# Algoritmos e Estruturas de Dados

## Coleções Não Iteráveis de Referências para Objetos

**Autores:**
Carlos Urbano
Catarina Reis
João Ramos
José Magno
Marco Ferreira

# Coleções Não Iteráveis

- Uma coleção não iterável de referências para objetos é um tipo abstrato de dados (ADT) definido pela interface **ColecaoNaoIteravel<T>**

```
public interface ColecaoNaoIteravel<T> extends Colecao<T> {
    void inserir(T elem);

    T remover();

    T consultar();

    boolean isVazia();
}
```

# Algoritmos e Estruturas de Dados

## Filas
## de Referências para Objetos

**Autores:**
Carlos Urbano
Catarina Reis
João Ramos
José Magno
Marco Ferreira

# Filas

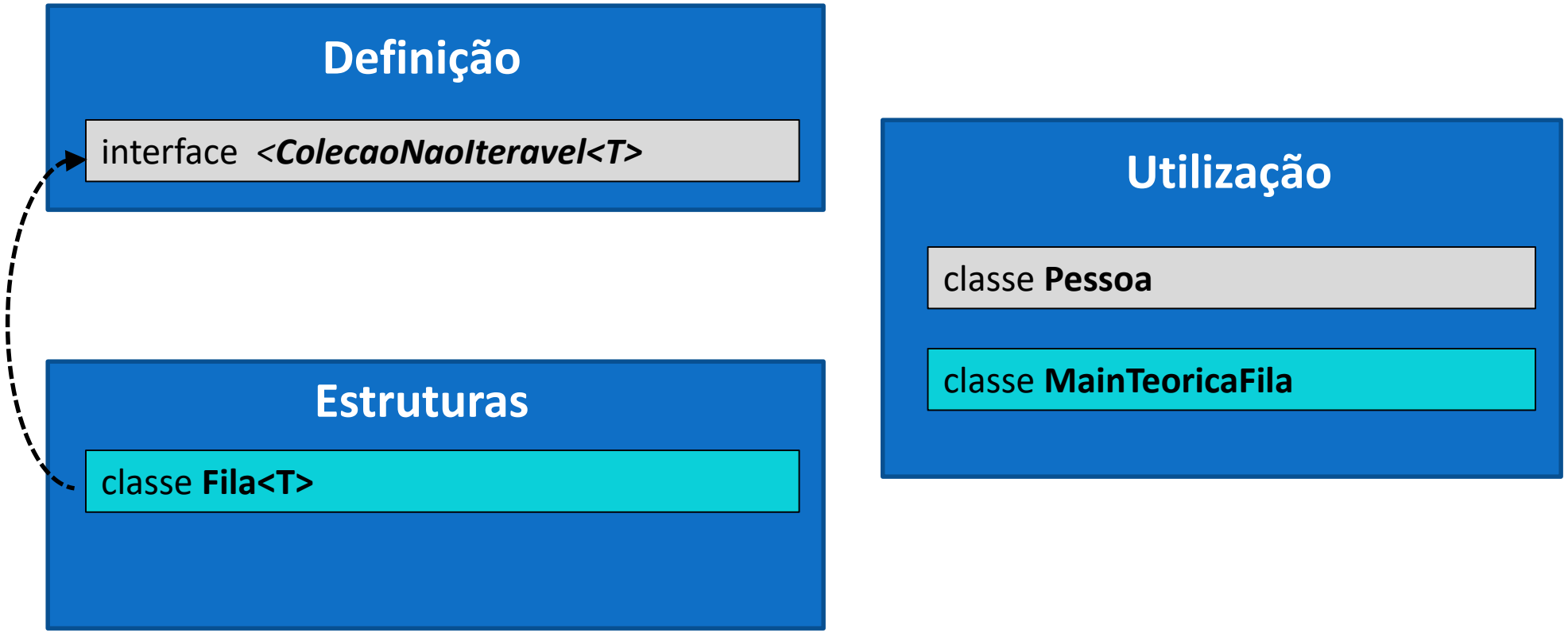
- Uma **fila** de referências para objetos é uma estrutura abstrata de dados (ADT), que obedece à regra seguinte: os elementos são retirados e consultados pela ordem por que foram inseridos
- Para isso, os elementos são inseridos no fim da fila e consultados/retirados do seu início
- As filas são também conhecidas por estruturas do tipo FIFO (*first in first out*)

# Filas

- Operações básicas sobre Filas:
  - criar uma fila vazia
  - inserir um elemento na cauda da fila
  - remover o elemento da frente da fila
  - consultar elemento da frente da fila
  - verificar se a fila está vazia

