Aula anterior

- TadFlexível Fila

Tipo Abstrato de Dados (Pilha)

Prof. Diego Silva Caldeira Rocha

Objetivos

- Introdução a Pilha
- Tipos Abstratos de Dados Lineares: Pilha
- Tipos Abstratos de Dados Flexível: Pilha

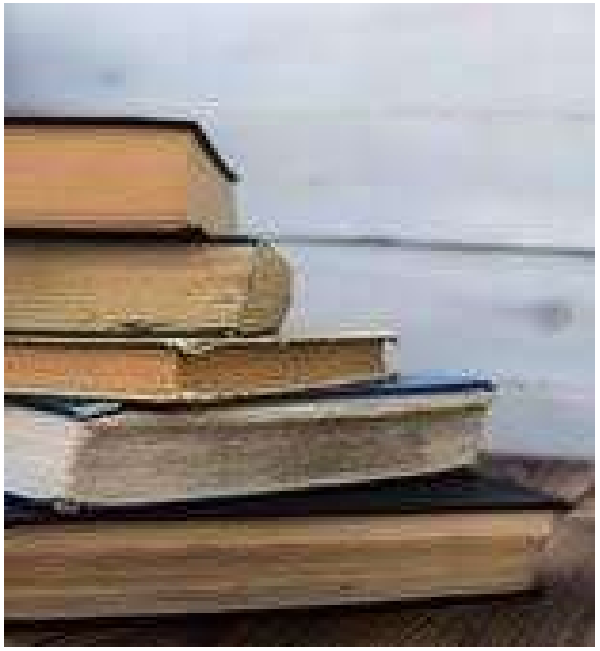
Introdução

- As pilhas são um Tipo Abstrato de Dados (TAD) no qual o **primeiro elemento que entra é o último a sair**

  - ***F**irst **I**n, **L**ast **O**ut* (FILO)
  - ***L**ast **I**n, **F**irst **O**ut* (LIFO)

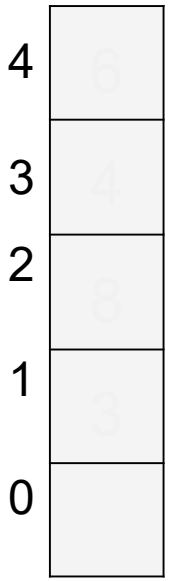
- Tem basicamente os métodos de inserir (empilhar, *push*) e remover (desempilhar, *pop*) além dos algotimos em estrutura linear : IF(inserir no fim), RF(Remover no fim) , II(Inserir no início), RI (Remover no Inicio).

Exemplos



Funcionamento

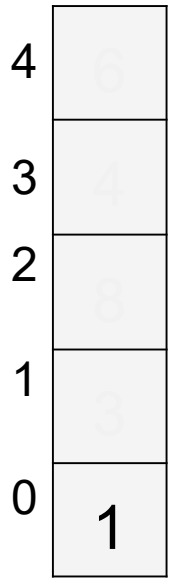
- Primeira solução IF e RF
  - Por exemplo, inserindo o 1, 3, 5 e 7 e efetuando duas remoções teremos:



- Segunda solução II e RI (inserção e remoção não eficientes)

Funcionamento

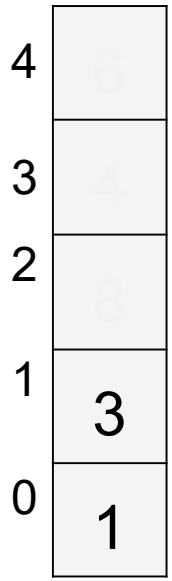
- Primeira solução IF e RF
  - Por exemplo, inserindo o **1**, 3, 5 e 7 e efetuando duas remoções teremos:



- Segunda solução II e RI (inserção e remoção não eficientes)

Funcionamento

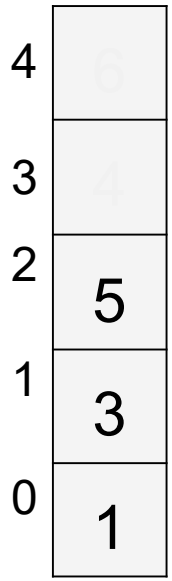
- Primeira solução IF e RF
  - Por exemplo, inserindo o 1, **3**, 5 e 7 e efetuando duas remoções teremos:



- Segunda solução II e RI (inserção e remoção não eficientes)

Funcionamento

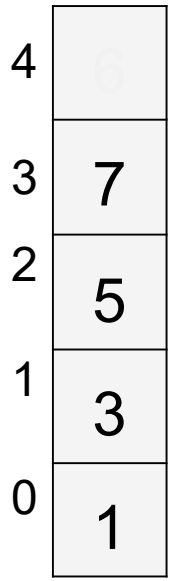
- Primeira solução IF e RF
  - Por exemplo, inserindo o 1, 3, **5** e 7 e efetuando duas remoções teremos:



- Segunda solução II e RI (inserção e remoção não eficientes)

Funcionamento

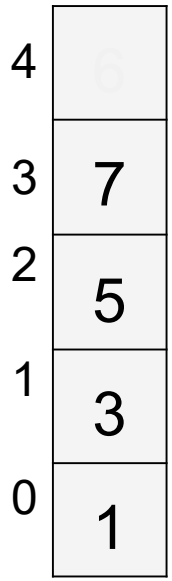
- Primeira solução IF e RF
  - Por exemplo, inserindo o 1, 3, 5 e **7** e efetuando duas remoções teremos:



- Segunda solução II e RI (inserção e remoção não eficientes)

Funcionamento

- Primeira solução IF e RF
  - Por exemplo, inserindo o 1, 3, 5 e 7 e efetuando duas remoções teremos:



- Segunda solução II e RI (inserção e remoção não eficientes)

Funcionamento

- Primeira solução IF e RF
  - Por exemplo, inserindo o 1, 3, 5 e 7 e efetuando duas remoções teremos:

Na primeira remoção, retiramos o número 7

- Segunda solução II e RI (inserção e remoção não eficientes)

Funcionamento

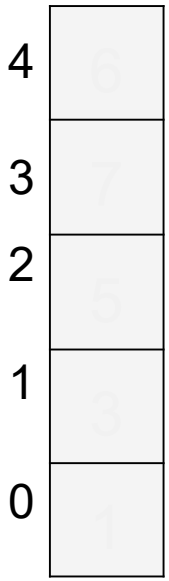
- Primeira solução IF e RF
  - Por exemplo, inserindo o 1, 3, 5 e 7 e efetuando duas remoções teremos:

Na segunda remoção, retiramos o número 5

- Segunda solução II e RI (inserção e remoção não eficientes)

Funcionamento

- Primeira solução IF e RF



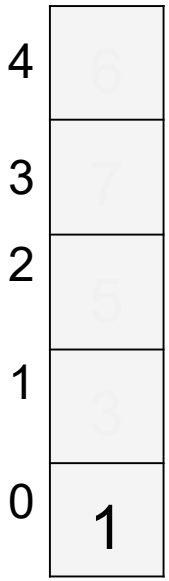
- Segunda solução II e RI (inserção e remoção não eficientes)
  - Por exemplo, inserindo o 1, 3, 5 e 7 e

efetuando duas remoções teremos:

Em cada inserção ou remoção, movemos todos os elementos

Funcionamento

- Primeira solução IF e RF



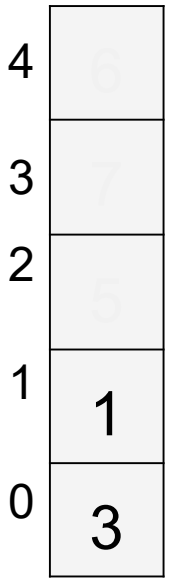
- Segunda solução II e RI (inserção e remoção não eficientes)
  - Por exemplo, inserindo o **1**, 3, 5 e 7 e

efetuando duas remoções teremos:

Em cada inserção ou remoção, movemos todos os elementos

Funcionamento

- Primeira solução IF e RF



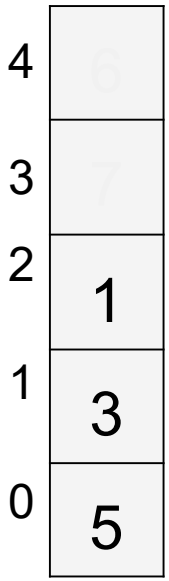
- Segunda solução II e RI (inserção e remoção não eficientes)
  - Por exemplo, inserindo o 1, **3**, 5 e 7 e

efetuando duas remoções teremos:

Em cada inserção ou remoção, movemos todos os elementos

Funcionamento

- Primeira solução IF e RF



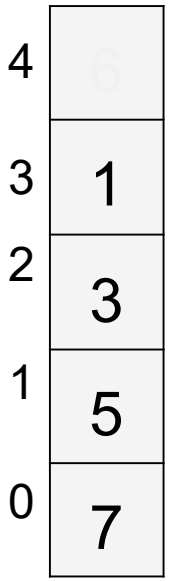
- Segunda solução II e RI (inserção e remoção não eficientes)
  - Por exemplo, inserindo o 1, 3, **5** e 7 e

efetuando duas remoções teremos:

Em cada inserção ou remoção, movemos todos os elementos

Funcionamento

- Primeira solução IF e RF



- Segunda solução II e RI (inserção e remoção não eficientes)
  - Por exemplo, inserindo o 1, 3, 5 e **7** e

