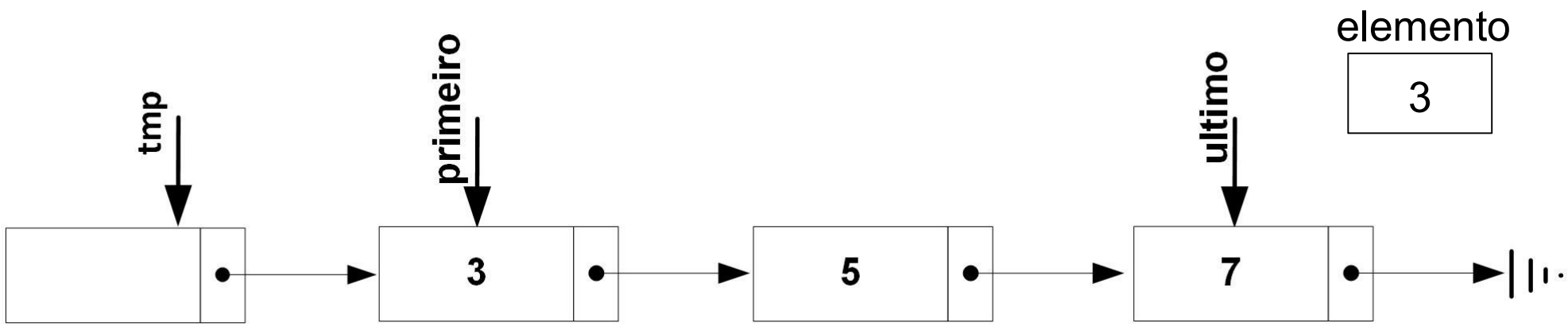
```
public int remover() throws Exception{
  if (primeiro == ultimo)
    throw new Exception("Erro!");
  Celula tmp = primeiro;
  primeiro = primeiro.prox;
  int elemento = primeiro.elemento;
  tmp.prox = null;
  tmp = null;
  return elemento;
}
```



Remover (ou Desenfileirar ou Dequeue)

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

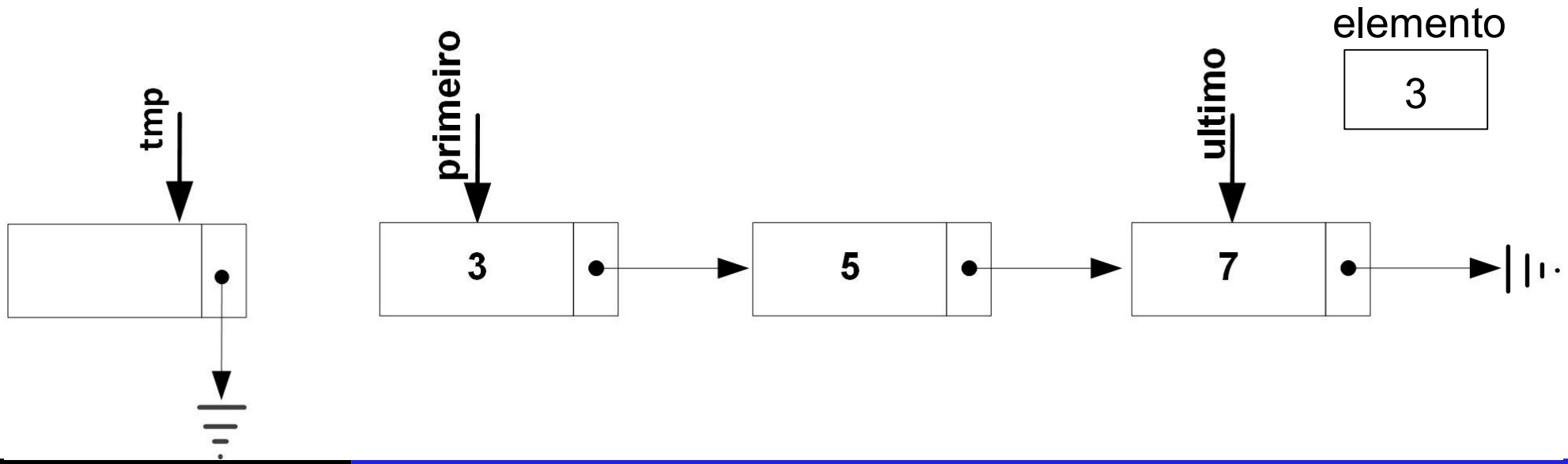
```
public int remover() throws Exception{
  if (primeiro == ultimo)
    throw new Exception("Erro!");
  Celula tmp = primeiro;
  primeiro = primeiro.prox;
  int elemento = primeiro.elemento;
  tmp.prox = null;
  tmp = null;
  return elemento;
}
```



