Pilhas

Bibliografia

- LAFORE, Robert (Robert W.). **Estruturas de dados & algoritmos em Java**. Rio de Janeiro : Ciência Moderna, 2004. 702 p, il.
- PREISS, B. R. **Estruturas de Dados e Algoritmos**. 3ª. ed. Rio de Janeiro: Elsevier, 2000.

Introdução

- A pilha é uma estrutura de dados que possui as seguintes características:
  - Novos elementos são adicionados sempre numa extremidade da estrutura de dados – denominada de **topo da pilha**
  - Não é possível percorrer a estrutura de dados
  - É possível acessar o elemento que estiver no topo da pilha
  - Somente o elemento que estiver no topo da pilha que pode ser removido
    - O último a entrar é o primeiro a sair – LIFO (last in first out)

Exemplos de aplicação

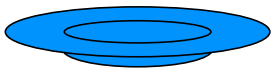
- Exemplos de onde podem ser usadas:
  - Na análise e resolução de expressões aritméticas:
    - [2 x (10 – 4)] / 3
  - Os processadores possuem arquitetura baseada em pilha
    - Quando uma rotina é chamada, o endereço de retorno e os parâmetros são colocados numa pilha
  - Algoritmos de manipulação de outras estruturas de dados
    - Árvore binárias
    - Grafos

Introdução

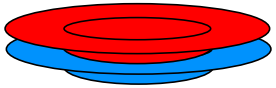
- Operações essenciais executadas na estrutura:
  - Empilhar (*push*) elementos na estrutura de dados
    - Sempre adiciona no topo da pilha
  - Desempilhar (*pop*) elementos da estrutura de dados
    - Sempre remove do topo da pilha

Exemplo de manipulação

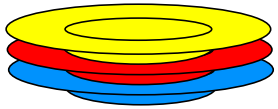
push(pratoazul)



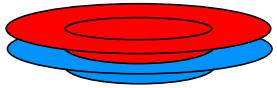
push(pratovermelho)



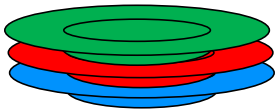
push(pratoamarelo)



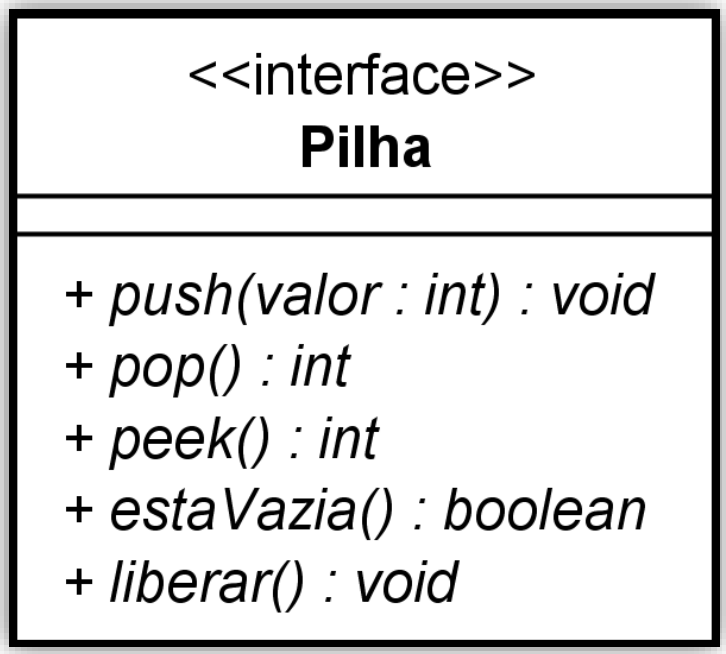
pop()



push(pratoverde)