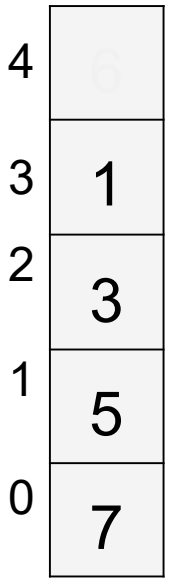
efetuando duas remoções teremos:

Em cada inserção ou remoção, movemos todos os elementos

Funcionamento

- Primeira solução IF e RF

Primeira remoção: Retorna o 7 e move todos os demais



- Segunda solução II e RI (inserção e remoção não eficientes)
  - Por exemplo, inserindo o 1, 3, 5 e 7 e

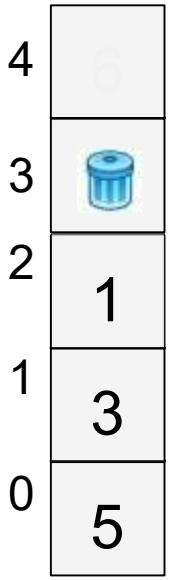
efetuando duas remoções teremos:

Em cada inserção ou remoção, movemos todos os elementos

Funcionamento

- Primeira solução IF e RF

Primeira remoção: Retorna o 7 e move todos os demais



- Segunda solução II e RI (inserção e remoção não eficientes)
  - Por exemplo, inserindo o 1, 3, 5 e 7 e

efetuando duas remoções teremos:

Em cada inserção ou remoção, movemos todos os elementos

Funcionamento

- Primeira solução IF e RF

Primeira remoção: Retorna o 7 e move todos os demais

Segunda remoção: Retorna o 5 e move todos os demais

- Segunda solução II e RI (inserção e remoção não eficientes)
  - Por exemplo, inserindo o 1, 3, 5 e 7 e

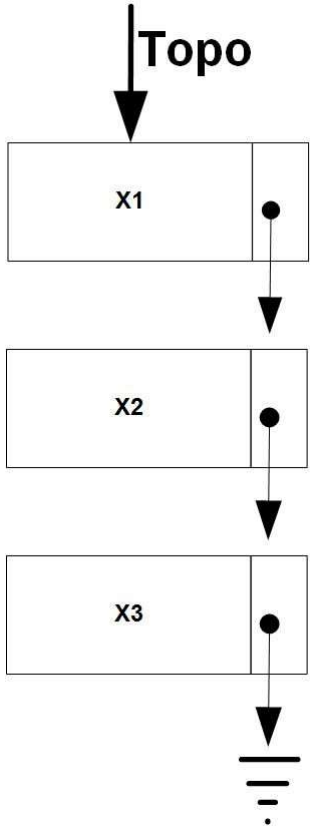
efetuando duas remoções teremos:

Em cada inserção ou remoção, movemos todos os elementos

Código Fonte

- igual ao da estrutura sequencial

- Pilha.java, criará instâncias como:

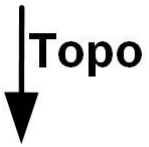


Classe Pilha

```
class Pilha {
    private Celula topo;
    public Pilha () {
        topo = null;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```

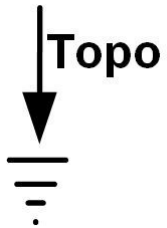
Classe Pilha

```
class Pilha {
    private Celula topo;
    public Pilha () {
        topo = null;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```



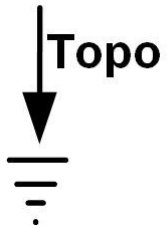
Classe Pilha

```
class Pilha {
    private Celula topo;
    public Pilha () {
        topo = null;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```



Inserir (ou Empilhar ou push)

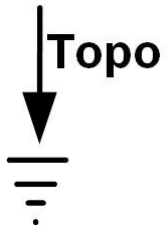
```
class Pilha {
    private Celula topo;
    public Pilha () {
        topo = null;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```




```
public void inserir(int x) {
   Celula tmp = new Celula(x);
   tmp.prox = topo;
   topo = tmp;
   tmp = null;
}
```

Inserir (ou Empilhar ou push)

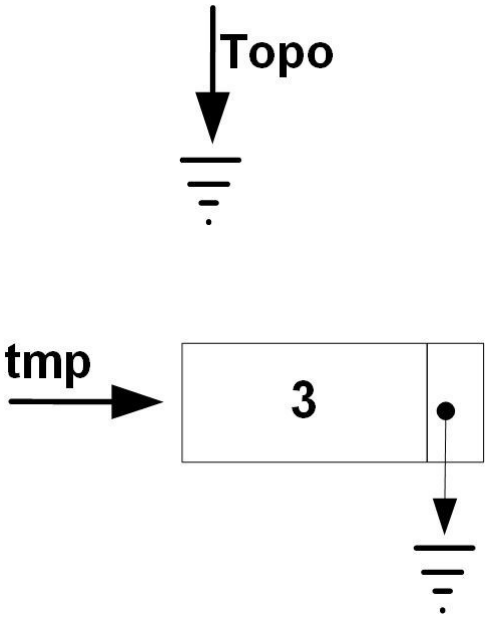
```
class Pilha {
    private Celula topo;
    public Pilha () {
        topo = null;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```




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

Inserir (ou Empilhar ou push)

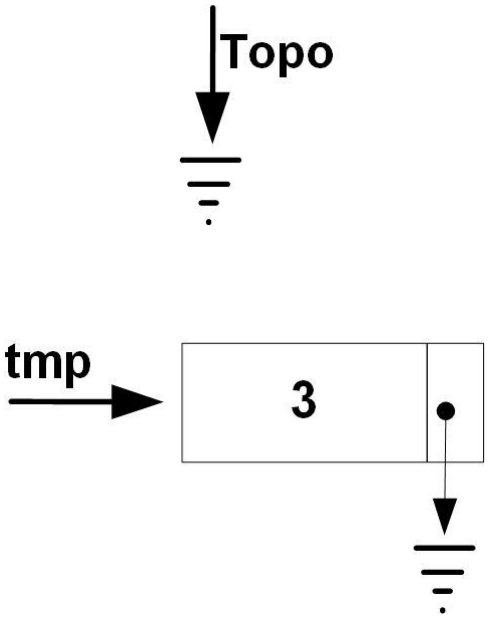
```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```




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

Inserir (ou Empilhar ou push)

```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```

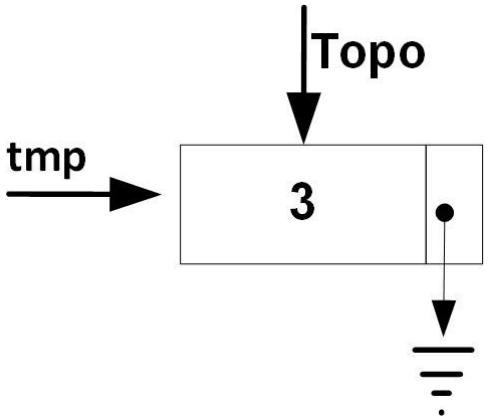



```
public void inserir(int x) {  //Inserir(3)
   Celula tmp = new Celula(x);
   tmp.prox = topo;
   topo = tmp;
   tmp = null;
}
```

Como topo aponta para null, tmp.prox continua apontando para null

Inserir (ou Empilhar ou push)

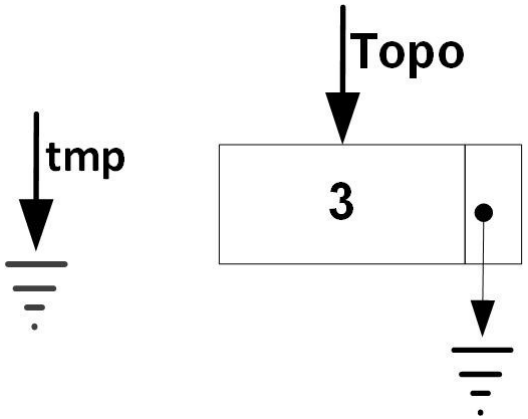
```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```




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

Inserir (ou Empilhar ou push)

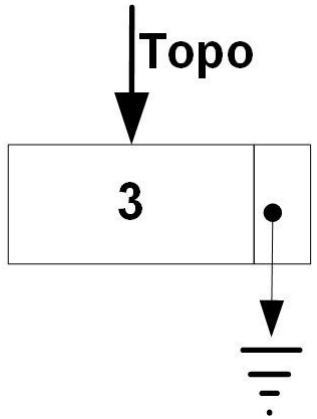
```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```




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

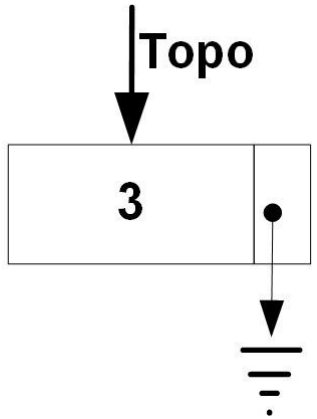
Classe Pilha

```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```



Inserir (ou Empilhar ou push)

