

Algoritmos e Estruturas de Dados

Pilhas de Referências para Objetos

Autores:

Carlos Urbano

Catarina Reis

João Ramos

José Magno

Marco Ferreira

Pilhas

- Uma pilha é uma estrutura abstrata de dados (ADT) em que o **último** elemento a ser **colocado** nesta é o **primeiro** a ser **retirado** (LIFO - *last in first out*), isto é, apenas se tem **acesso** ao elemento do **topo da pilha**
- Uma pilha pode ser implementada utilizando, por exemplo, um *array* ou uma lista
 - Vantagens e desvantagens?
- Apenas vamos abordar a implementação de um pilha recorrendo a uma lista (ou de forma semelhante a uma lista)

Pilhas

- Operações básicas sobre Pilhas:
 - criar uma pilha vazia
 - inserir um elemento no topo da pilha
 - remover o elemento do topo da pilha
 - consultar o elemento do topo da pilha
 - verificar se está vazia

Pilhas

- Classes a construir de seguida

Definição

interface *<ColecaoNaalteravel<T>*

Implementação

classe **NoPilha<T>**

classe **No**

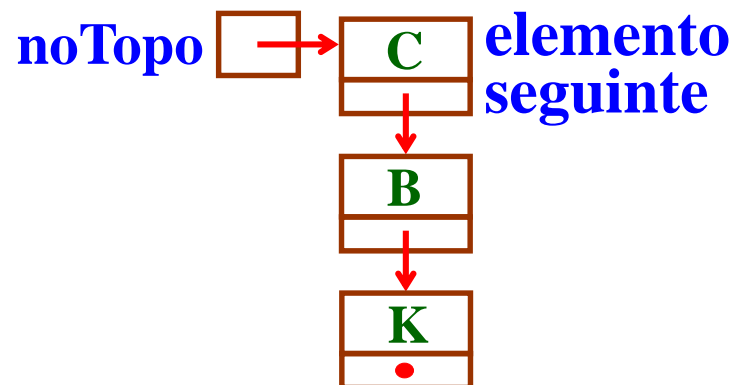
Utilização

classe **Pessoa**

classe **MainTeoricaPilha**

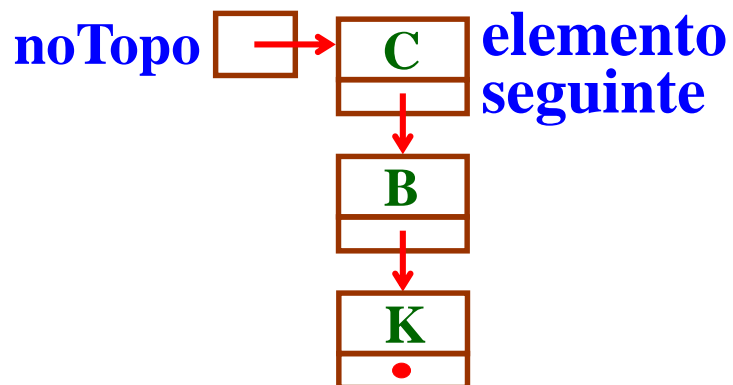
Pilhas

- Uma **pilha** de referências para objetos pode ser definida do seguinte modo:
- **Estrutura**



Pilhas

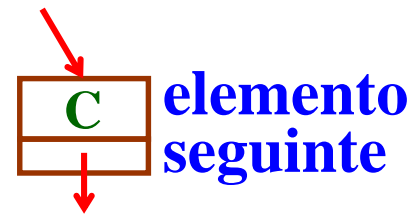
- Uma **pilha** de referências para objetos pode ser definida do seguinte modo:
- **Estrutura**



```
public class Pilha<T> implements ColecaoNaoIteravel<T> {  
  
    private No noTopo;
```

Pilhas

- Um **nó** de uma pilha



Pilhas

- Um **nó** de uma pilha pode ser **implementado** pela **classe** interna **Pilha<T>.No** :

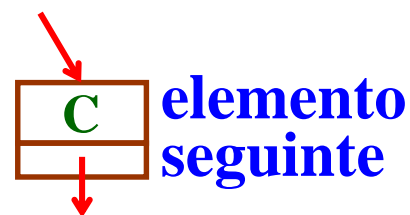
```
class No implements Serializable {
```

```
    T elemento;  
    No seguinte;
```

```
    // Criação de nó com elem antes o topo
```

```
    public No(T elem) {  
        elemento = elem;  
        seguinte = noTopo;  
    }
```

```
}
```



Pilhas

- Um **nó** de uma pilha pode ser **implementado** pela **classe** interna **Pilha<T>.No** :

```
class No implements Serializable {
```