Remover (ou Desenfileirar ou Dequeue)

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

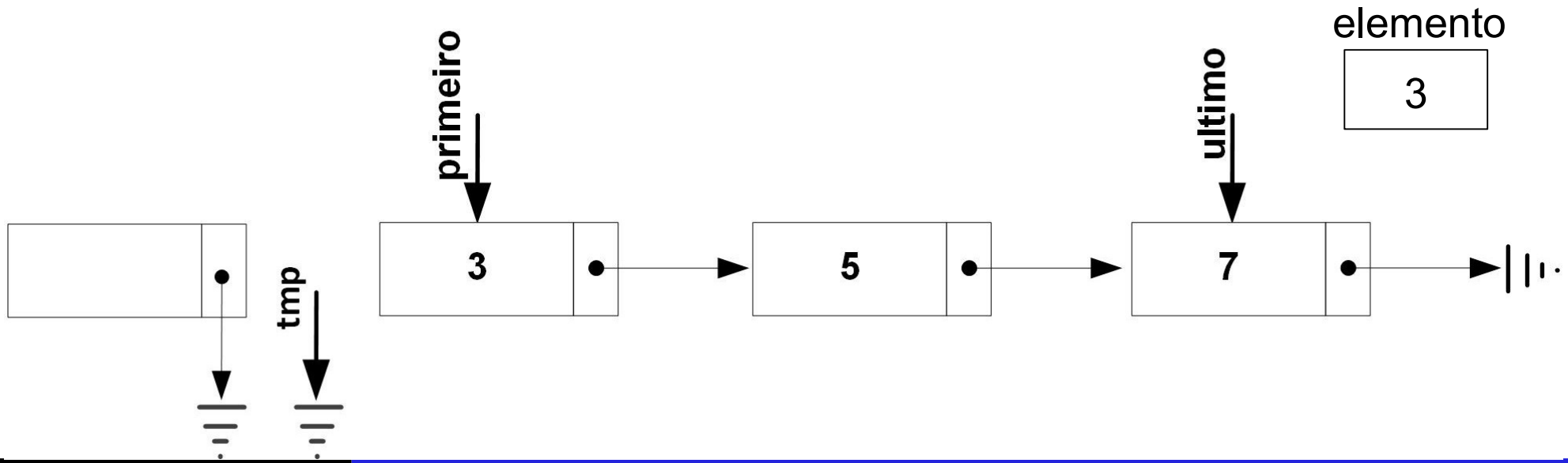
```
public int remover() throws Exception{
   if (primeiro == ultimo)
      throw new Exception("Erro!");
   Celula tmp = primeiro;
   primeiro = primeiro.prox;
   int elemento = primeiro.elemento;
   tmp.prox = null;
   tmp = null;
   return elemento;
}
```



Remover (ou Desenfileirar ou Dequeue)

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

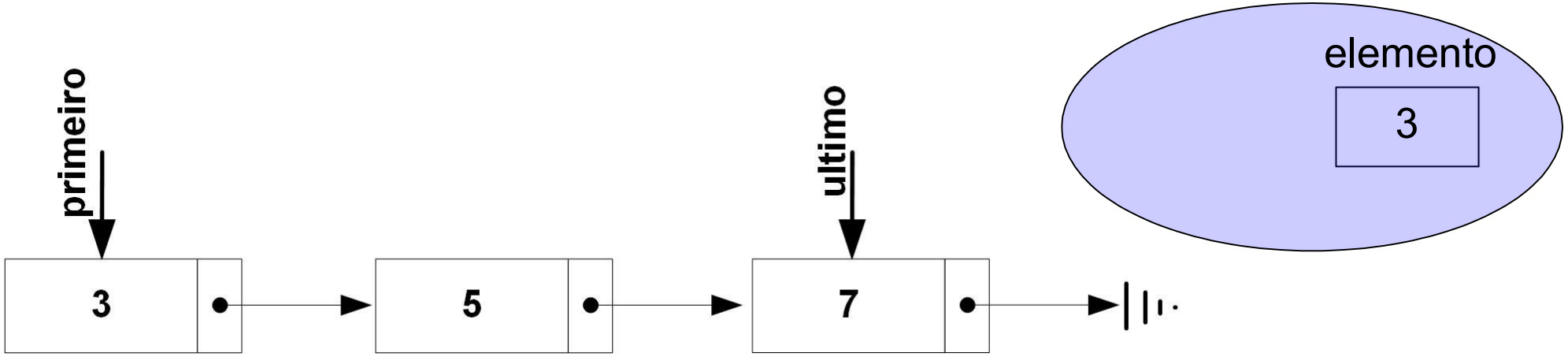
```
public int remover() throws Exception{
  if (primeiro == ultimo)
    throw new Exception("Erro!");
  Celula tmp = primeiro;
  primeiro = primeiro.prox;
  int elemento = primeiro.elemento;
  tmp.prox = null;
  tmp = null;
  return elemento;
}
```