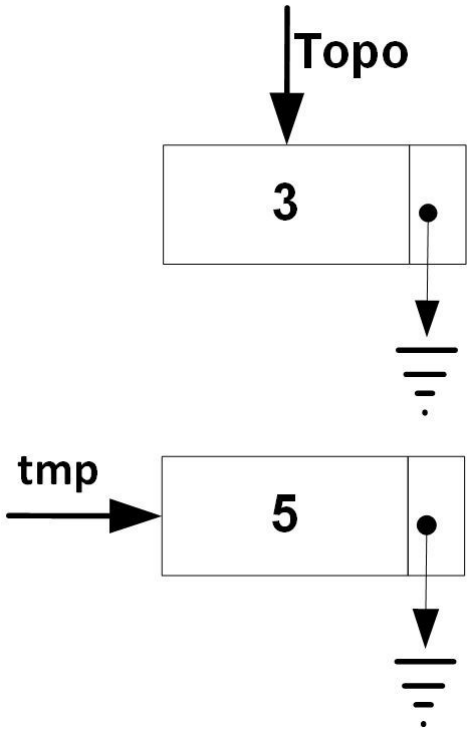
```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```




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

Inserir (ou Empilhar ou push)

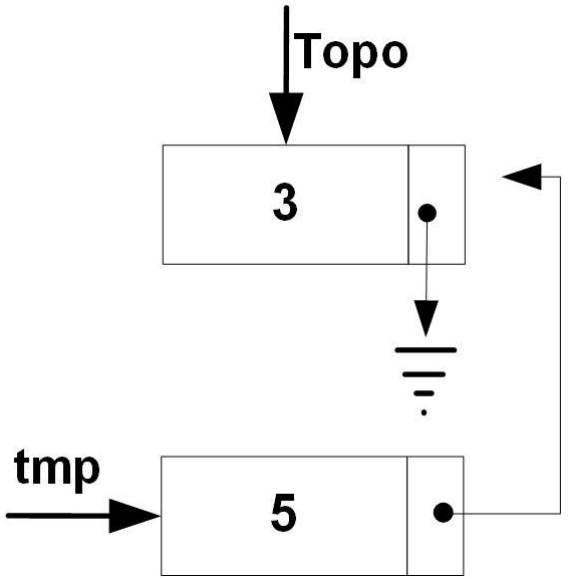
```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```




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

Inserir (ou Empilhar ou push)

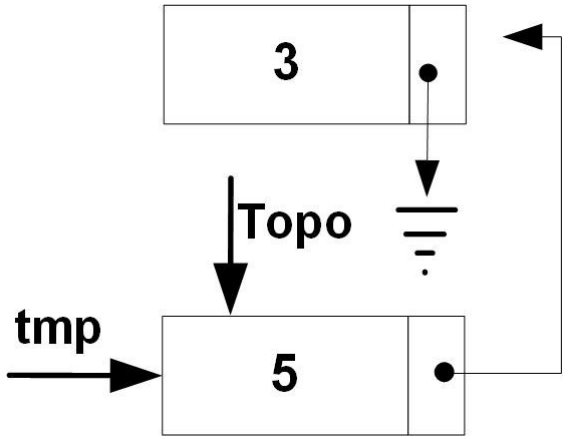
```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```




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

Inserir (ou Empilhar ou push)

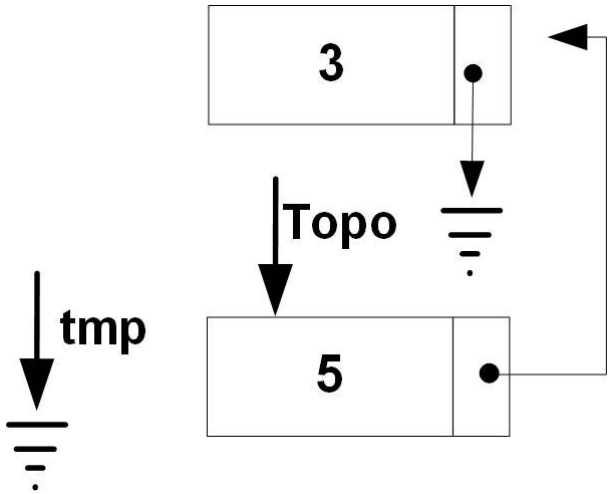
```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```




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

Inserir (ou Empilhar ou push)

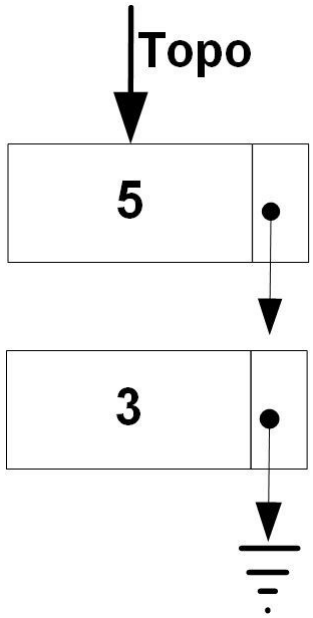
```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```




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

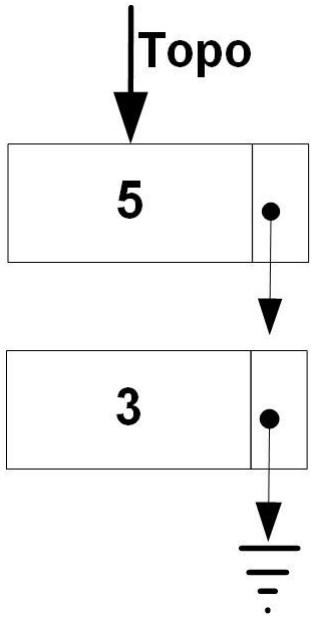
Classe Pilha

```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```



Inserir (ou Empilhar ou push)

```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```

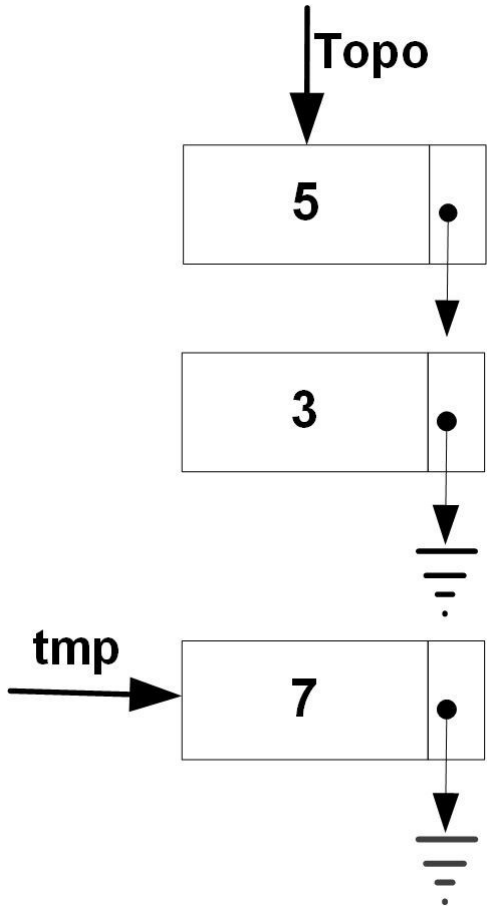



```
public void inserir(int x) {  //Inserir(7)
   Celula tmp = new Celula(x);
   tmp.prox = topo;
   topo = tmp;
   tmp = null;
}
```

Inserir (ou Empilhar ou push)

```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```

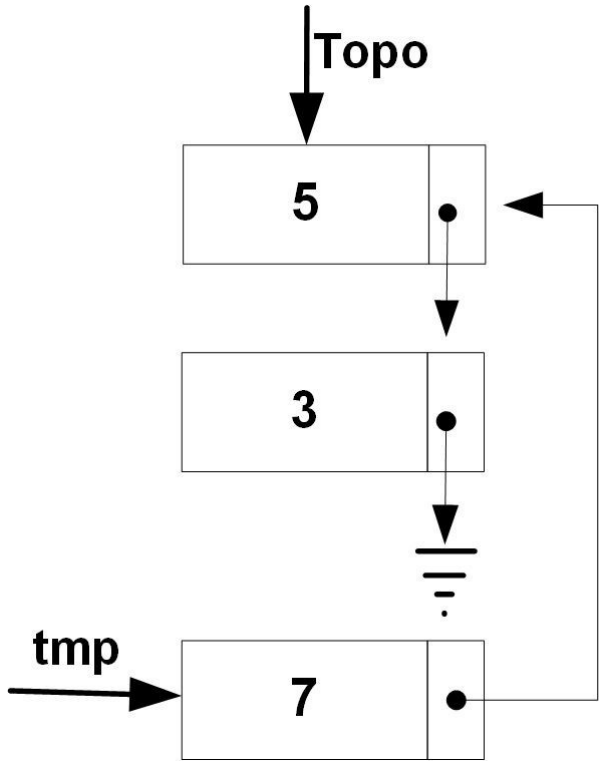
```
public void inserir(int x) {  //Inserir(7)
   Celula tmp = new Celula(x);
   tmp.prox = topo;
   topo = tmp;
   tmp = null;
}
```