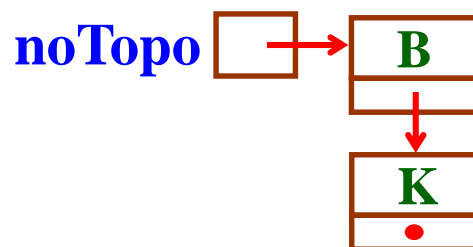
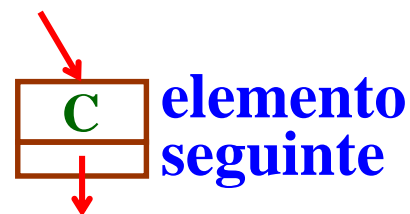
```
    T elemento;  
    No seguinte;
```

```
    // Criação de nó com elem antes o topo
```

```
    public No(T elem) {  
        elemento = elem;  
        seguinte = noTopo;
```

```
    }
```

```
}
```



Pilhas

- Um **nó** de uma pilha pode ser **implementado** pela **classe** interna **Pilha<T>.No** :

```
class No implements Serializable {
```

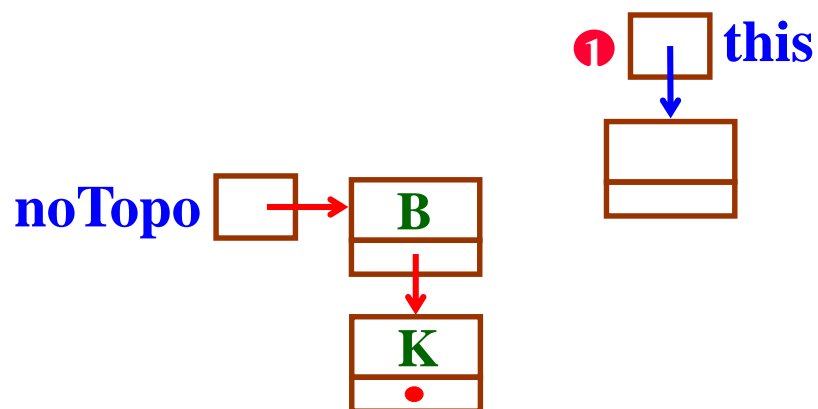
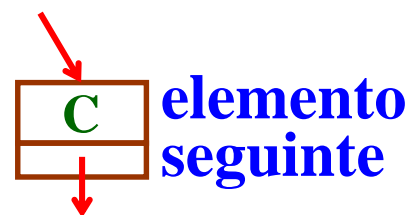
```
    T elemento;  
    No seguinte;
```

```
    // Criação de nó com elem antes o topo
```

```
    public No(T elem) {  
        elemento = elem;  
        seguinte = noTopo;
```

```
    }
```

```
}
```



Pilhas

- Um **nó** de uma pilha pode ser **implementado** pela **classe interna** `Pilha<T>.No` :

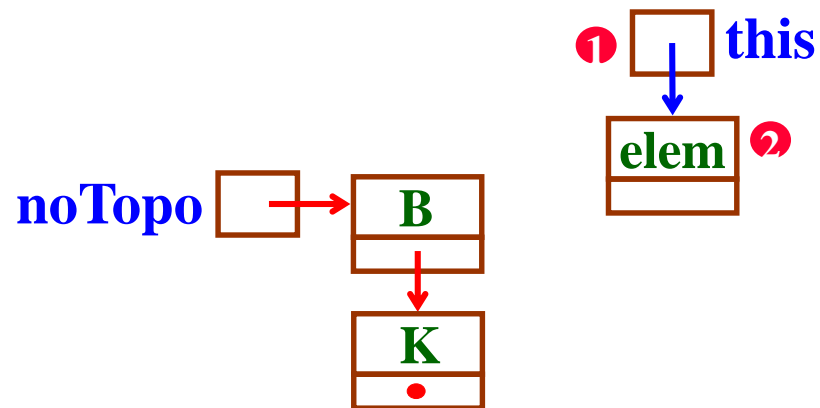
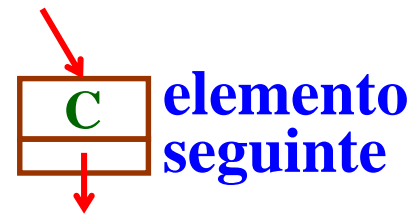
```
class No implements Serializable {
```

```
    T elemento;  
    No seguinte;
```

```
    // Criação de nó com elem antes o topo
```

```
    public No(T elem) {  
        elemento = elem;  
        seguinte = noTopo;  
    }
```

```
}
```



Pilhas

- Um **nó** de uma pilha pode ser **implementado** pela **classe interna** `Pilha<T>.No` :