Remover (ou Desenfileirar ou Dequeue)

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

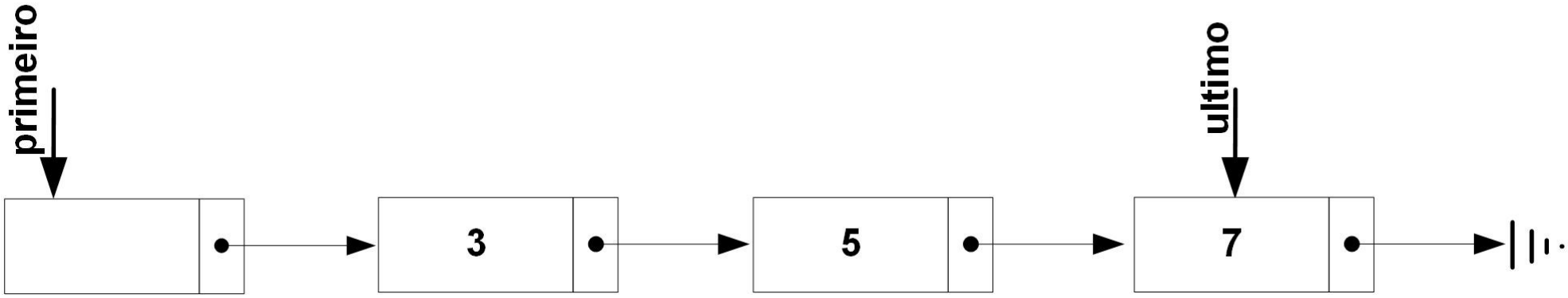
```
public int remover() throws Exception{
  if (primeiro == ultimo)
    throw new Exception("Erro!");
  Celula tmp = primeiro;
  primeiro = primeiro.prox;
  int elemento = primeiro.elemento;
  tmp.prox = null;
  tmp = null;
  return elemento;
}
```



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

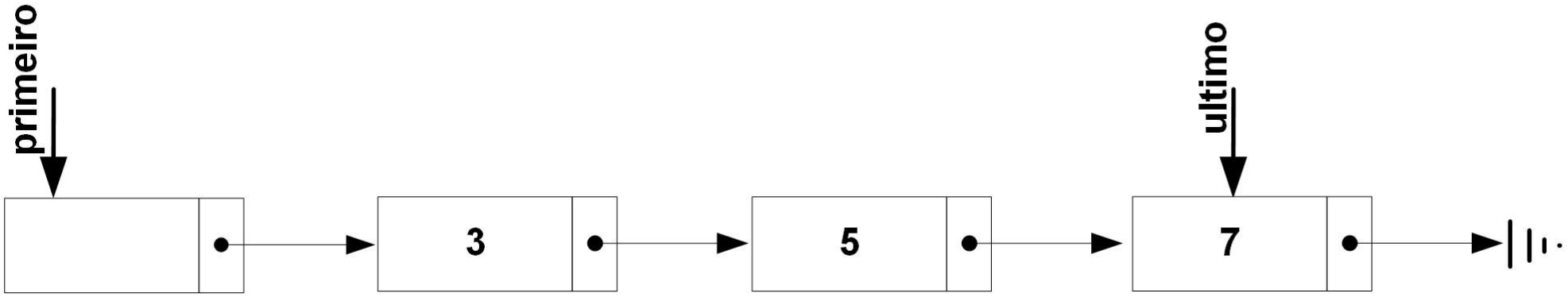


Exercício Resolvido (1)

- Nosso método *remover* remove fisicamente [illegible] célula do três seja a cabeça. Como o alter[illegible] fisicamente a célula do três ? **(FAZER AG C**

```
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!”);
    }
    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.prox != 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 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 {
\\primeiro.prox==null
      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;
} }
```

Exercício (9)

- Implemente a fila flexível sem nó cabeça (Sentinela)

Próxima aula

- Introdução à tipo Flexível de abstrato de dados - Pilha