Inserir (ou Empilhar ou push)

```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```

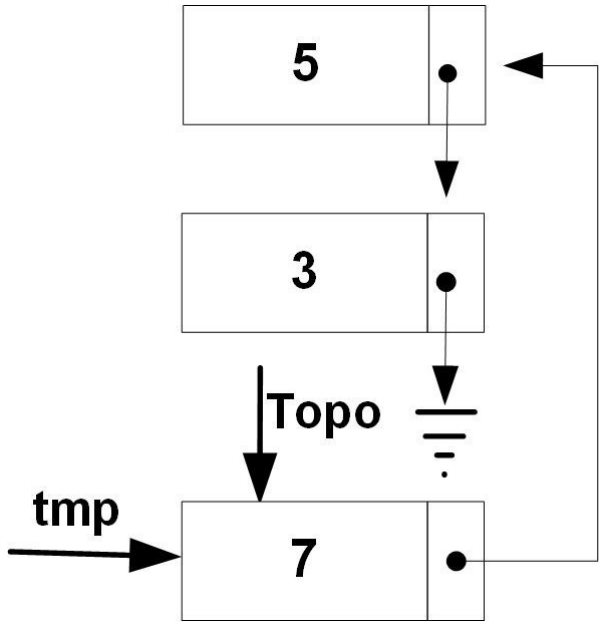
```
public void inserir(int x) {  //Inserir(7)
   Celula tmp = new Celula(x);
   tmp.prox = topo;
   topo = tmp;
   tmp = null;
}
```



Inserir (ou Empilhar ou push)

```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```

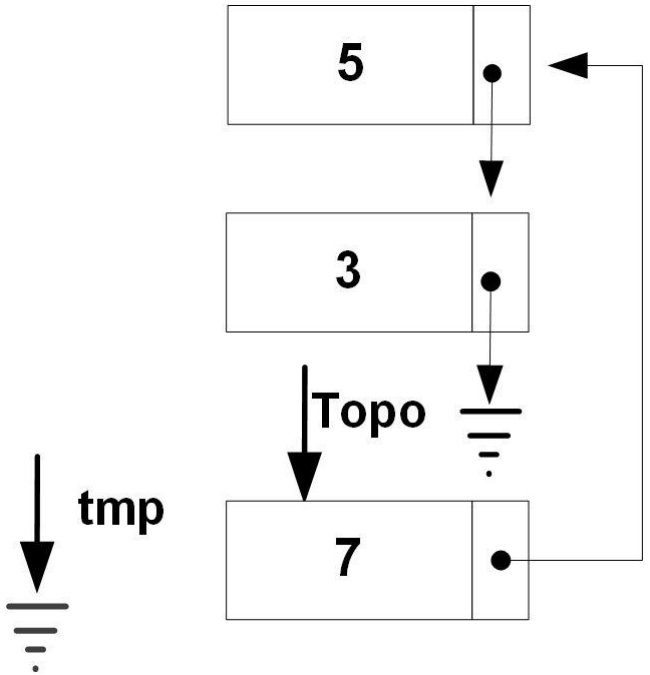
```
public void inserir(int x) {  //Inserir(7)
   Celula tmp = new Celula(x);
   tmp.prox = topo;
   topo = tmp;
   tmp = null;
}
```



Inserir (ou Empilhar ou push)

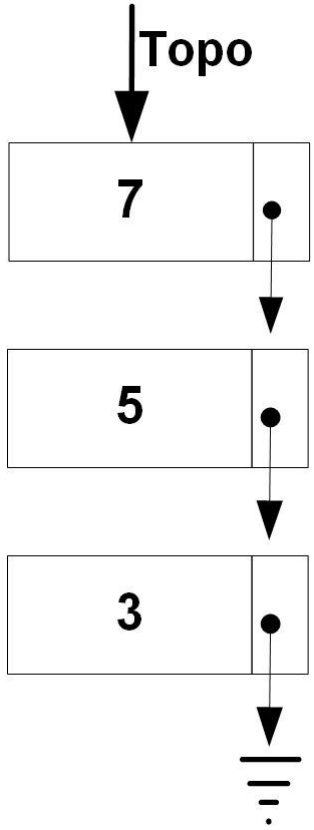
```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```

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



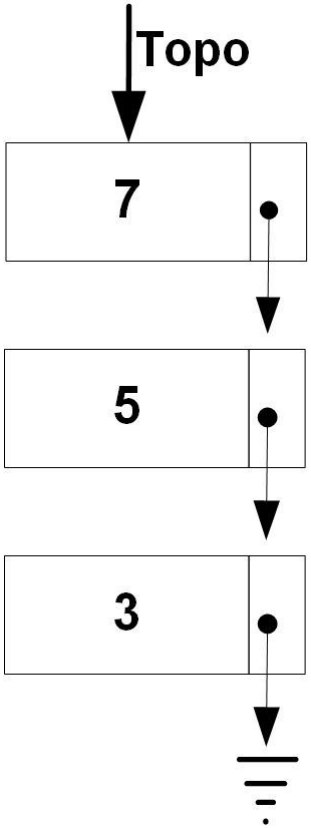
Classe Pilha

```
class Pilha {
    private Celula topo;
    public Pilha () {
        topo = null;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```



Remover (ou Desempilhar ou pop)

```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```



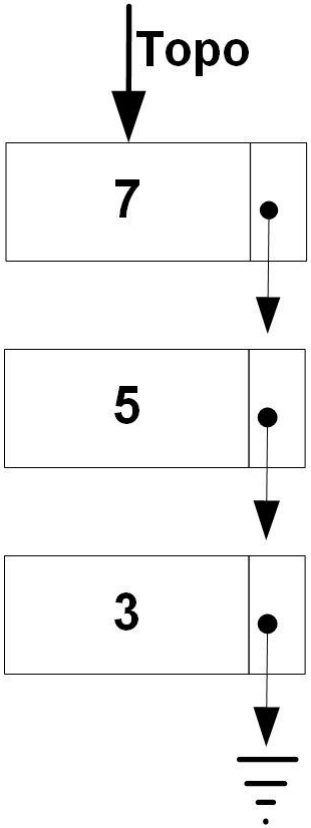

```
public int remover() throws Exception {
   if (topo == null)
      throw new Exception("Erro!”);
   int elemento = topo.elemento;
   Celula tmp = topo;
   topo = topo.prox;
   tmp.prox = null;
   tmp = null;
   return elemento;
}
```

Remover (ou Desempilhar ou pop)

```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```

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

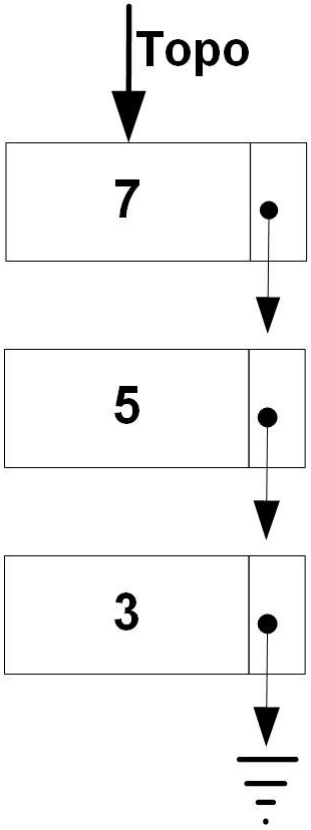
false



Remover (ou Desempilhar ou pop)

```
class Pilha {
    private Celula topo;
    public Pilha () {
        topo = null;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```

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

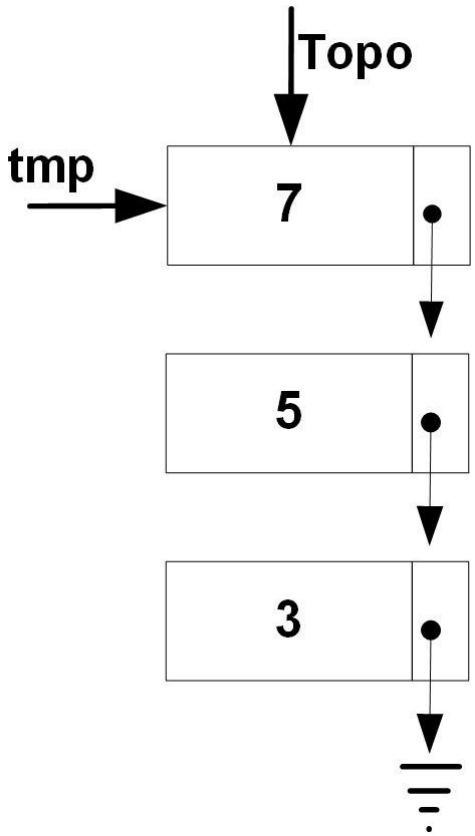


elemento | 7

Remover (ou Desempilhar ou pop)