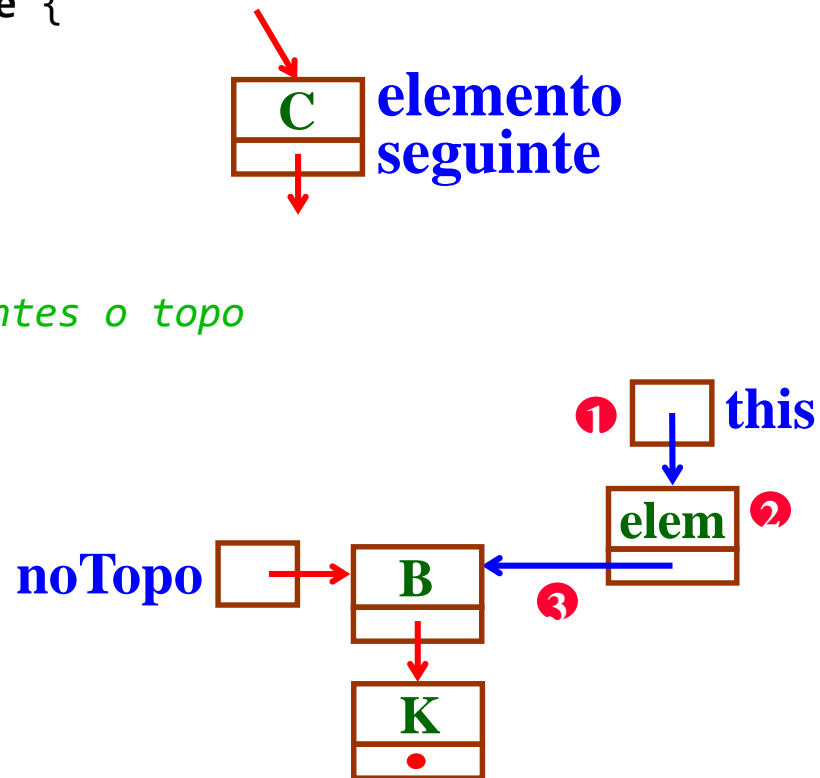
```
class No implements Serializable {
```

```
    T elemento;  
    No seguinte;
```

```
    // Criação de nó com elem antes o topo
```

```
    public No(T elem) {  
        elemento = elem;  
        seguinte = noTopo;  
    }
```

```
}
```