Interface para implementação de pilhas

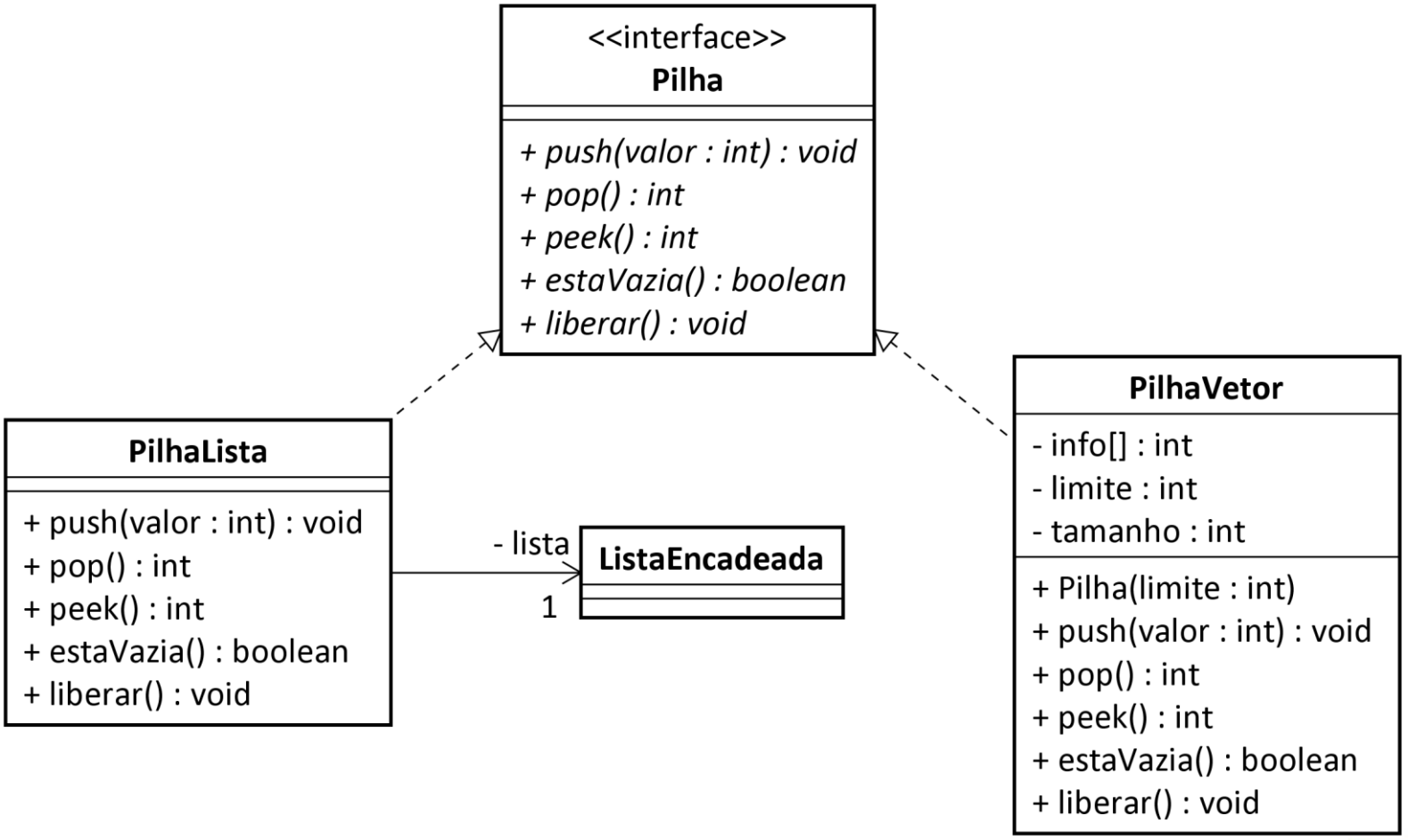


Método	Descrição
`push()`	Incluir um valor na pilha
`pop()`	Retira um valor da pilha. Lança exceção quando não consegue
`peek()`	Retorna o valor armazenado no topo da pilha
`estaVazia()`	Verifica se a pilha está vazia
`liberar()`	Remove todos os dados armazenados na pilha