```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```

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

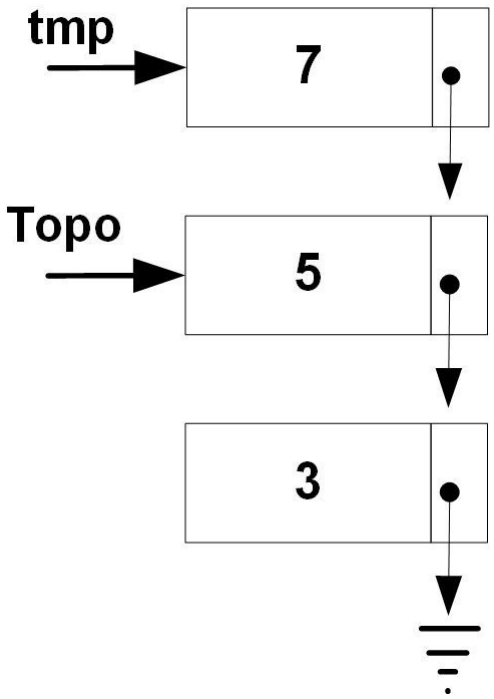


elemento 7

Remover (ou Desempilhar ou pop)

```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```

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

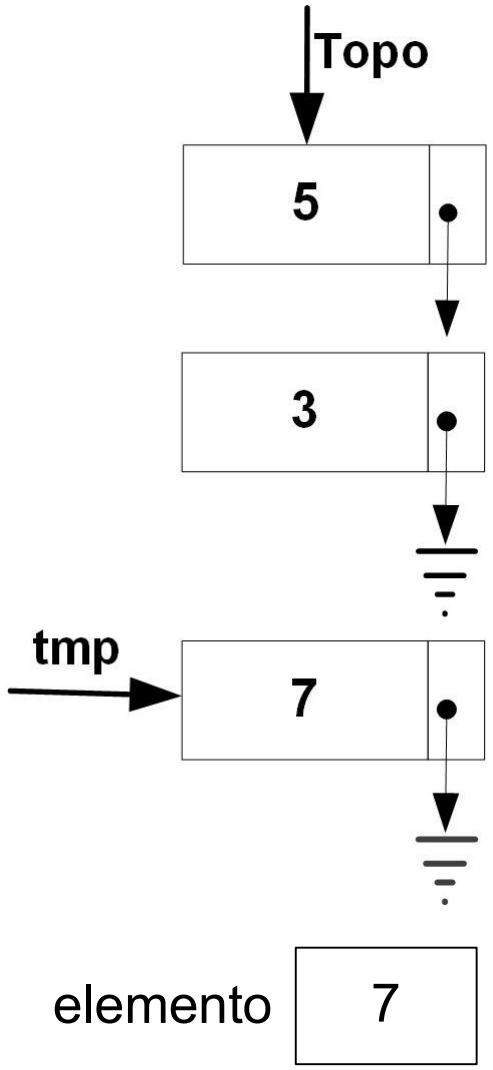


elemento 7

Remover (ou Desempilhar ou pop)

```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```

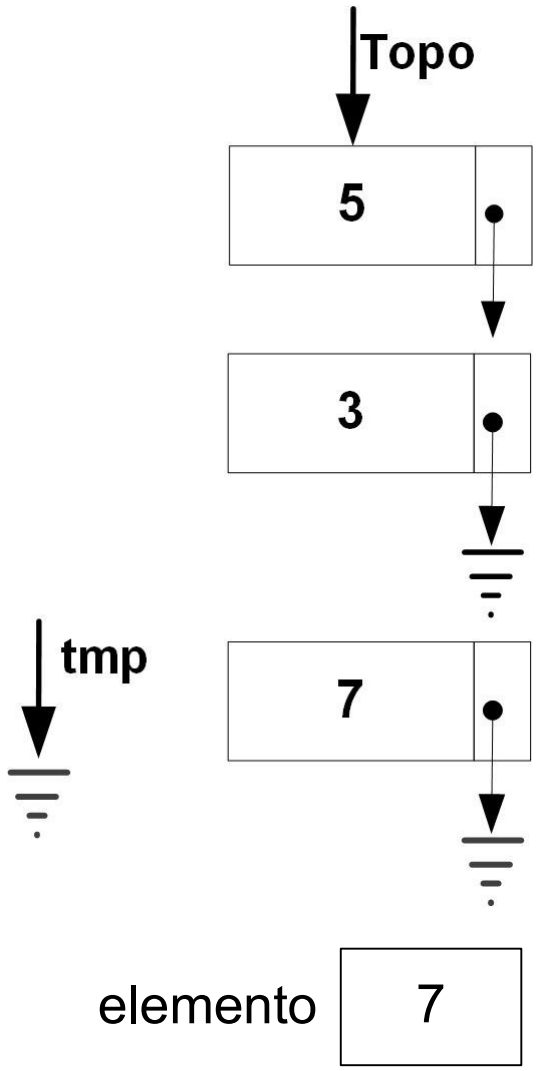
```
public int remover() throws Exception {
   if (topo == null)
      throw new Exception("Erro!”);
   int elemento = topo.elemento;
   Celula tmp = topo;
   topo = topo.prox;
   tmp.prox = null;
   tmp = null;
   return elemento;
}
```



Remover (ou Desempilhar ou pop)

```
class Pilha {
    private Celula topo;
    public Pilha () {
        topo = null;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```

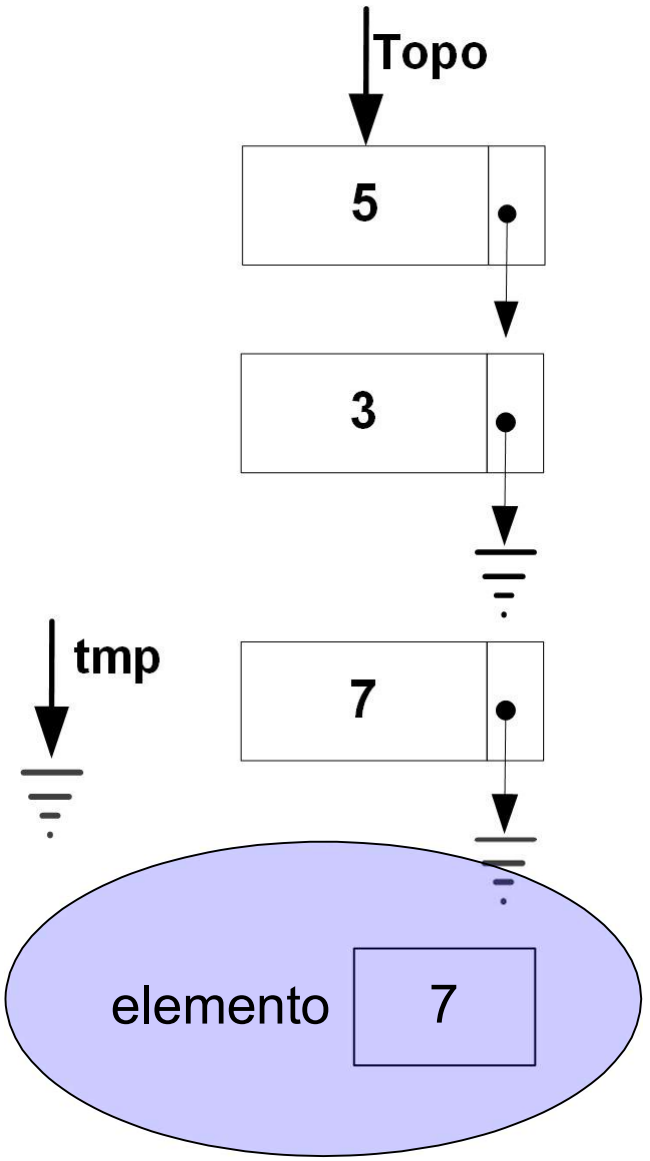
```
public int remover() throws Exception {
  if (topo == null)
    throw new Exception("Erro!");
  int elemento = topo.elemento;
  Celula tmp = topo;
  topo = topo.prox;
  tmp.prox = null;
  tmp = null;
  return elemento;
}
```



Remover (ou Desempilhar ou pop)

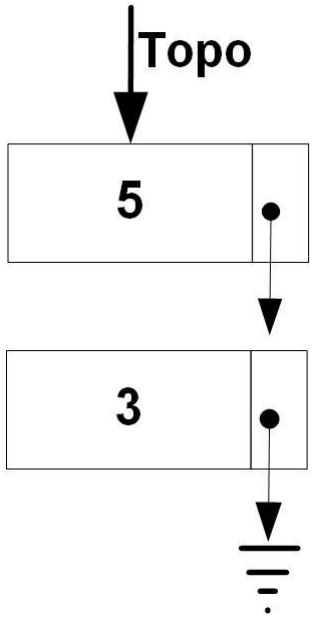
```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```

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



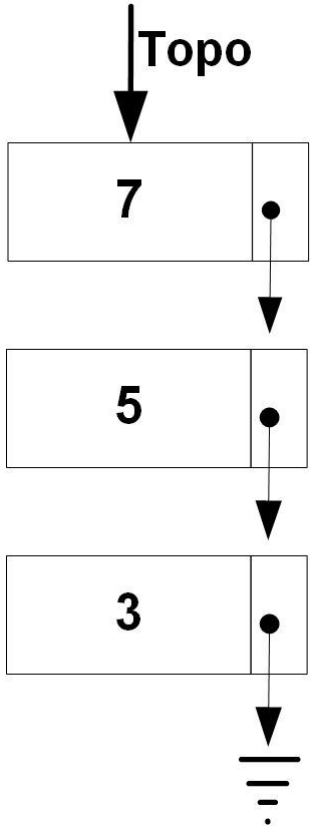
Classe Pilha

```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```



Classe Pilha

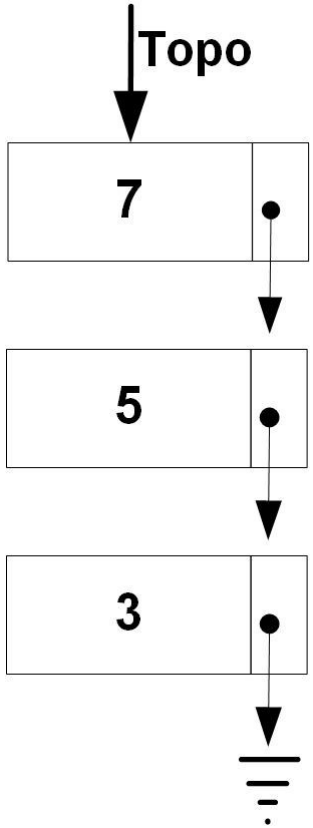
```
class Pilha {
    private Celula topo;
    public Pilha () {
        topo = null;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```