Pilhas

- Criação

noTopo 

Pilhas

- Criação

noTopo 

```
public Pilha() {  
    noTopo = null;  
}
```

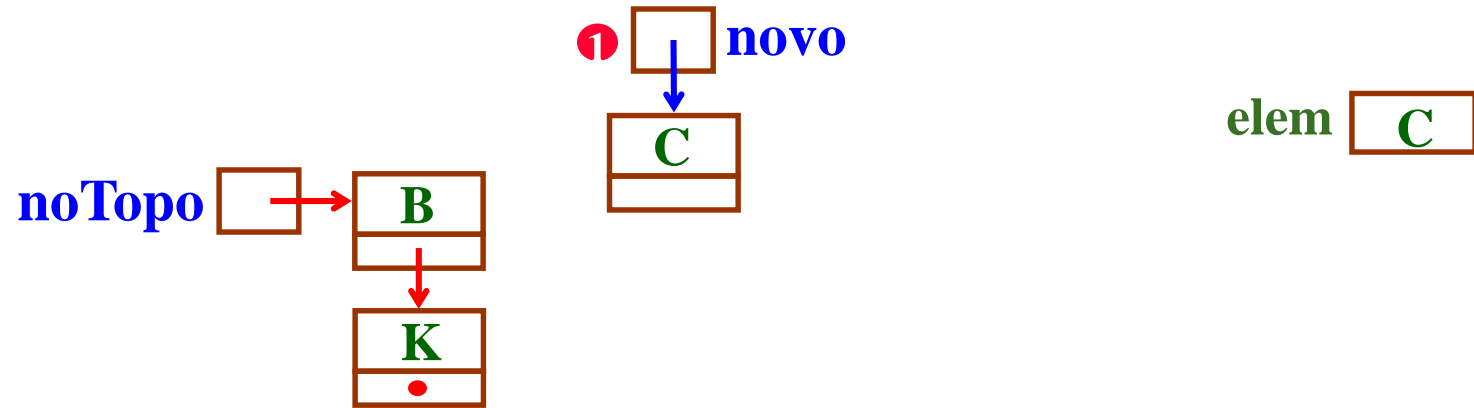
Pilhas

- Inserção



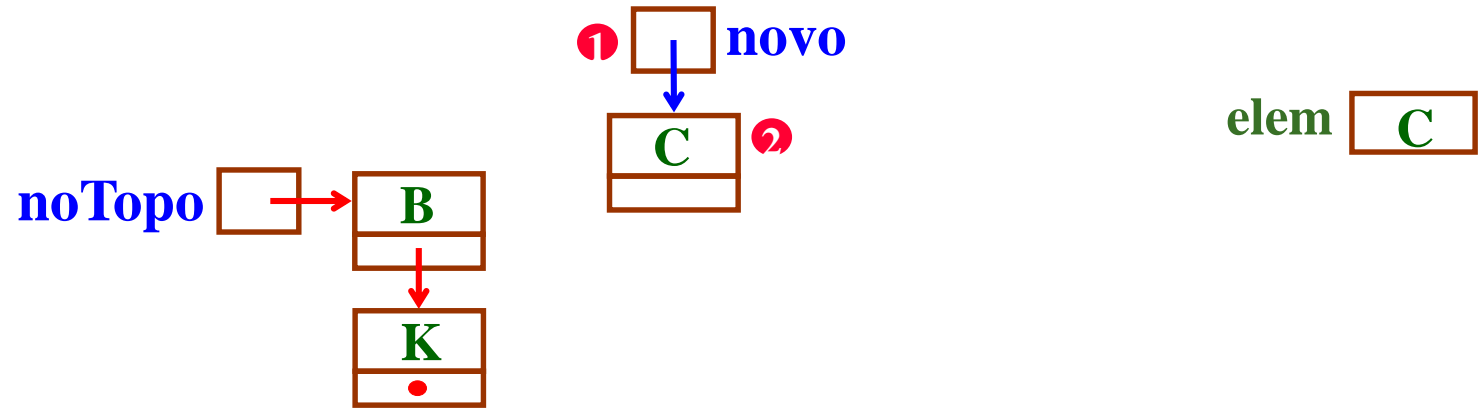
Pilhas

- Inserção



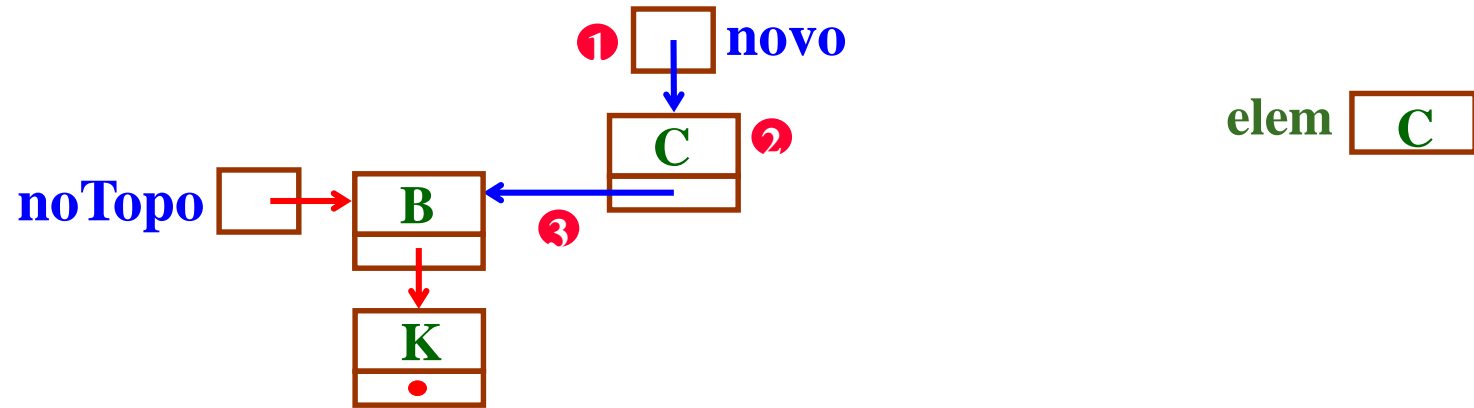
Pilhas