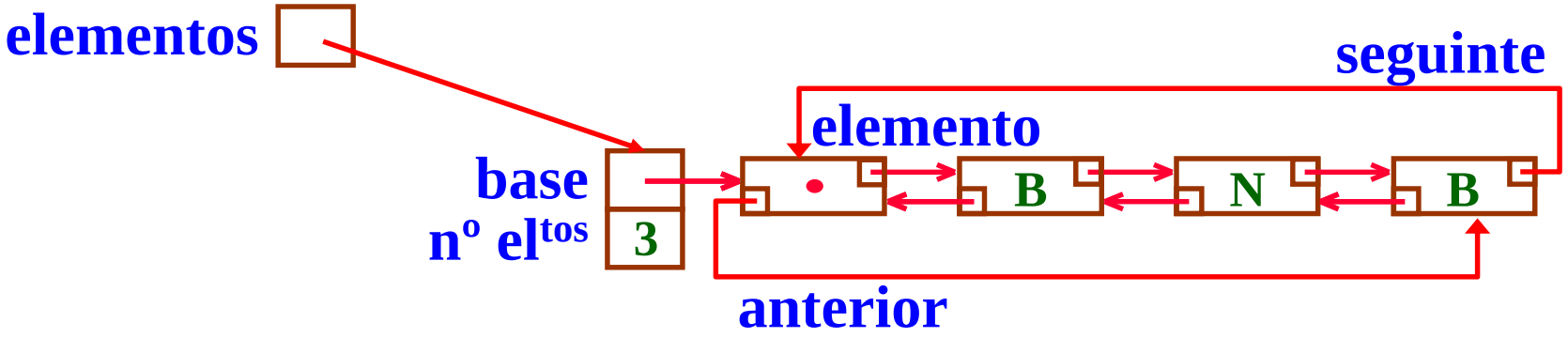
# Filas

- Classes a construir de seguida



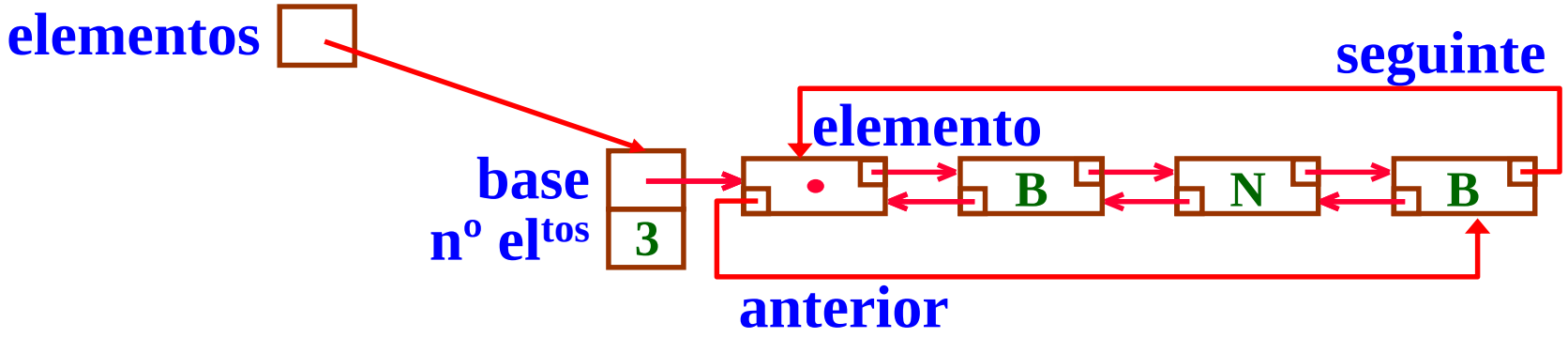
# Filas

- Uma fila de referências para objetos pode ser definida, com recurso a uma lista dupla circular com base não ordenada, do seguinte modo:
- **Estrutura**



# Filas

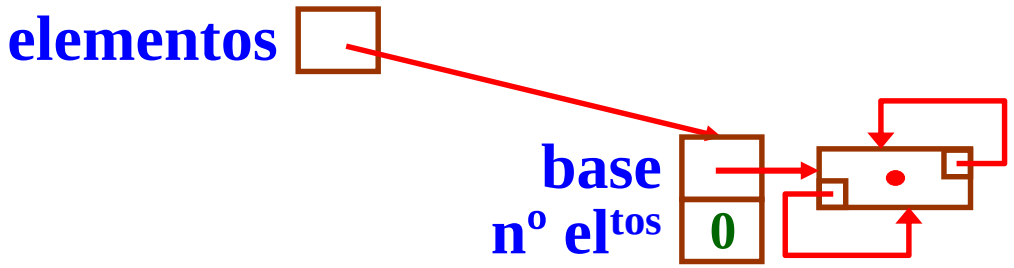
- Uma fila de referências para objetos pode ser definida, com recurso a uma lista dupla circular com base não ordenada, do seguinte modo:
- **Estrutura**




```
public class Fila<T> implements ColecaoNaoIteravel<T> {
    protected ListaDuplaCircularBaseNaoOrdenada<T> elementos;
```