Mostrar

```
class Pilha {
    private Celula topo;
    public Pilha () {
        topo = null;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```

```
public void mostrar() {
    System.out.print("[ ");
    for (Celula i = topo; i != null; i = i.prox){
      System.out.print(i.elemento + " ");
    }
    System.out.println("]");
}
```

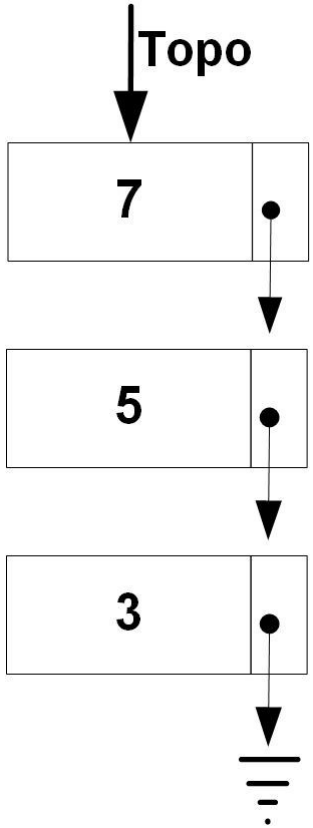


Saída na tela

Mostrar

```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```

```
public void mostrar() {
    System.out.print("[ ");
    for (Celula i = topo; i != null; i = i.prox){
       System.out.print(i.elemento + " ");
    }
    System.out.println("]");
}
```



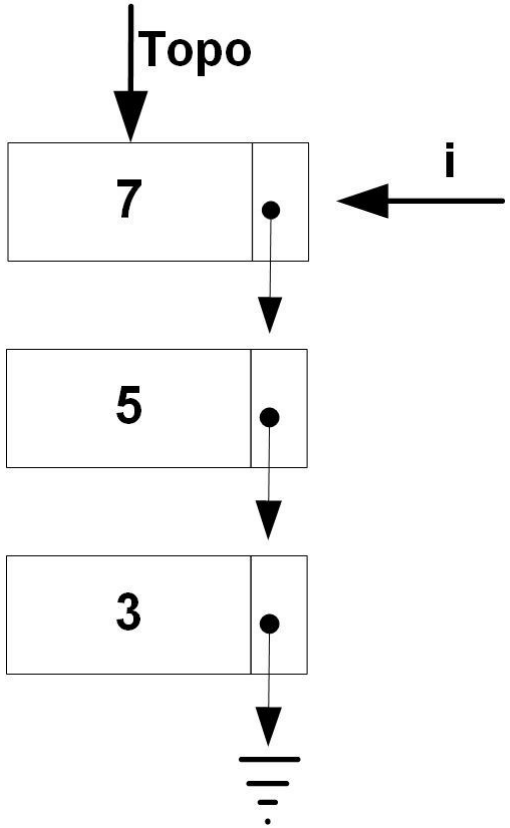
Saída na tela

[

Mostrar

```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```

```
public void mostrar() {
      System.out.print("[ ");
      for (Celula i = topo; i != null; i = i.prox){
         System.out.print(i.elemento + " ");
      }
      System.out.println("]");
}
```



Saída na tela

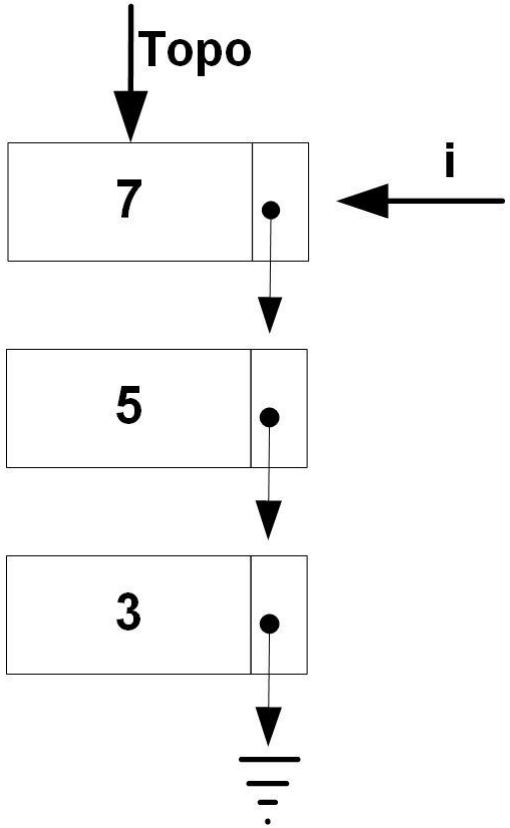
[

Mostrar

```
class Pilha {
    private Celula topo;
    public Pilha () {
        topo = null;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```

```
public void mostrar() {
    System.out.print("[ ");
    for (Celula i = topo; i != null; i = i.prox){
        System.out.print(i.elemento + " ");
    }
    System.out.println("]");
}
```

true



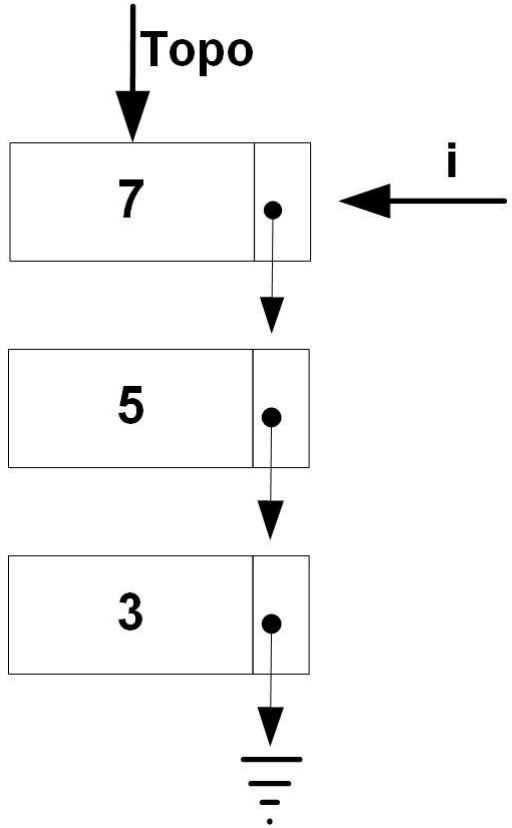
Saída na tela

[

Mostrar

```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```

```
public void mostrar() {
    System.out.print("[ ");
    for (Celula i = topo; i != null; i = i.prox){
        System.out.print(i.elemento + " ");
    }
    System.out.println("]");
}
```



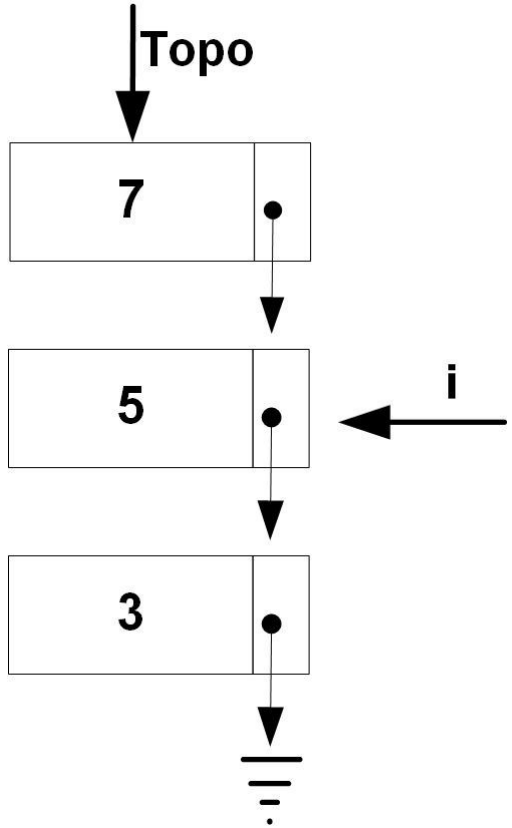
Saída na tela

```
[ 7
```

Mostrar

```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```

```
public void mostrar() {
      System.out.print("[ ");
      for (Celula i = topo; i != null; i = i.prox){
         System.out.print(i.elemento + " ");
      }
      System.out.println("]");
}
```



Saída na tela

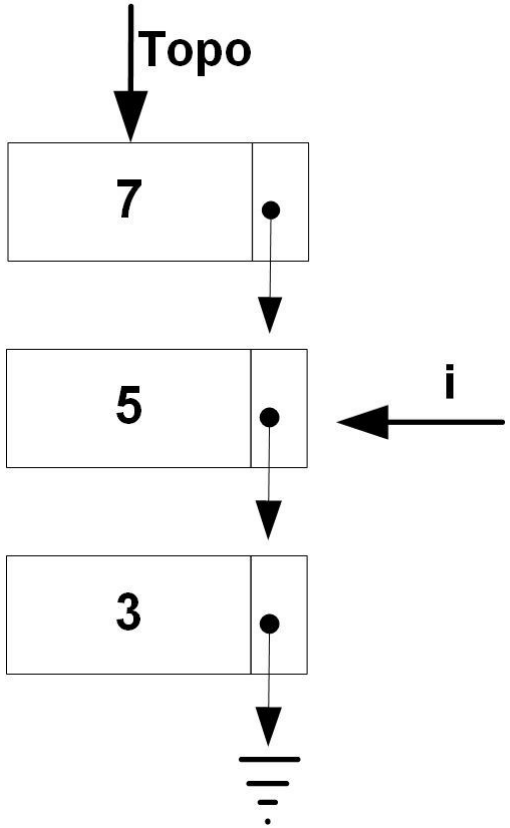
```
[ 7
```

Mostrar

```
class Pilha {
    private Celula topo;
    public Pilha () {
        topo = null;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```