- Inserção



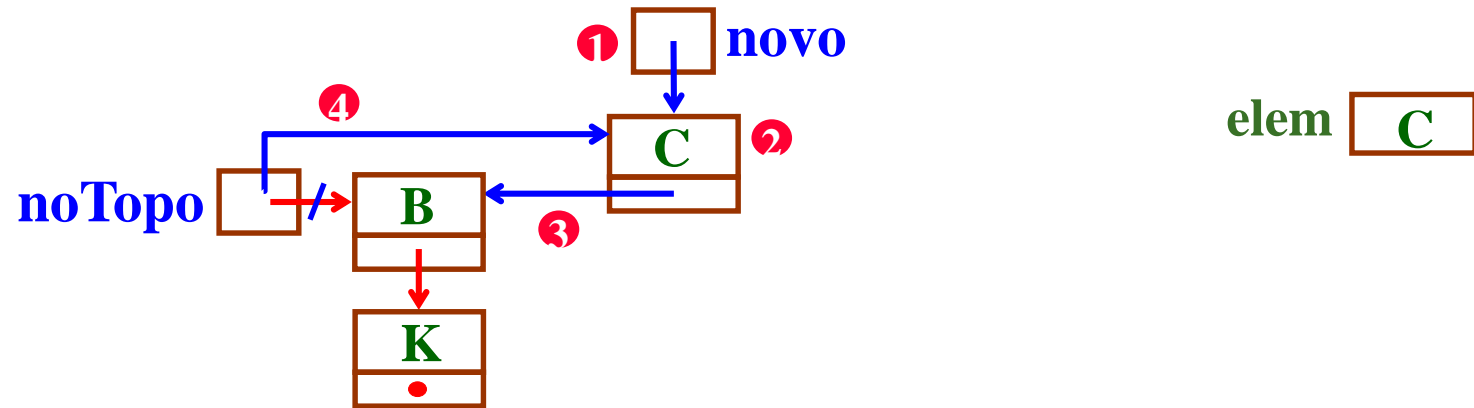
Pilhas

- Inserção



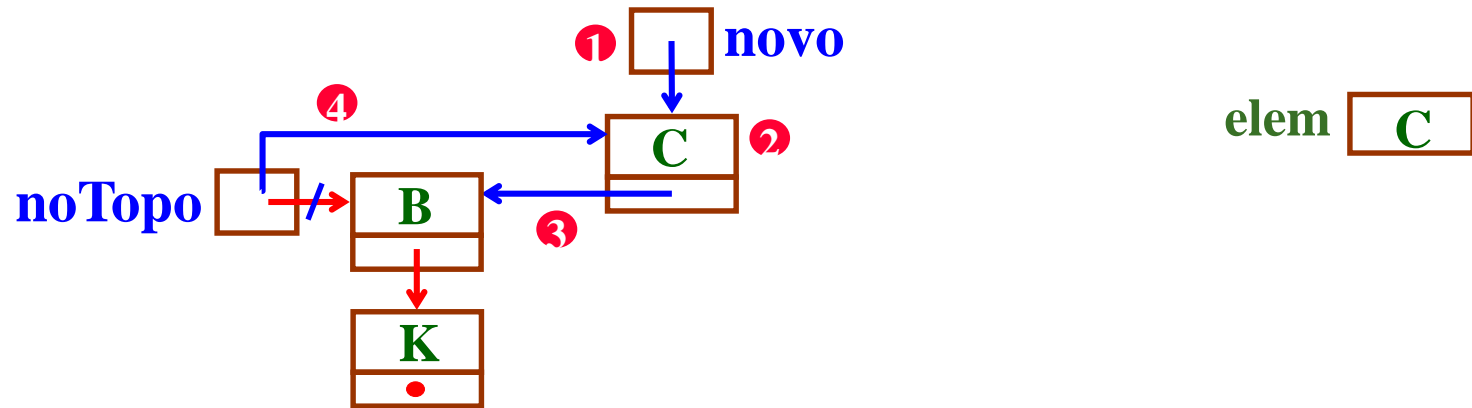
Pilhas

- Inserção



Pilhas

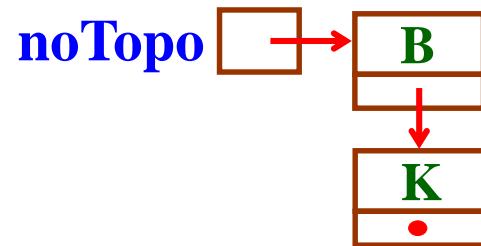
- Inserção



```
public void inserir(T elem) {  
    noTopo = new No(elem);  
}
```