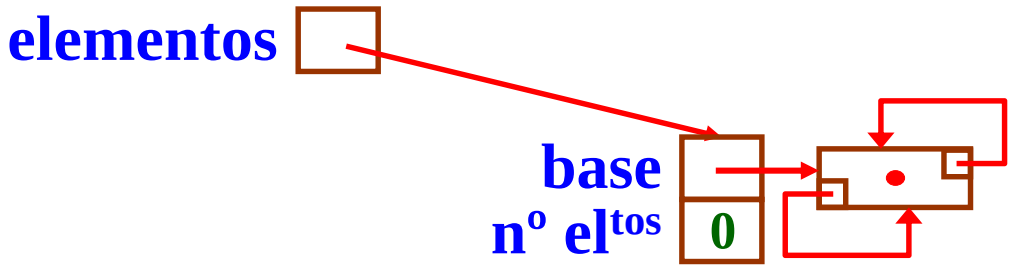
# Filas

- **Criação**



# Filas

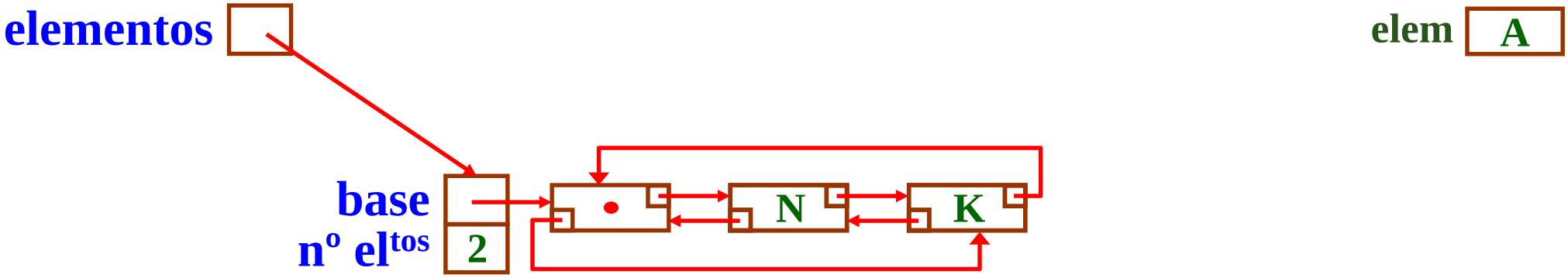
- **Criação**




```
public Fila() {
    elementos = new ListaDuplaCircularBaseNaoOrdenada<>();
}
```