```
public void mostrar() {
    System.out.print("[ ");
    for (Celula i = topo; i != null; i = i.prox){
        System.out.print(i.elemento + " ");
    }
    System.out.println("]");
}
```

true



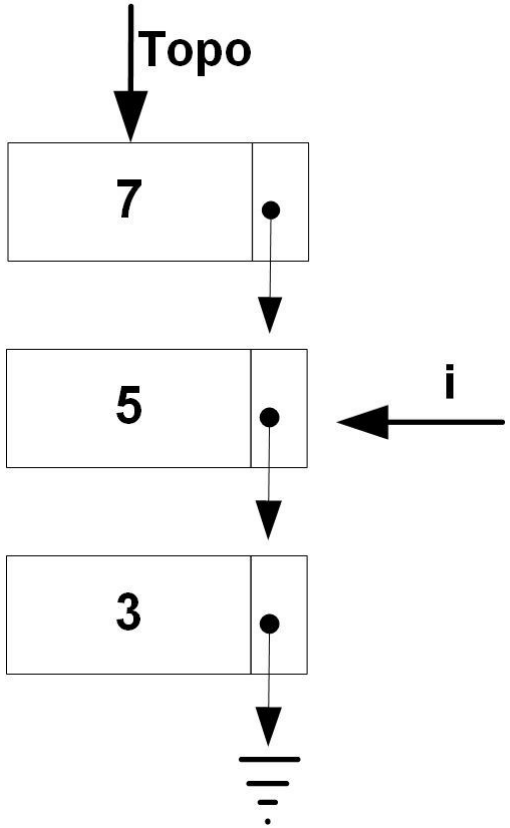
Saída na tela

```
[ 7
```

Mostrar

```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```

```
public void mostrar() {
    System.out.print("[ ");
    for (Celula i = topo; i != null; i = i.prox){
        System.out.print(i.elemento + " ");
    }
    System.out.println("]");
}
```



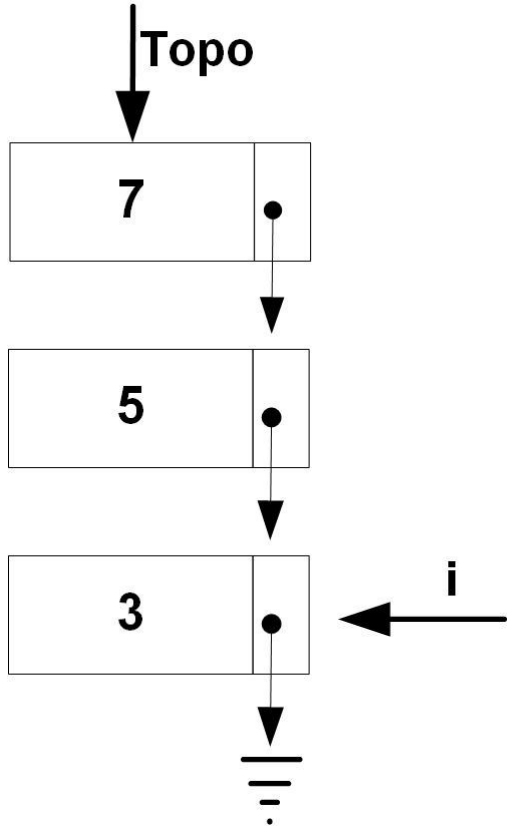
Saída na tela

```
[ 7 5
```

Mostrar

```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```

```
public void mostrar() {
      System.out.print("[ ");
      for (Celula i = topo; i != null; i = i.prox){
         System.out.print(i.elemento + " ");
      }
      System.out.println("]");
}
```



Saída na tela

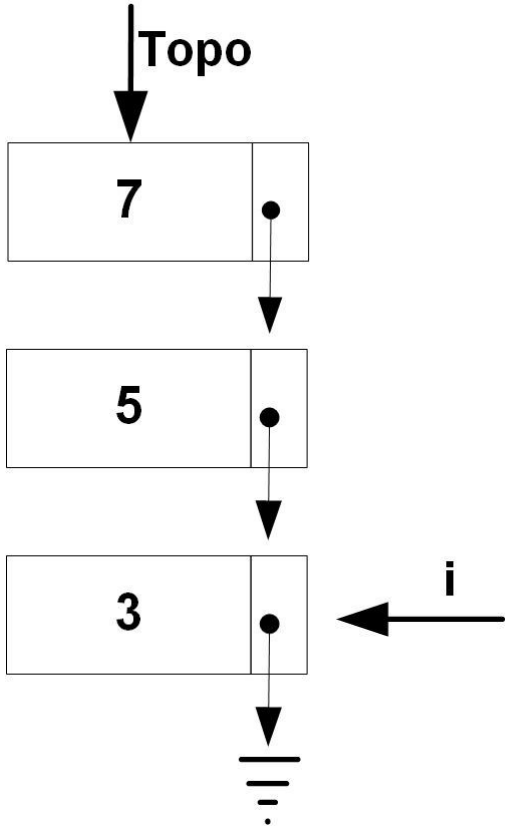
```
[ 7 5
```

Mostrar

```
class Pilha {
    private Celula topo;
    public Pilha () {
        topo = null;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```

```
public void mostrar() {
    System.out.print("[ ");
    for (Celula i = topo; i != null; i = i.prox){
        System.out.print(i.elemento + " ");
    }
    System.out.println("]");
}
```

true



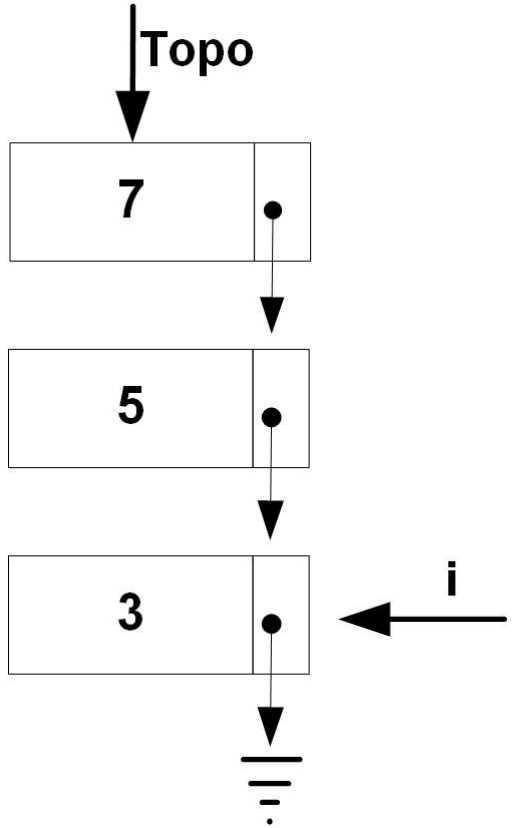
Saída na tela

```
[ 7 5
```

Mostrar

```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```

```
public void mostrar() {
      System.out.print("[ ");
      for (Celula i = topo; i != null; i = i.prox){
         System.out.print(i.elemento + " ");
      }
      System.out.println("]");
}
```



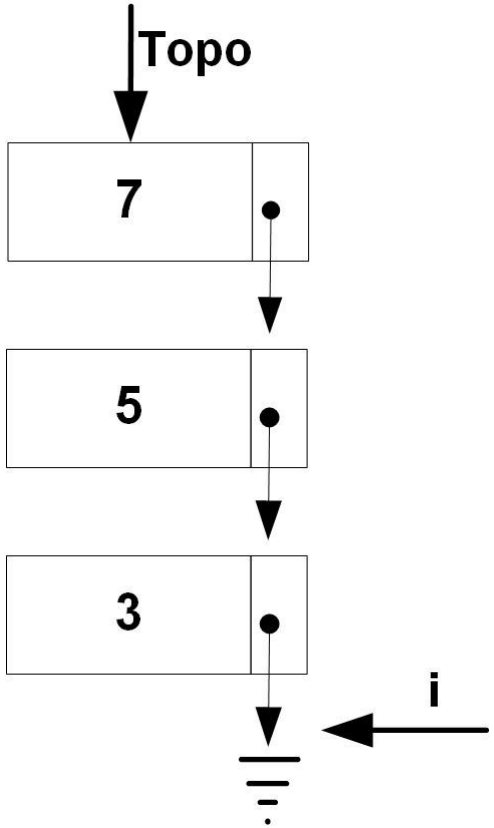
Saída na tela

```
[ 7 5 3
```

Mostrar

```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```

```
public void mostrar() {
      System.out.print("[ ");
      for (Celula i = topo; i != null; i = i.prox){
         System.out.print(i.elemento + " ");
      }
      System.out.println("]");
}
```