Pilhas

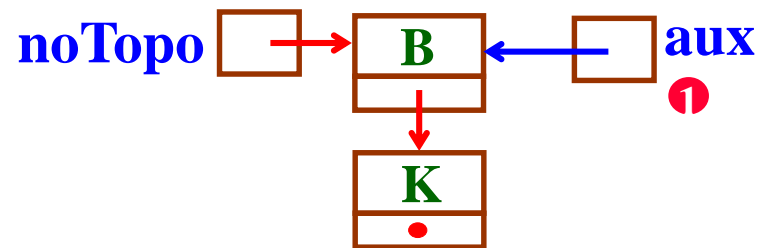
- Remoção



- remove e devolve o elemento do topo da pilha ou `null` caso esteja vazia

Pilhas

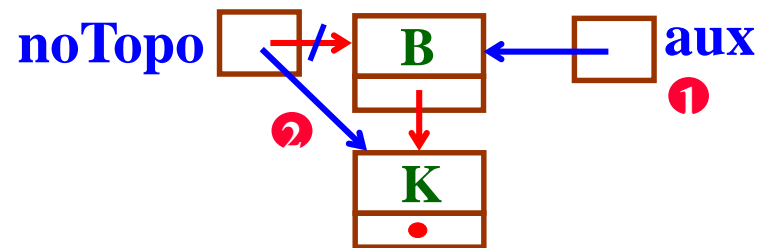
- Remoção



- remove e devolve o elemento do topo da pilha ou null caso esteja vazia

Pilhas

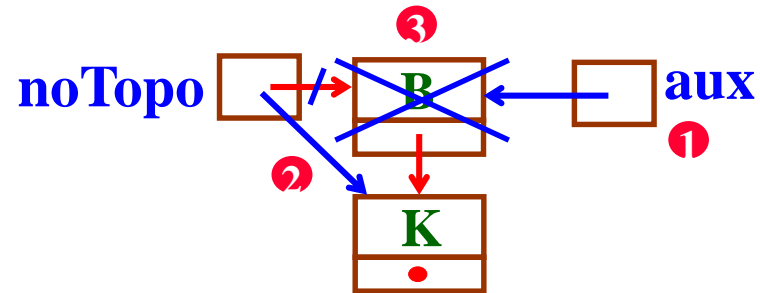
- Remoção



- remove e devolve o elemento do topo da pilha ou null caso esteja vazia

Pilhas

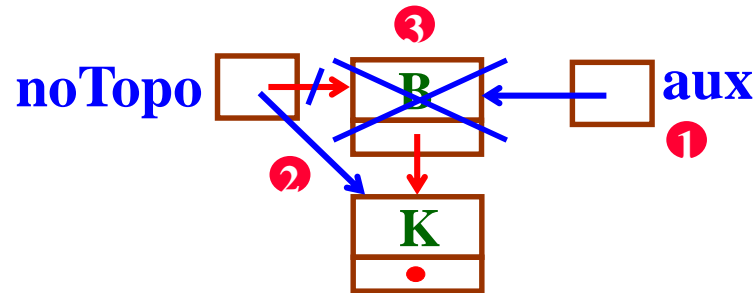
- **Remoção**



- remove e devolve o elemento do topo da pilha ou null caso esteja vazia

Pilhas

- Remoção



- remove e devolve o elemento do topo da pilha ou null caso esteja vazia

```
public T remover() {  
    if (isVazia()) {  
        return null;  
    }  
    T aux = noTopo.elemento;  
    noTopo = noTopo.seguinte;  
    return aux;  
}
```

Pilhas

- Consulta

```
public T consultar() {  
    return isVazia() ? null : noTopo.elemento;  
}
```

- devolve o elemento do topo da pilha ou null caso esta esteja vazia

- Vazia

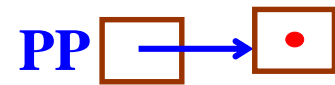
```
public boolean isEmpty() {  
    return noTopo == null;  
}
```

Pilhas

- Exemplo de Utilização

```
public class MainTeoricaPilha {  
  
    public static void main(String[] args) {  
        new MainTeoricaPilha();  
    }  
  
    public MainTeoricaPilha() {  
        Pilha<Pessoa> pilhaPessoas = new Pilha<>();  
  
        pilhaPessoas.inserir(new Pessoa(3, "B"));  
  
        pilhaPessoas.inserir(new Pessoa(1, "C"));  
  
        pilhaPessoas.inserir(new Pessoa(2, "A"));  
    }  
}
```

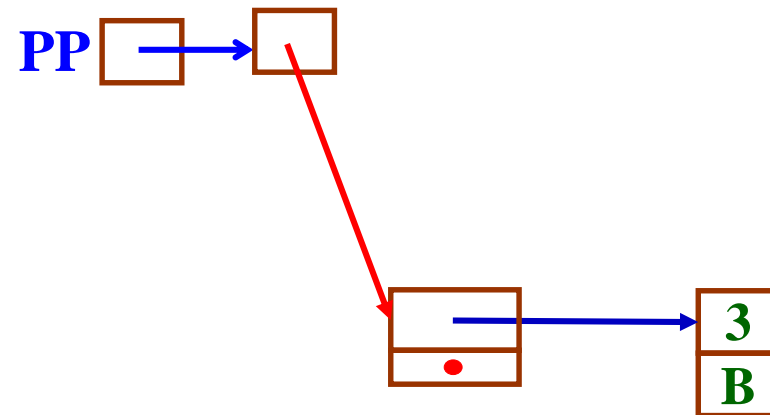
Pilhas



Pilhas

```
public class MainTeoricaPilha {  
  
    public static void main(String[] args) {  
        new MainTeoricaPilha();  
    }  
  
    public MainTeoricaPilha() {  
        Pilha<Pessoa> pilhaPessoas = new Pilha<>();  
  
        pilhaPessoas.inserir(new Pessoa(3, "B"));  
  
        pilhaPessoas.inserir(new Pessoa(1, "C"));  
  
        pilhaPessoas.inserir(new Pessoa(2, "A"));  
    }  
}
```