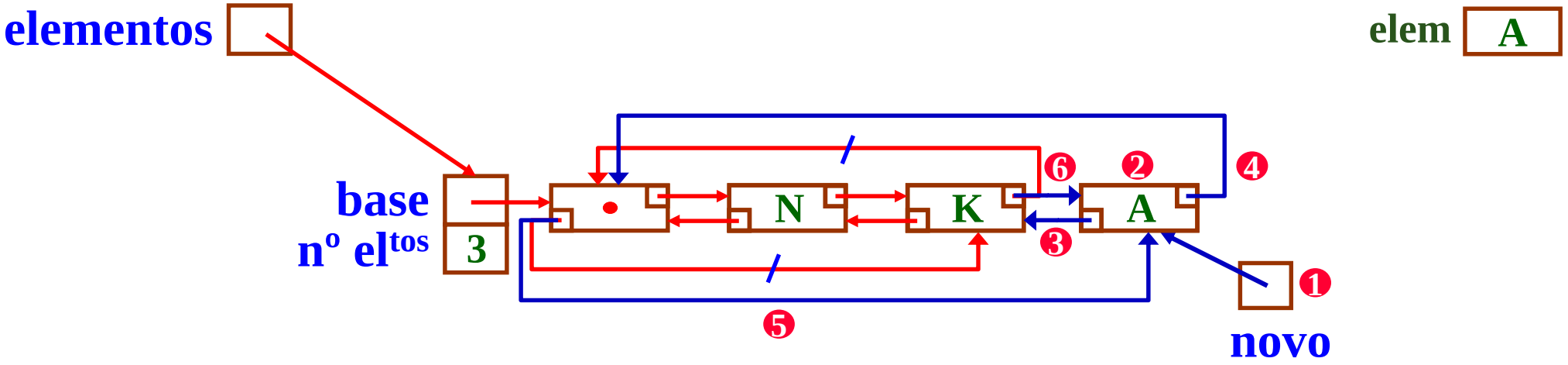
# Filas

- **Inserção –** de ***elem***



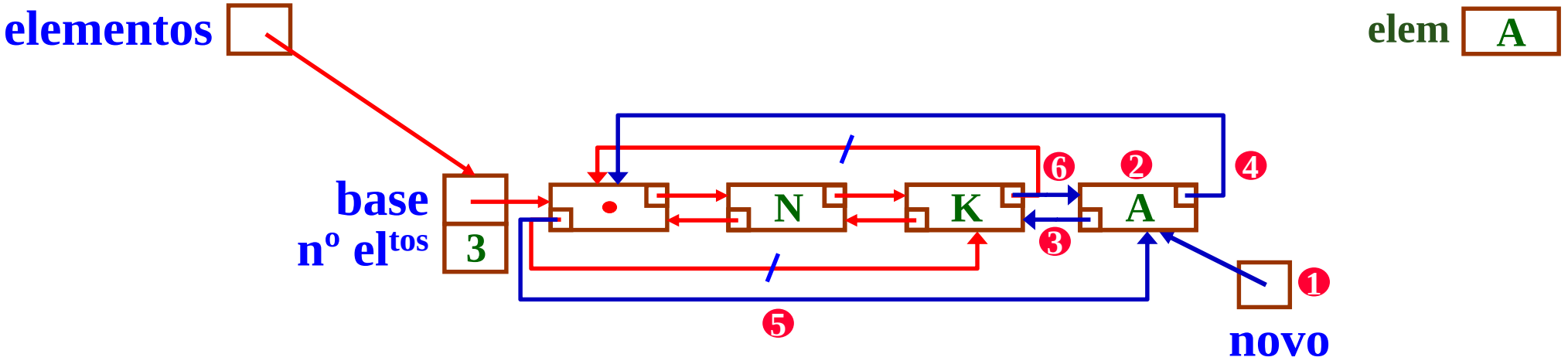
# Filas

- **Inserção –** de ***elem***



# Filas

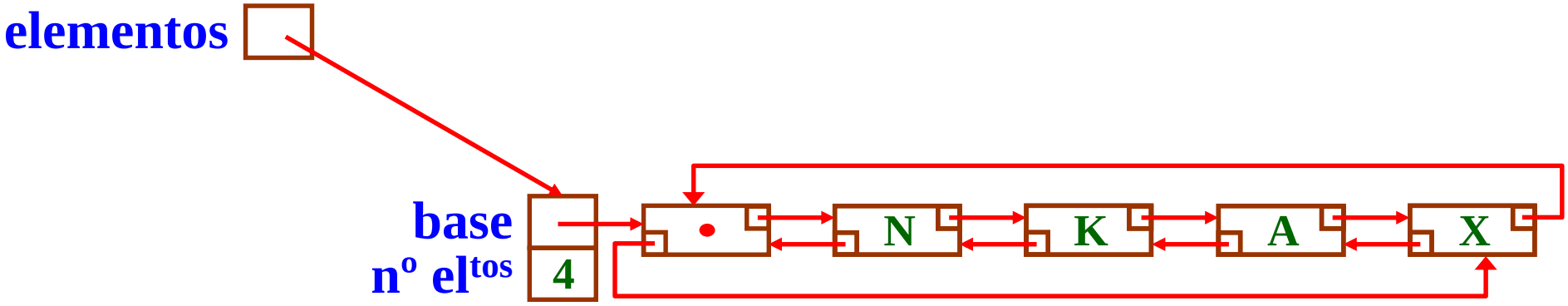
- **Inserção –** de ***elem***




```
public void inserir(T elem) {
    elementos.inserirNoFim(elem);
}
```