Saída na tela

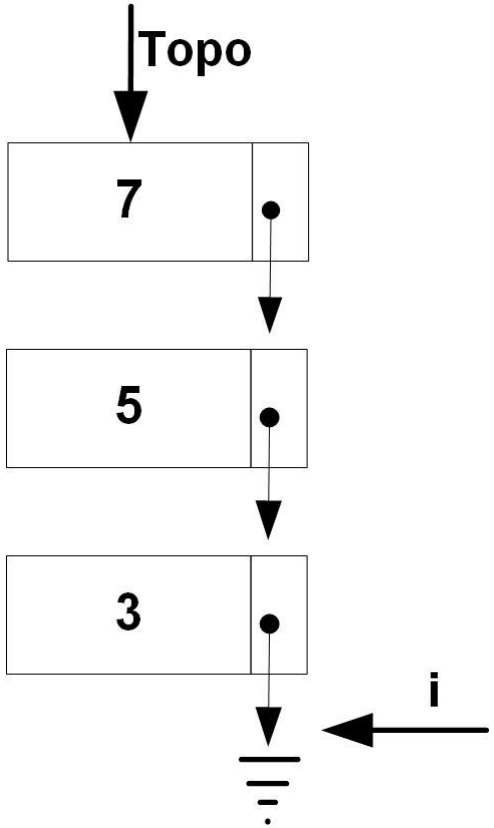
```
[ 7 5 3
```

Mostrar

```
class Pilha {
    private Celula topo;
    public Pilha () {
        topo = null;
    }
    public void inserir(int x) { ... }
    public int remover() { ... }
    public void mostrar() { ... }
}
```

```
public void mostrar() {
    System.out.print("[ ");
    for (Celula i = topo; i != null; i = i.prox){
        System.out.print(i.elemento + " ");
    }
    System.out.println("]");
}
```

false



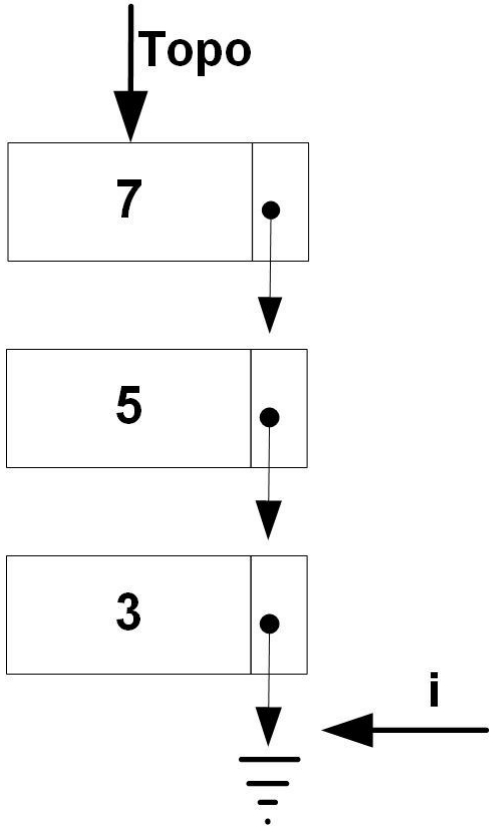
Saída na tela

```
[ 7 5 3
```

Mostrar

```
class Pilha {
        private Celula topo;
        public Pilha () {
                topo = null;
        }
        public void inserir(int x) { ... }
        public int remover() { ... }
        public void mostrar() { ... }
}
```

```
public void mostrar() {
      System.out.print("[ ");
      for (Celula i = topo; i != null; i = i.prox){
         System.out.print(i.elemento + "  ");
      }
      System.out.println("]");
}
```



Saída na tela

```
[ 7  5  3 ]
```

Exercício Resolvido (1)

- Seja nossa Pilha, faça um método que retorna soma dos elementos contidos na mesma

Exercício Resolvido (1)

- Seja nossa Pilha, faça um método que retorna soma dos elementos contidos na mesma

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

Exercício (1)

- Seja nossa Pilha, faça um método RECURSIVO que retorna soma dos elementos contidos na mesma

Exercício (2)

- Seja nossa Pilha, faça um método que retorna o maior elemento contido na mesma

Exercício (3)

- Seja nossa Pilha, faça um método RECURSIVO que retorna o maior elemento contido na mesma

Exercício (4)

- Seja nossa Pilha, faça um método RECURSIVO para mostrar os elementos da pilha na ordem em que os mesmos serão removidos

Exercício (5)

- Seja nossa Pilha, faça um método RECURSIVO para mostrar os elementos da pilha na ordem em que os mesmos foram inseridos

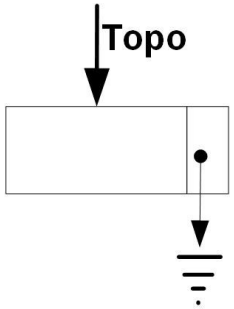
Exercício (7)

- Seja nossa Pilha, faça um método ITERATIVO para Mostar a média dos elementos da pilha.

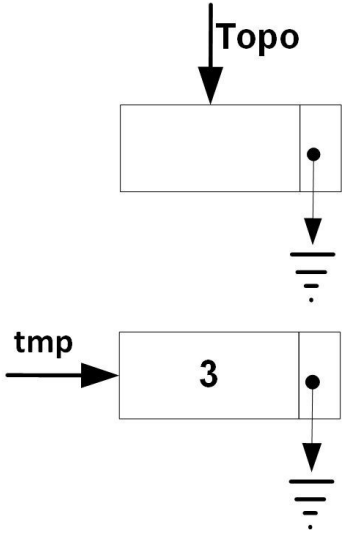
Exercício (8)

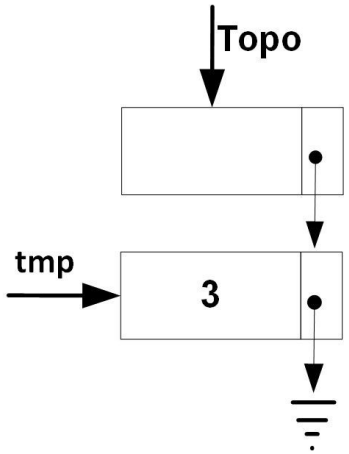
- As ilustrações abaixo mostram a execução dos métodos construtor e do inserir de uma pilha, apresente o código dessa classe e desses métodos

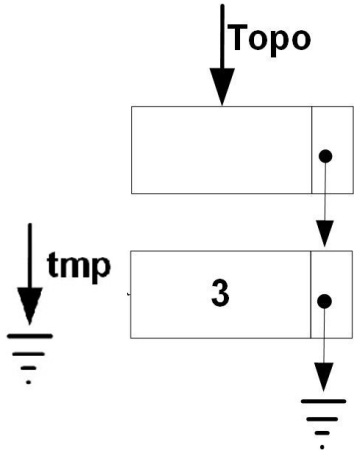
(1) Construtor



(2) Inserir 3



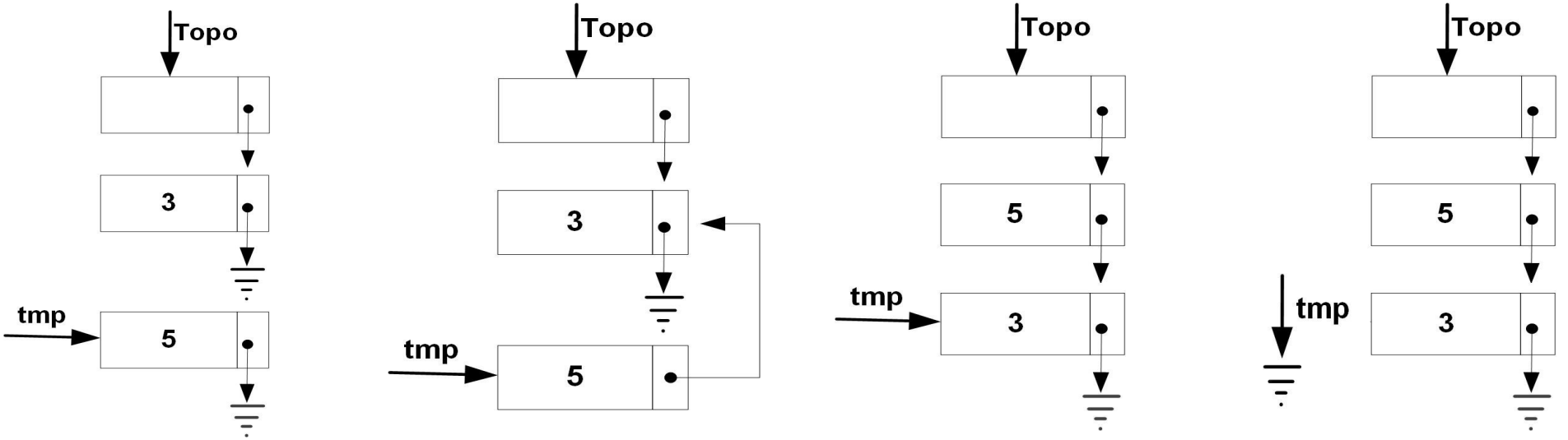




Exercício (8)

- As ilustrações abaixo mostram a execução dos métodos construtor e do inserir de uma pilha, apresente o código dessa classe e desses métodos

(3) Inserir 5



Próxima Aula

- **Tipos Abstratos de Dados Lineares: Lista**