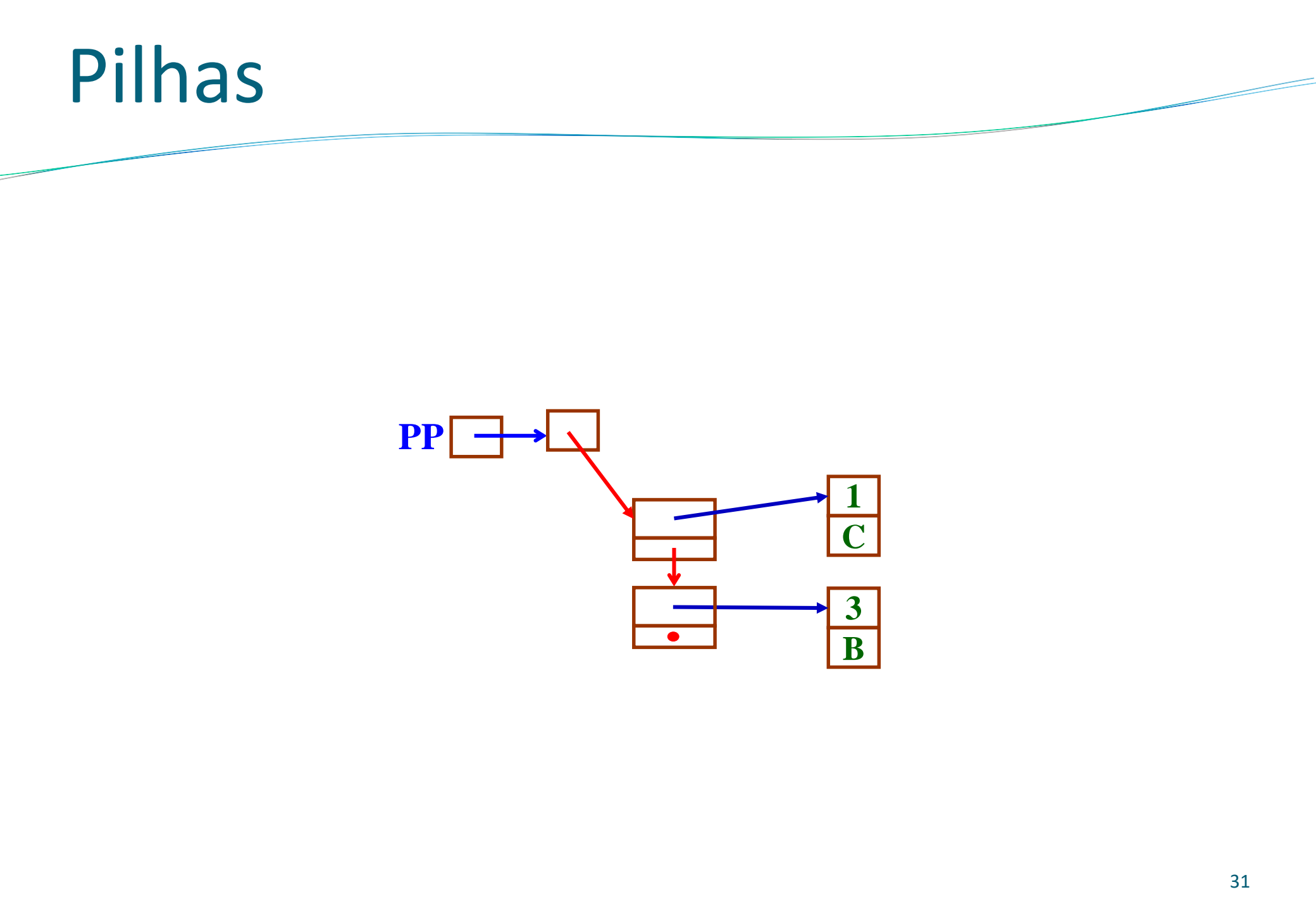
Pilhas



Pilhas

```
public class MainTeoricaPilha {  
  
    public static void main(String[] args) {  
        new MainTeoricaPilha();  
    }  
  
    public MainTeoricaPilha() {  
        Pilha<Pessoa> pilhaPessoas = new Pilha<>();  
  
        pilhaPessoas.inserir(new Pessoa(3, "B"));  
  
        pilhaPessoas.inserir(new Pessoa(1, "C"));  
  
        pilhaPessoas.inserir(new Pessoa(2, "A"));  
    }  
}
```

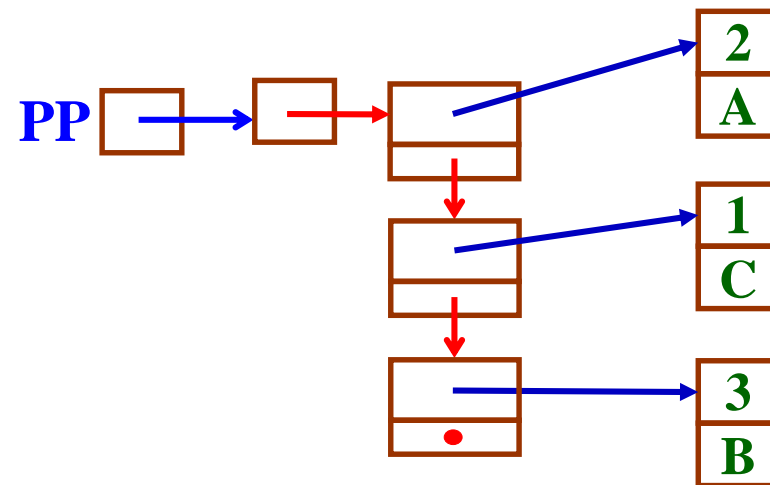
Pilhas



Pilhas

```
public class MainTeoricaPilha {  
  
    public static void main(String[] args) {  
        new MainTeoricaPilha();  
    }  
  
    public MainTeoricaPilha() {  
        Pilha<Pessoa> pilhaPessoas = new Pilha<>();  
  
        pilhaPessoas.inserir(new Pessoa(3, "B"));  
  
        pilhaPessoas.inserir(new Pessoa(1, "C"));  
  
        pilhaPessoas.inserir(new Pessoa(2, "A"));  
    }  
}
```