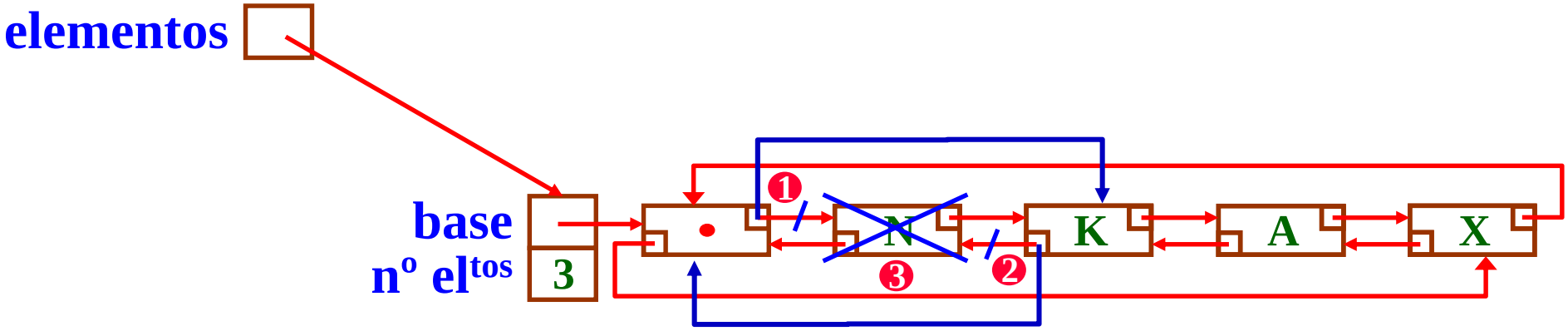
# Filas

- **Remoção**



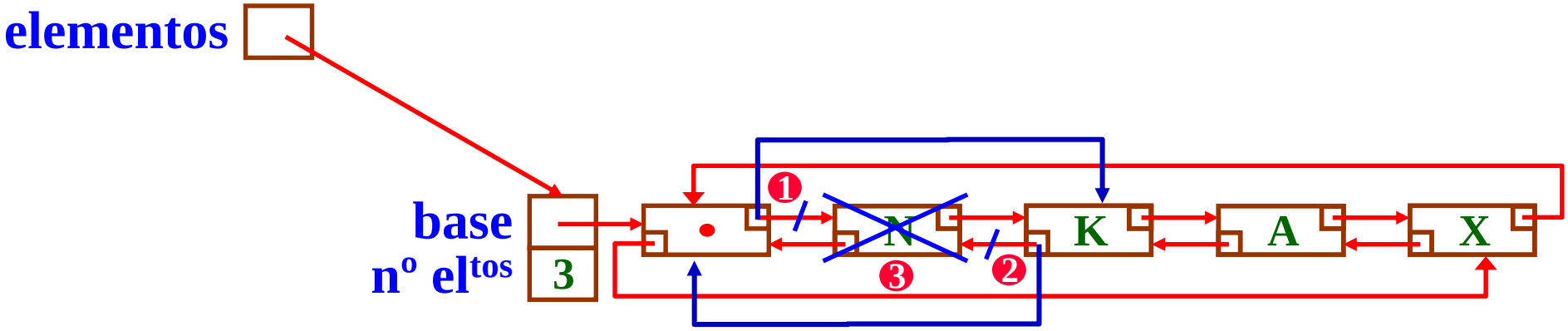
# Filas

- **Remoção**



# Filas

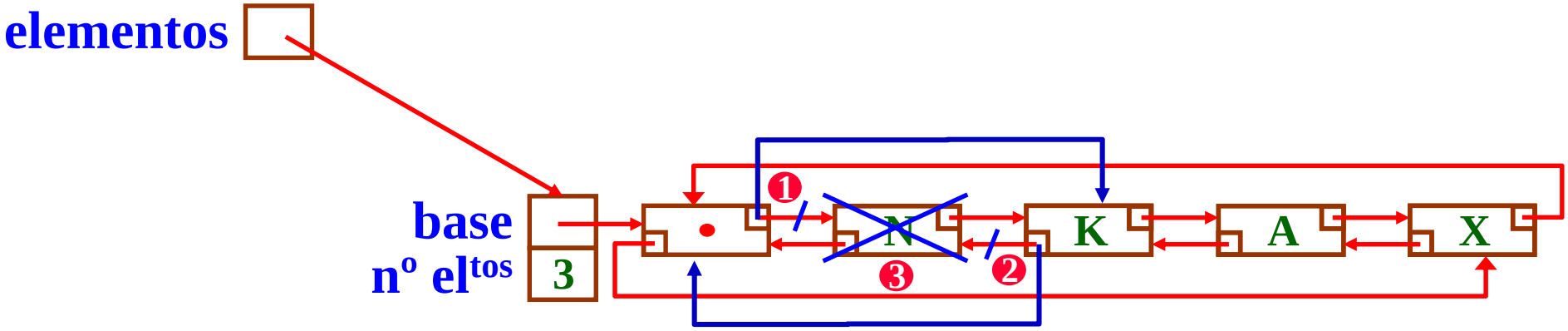
- **Remoção**




```
public T remover() {
    return elementos.removerDoInicio();
}
```

# Filas

- **Remoção**




```
public T remover() {
    return elementos.removerDoInicio();
}
```

- Se a fila não estiver vazia é devolvido o elemento da frente da fila, caso contrário, é devolvido `null`

# Filas

- **Consulta**

```
public T consultar() {
    try {
        return elementos.consultar(0);
    } catch (IndexOutOfBoundsException ex) {
        return null;
    }
}
```

  - devolve o elemento da frente da fila, ou `null` no caso da fila estar vazia

- **Vazia**

```
public boolean isVazia() {
    return elementos.isVazia();
}
```

# Filas

- **Representação textual**

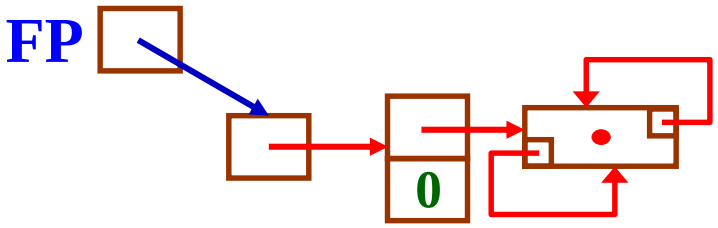
```
    public String toString() {
        StringBuilder s = new StringBuilder();
        s.append("Fila = {\n");
        for (T elemento : elementos) {
            s.append(elemento).append("\n");
        }
        s.append("}\n");
        return s.toString();
    }
} // Fim da classe Fila
```

# Filas

- Exemplo de Utilização

```
public class MainTeoricaPilha {

    public static void main(String[] args) {
        new MainTeoricaFila();
    }

    public MainTeoricaFila() {
        Fila<Pessoa> filaPessoas = new Fila<>();

        filaPessoas.inserir(new Pessoa(3, "B"));

        filaPessoas.inserir(new Pessoa(1, "C"));

        filaPessoas.inserir(new Pessoa(2, "A"));
```

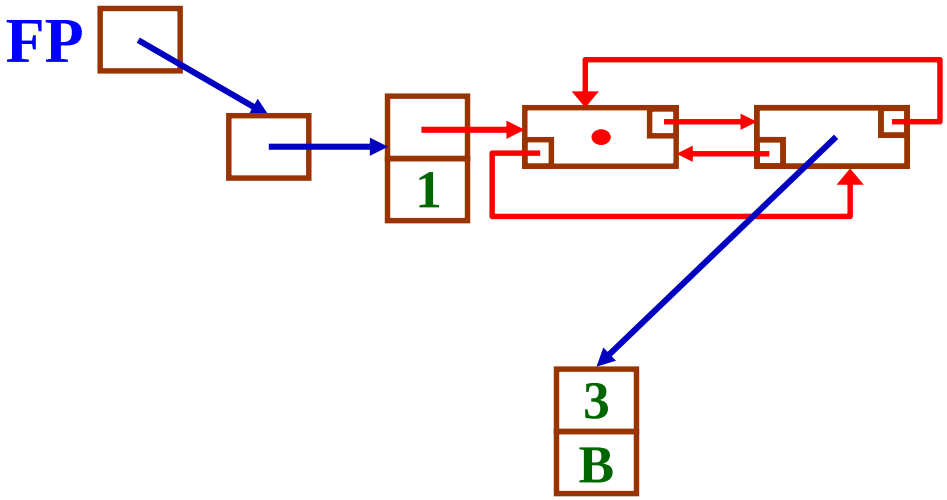
# Filas



# Filas

```
public class MainTeoricaFila {

    public static void main(String[] args) {
        new MainTeoricaFila();
    }

    public MainTeoricaFila() {
        Fila<Pessoa> filaPessoas = new Fila<>();

        filaPessoas.inserir(new Pessoa(3, "B"));

        filaPessoas.inserir(new Pessoa(1, "C"));

        filaPessoas.inserir(new Pessoa(2, "A"));
```

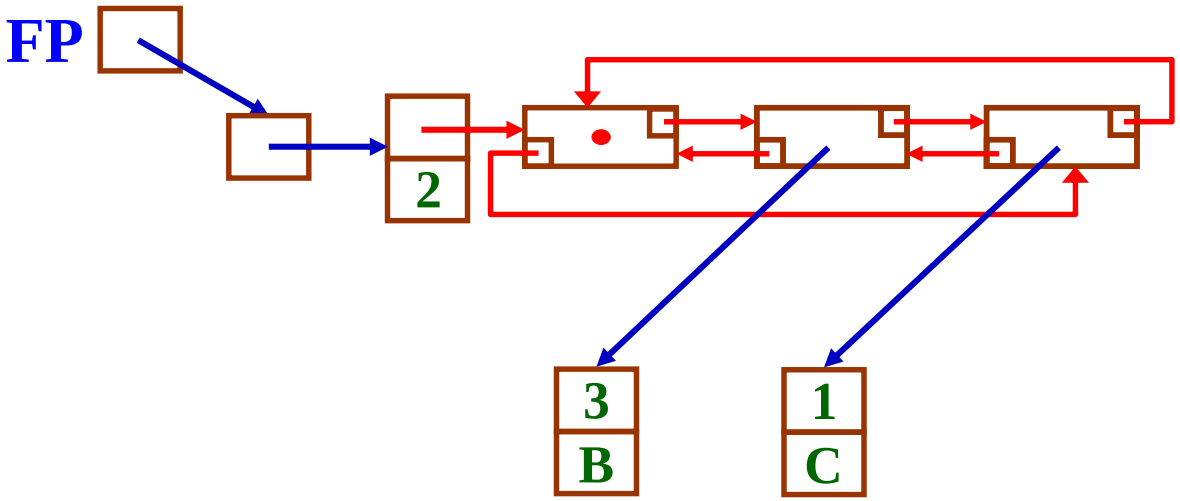
# Filas



# Filas

```
public class MainTeoricaFila {

    public static void main(String[] args) {
        new MainTeoricaFila();
    }

    public MainTeoricaFila() {
        Fila<Pessoa> filaPessoas = new Fila<>();

        filaPessoas.inserir(new Pessoa(3, "B"));

        filaPessoas.inserir(new Pessoa(1, "C"));

        filaPessoas.inserir(new Pessoa(2, "A"));
```