Pilhas



Pilhas

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

```
pilhaPessoas.remove();
```

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

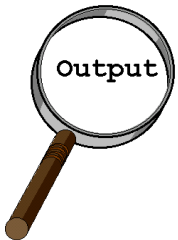
```
System.out.println(pilhaPessoas.consultar());
```

```
pilhaPessoas.remove();
```

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

```
}
```

```
}
```



```
pilhaPessoas  
Pilha = {  
BI: 2 Nome: A  
BI: 1 Nome: C  
BI: 3 Nome: B  
}
```



Pilhas

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

```
pilhaPessoas.remove();
```

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

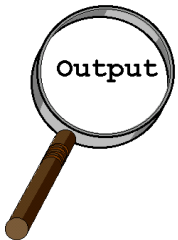
```
System.out.println(pilhaPessoas.consultar());
```

```
pilhaPessoas.remove();
```

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

```
}
```

```
}
```



```
pilhaPessoas  
Pilha = {  
BI: 2 Nome: A  
BI: 1 Nome: C  
BI: 3 Nome: B  
}
```



Pilhas

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

```
pilhaPessoas.remove();
```

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

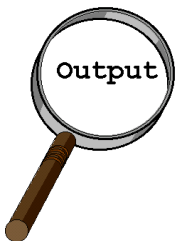
```
System.out.println(pilhaPessoas.consultar());
```

```
pilhaPessoas.remove();
```

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

```
}
```

```
}
```



```
pilhaPessoas  
Pilha = {  
BI: 2 Nome: A  
BI: 1 Nome: C  
BI: 3 Nome: B  
}
```

```
pilhaPessoas  
Pilha = {  
BI: 1 Nome: C  
BI: 3 Nome: B  
}
```

Pilhas

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

```
pilhaPessoas.remove();
```

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

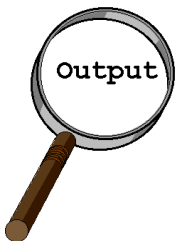
```
System.out.println(pilhaPessoas.consultar());
```

```
pilhaPessoas.remove();
```

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

```
}
```

```
}
```



```
pilhaPessoas  
Pilha = {  
BI: 2 Nome: A  
BI: 1 Nome: C  
BI: 3 Nome: B  
}
```

```
pilhaPessoas  
Pilha = {  
BI: 1 Nome: C  
BI: 3 Nome: B  
}  
  
BI: 1 Nome: C
```

Pilhas

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

```
pilhaPessoas.remove();
```

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

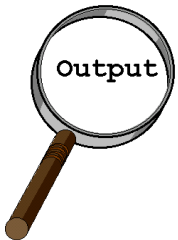
```
System.out.println(pilhaPessoas.consultar());
```

```
pilhaPessoas.remove();
```

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

```
}
```

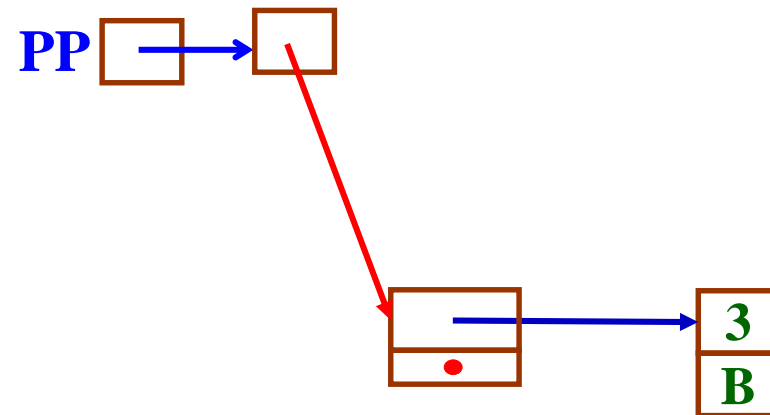
```
}
```



```
pilhaPessoas  
Pilha = {  
BI: 2 Nome: A  
BI: 1 Nome: C  
BI: 3 Nome: B  
}
```

```
pilhaPessoas  
Pilha = {  
BI: 1 Nome: C  
BI: 3 Nome: B  
}  
  
BI: 1 Nome: C
```

Pilhas



Pilhas

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

```
pilhaPessoas.remove();
```

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

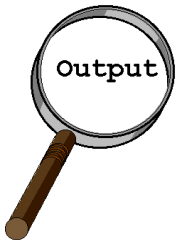
```
System.out.println(pilhaPessoas.consultar());
```

```
pilhaPessoas.remove();
```

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

```
}
```

```
}
```



```
pilhaPessoas  
Pilha = {  
BI: 2 Nome: A  
BI: 1 Nome: C  
BI: 3 Nome: B  
}
```

```
pilhaPessoas  
Pilha = {  
BI: 1 Nome: C  
BI: 3 Nome: B  
}  
  
BI: 1 Nome: C
```

```
pilhaPessoas  
Pilha = {  
BI: 3 Nome: B  
}
```