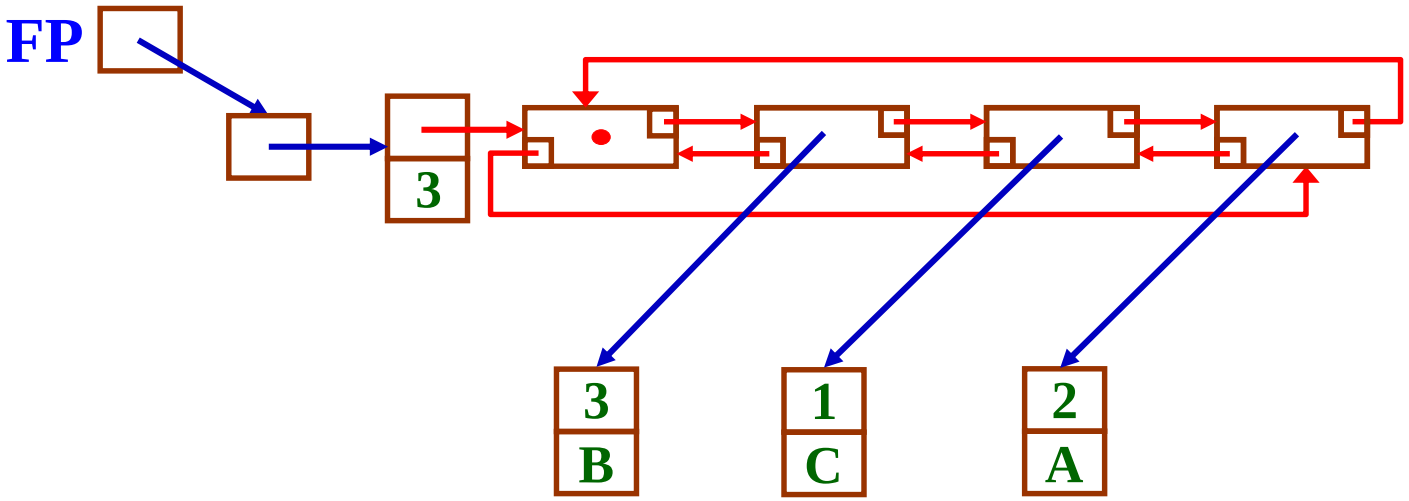
# Filas



# Filas

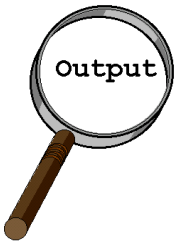
```
public class MainTeoricaFila {

    public static void main(String[] args) {
        new MainTeoricaFila();
    }

    public MainTeoricaFila() {
        Fila<Pessoa> filaPessoas = new Fila<>();

        filaPessoas.inserir(new Pessoa(3, "B"));

        filaPessoas.inserir(new Pessoa(1, "C"));

        filaPessoas.inserir(new Pessoa(2, "A"));
```

# Filas



# Filas

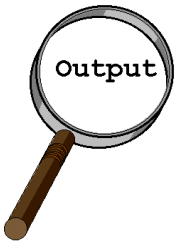
```
        System.out.println("filaPessoas\n" + filaPessoas);

        filaPessoas.remover();

        System.out.println("filaPessoas\n" + filaPessoas);

        System.out.println(filaPessoas.consultar());

        filaPessoas.remover();

        System.out.println("filaPessoas\n" + filaPessoas);
    }
}
```




```
filaPessoas
Fila = {
BI: 3 Nome: B
BI: 1 Nome: C
BI: 2 Nome: A
}
```

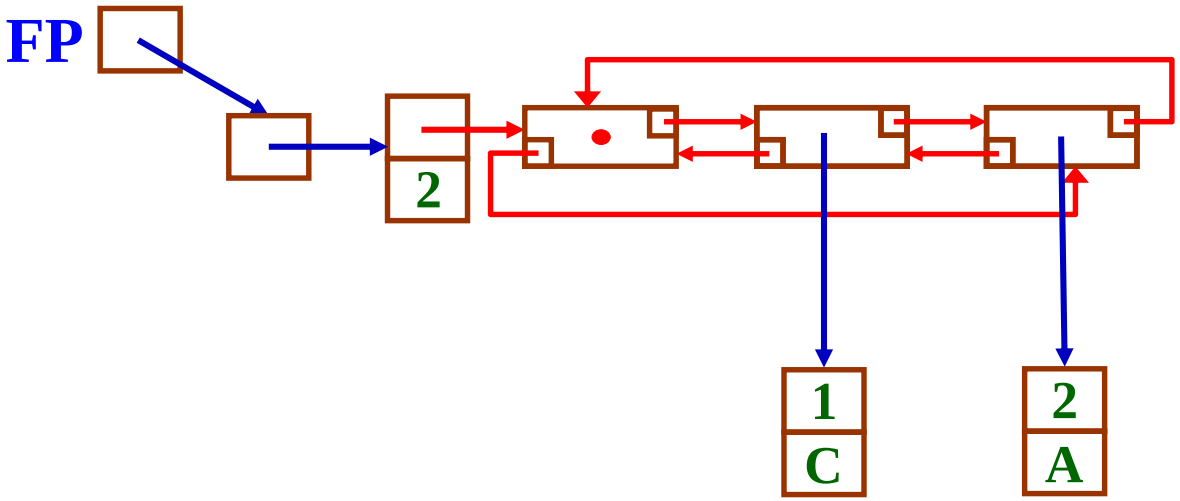
# Filas

```
        System.out.println("filaPessoas\n" + filaPessoas);

        filaPessoas.remover();

        System.out.println("filaPessoas\n" + filaPessoas);

        System.out.println(filaPessoas.consultar());

        filaPessoas.remover();

        System.out.println("filaPessoas\n" + filaPessoas);
    }
}
```



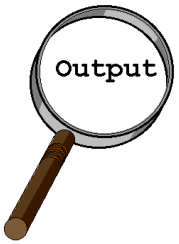

```
filaPessoas
Fila = {
BI: 3 Nome: B
BI: 1 Nome: C
BI: 2 Nome: A
}
```

# Filas



# Filas

```
        System.out.println("filaPessoas\n" + filaPessoas);

        filaPessoas.remover();

        System.out.println("filaPessoas\n" + filaPessoas);

        System.out.println(filaPessoas.consultar());

        filaPessoas.remover();

        System.out.println("filaPessoas\n" + filaPessoas);
    }
}
```

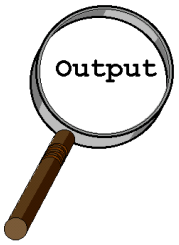



```
filaPessoas
Fila = {
BI: 3 Nome: B
BI: 1 Nome: C
BI: 2 Nome: A
}
```

```
filaPessoas
Fila = {
BI: 1 Nome: C
BI: 2 Nome: A
}
```

# Filas

```
        System.out.println("filaPessoas\n" + filaPessoas);

        filaPessoas.remover();

        System.out.println("filaPessoas\n" + filaPessoas);

        System.out.println(filaPessoas.consultar());

        filaPessoas.remover();

        System.out.println("filaPessoas\n" + filaPessoas);
    }
}
```

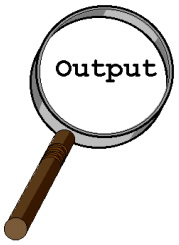



```
filaPessoas
Fila = {
BI: 3 Nome: B
BI: 1 Nome: C
BI: 2 Nome: A
}
```

```
filaPessoas
Fila = {
BI: 1 Nome: C
BI: 2 Nome: A
}

BI: 1 Nome: C
```

# Filas

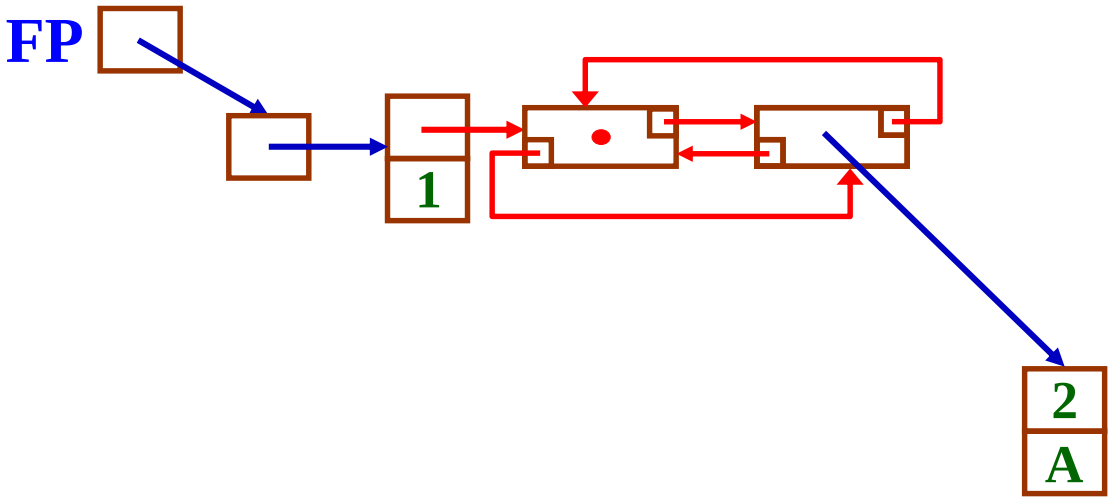
```
        System.out.println("filaPessoas\n" + filaPessoas);

        filaPessoas.remover();

        System.out.println("filaPessoas\n" + filaPessoas);

        System.out.println(filaPessoas.consultar());

        filaPessoas.remover();

        System.out.println("filaPessoas\n" + filaPessoas);
    }
}
```




```
filaPessoas
Fila = {
BI: 3 Nome: B
BI: 1 Nome: C
BI: 2 Nome: A
}
```

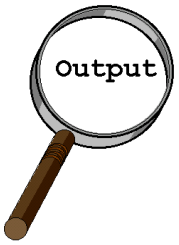
```
filaPessoas
Fila = {
BI: 1 Nome: C
BI: 2 Nome: A
}

BI: 1 Nome: C
```

# Filas



# Filas

```
        System.out.println("filaPessoas\n" + filaPessoas);

        filaPessoas.remover();

        System.out.println("filaPessoas\n" + filaPessoas);

        System.out.println(filaPessoas.consultar());

        filaPessoas.remover();

        System.out.println("filaPessoas\n" + filaPessoas);
    }
}
```




```
filaPessoas
Fila = {
BI: 3 Nome: B
BI: 1 Nome: C
BI: 2 Nome: A
}
```

```
filaPessoas
Fila = {
BI: 1 Nome: C
BI: 2 Nome: A
}

BI: 1 Nome: C
```

```
filaPessoas
Fila = {
BI: 2 Nome: A
}
```