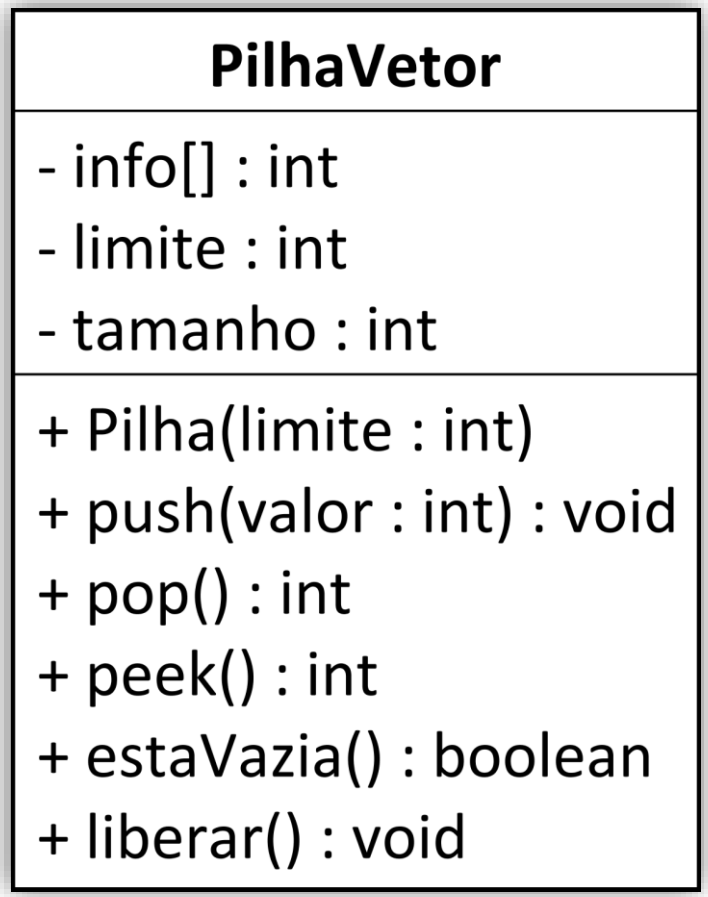
Interface Pilha em Java

```
package pilha;

public interface Pilha {

    void push(int v);
    int pop ();
    int peek();
    boolean estaVazia();
    void liberar();

}
```

Exemplo de projeto em Java

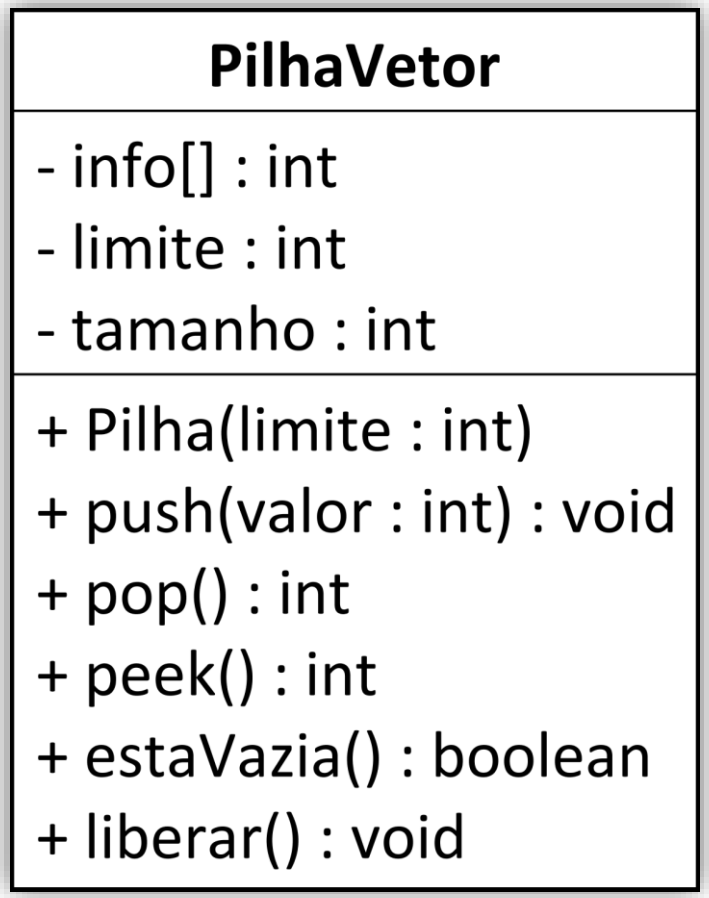


Implementação com vetor


PilhaVetor
- info[] : int
- limite : int
- tamanho : int
+ Pilha(limite : int)
+ push(valor : int) : void
+ pop() : int
+ peek() : int
+ estaVazia() : boolean
+ liberar() : void

Implementação estática

- Um vetor é utilizado para armazenar os dados da pilha
- O tamanho do vetor é estabelecido durante a criação da pilha. Este valor é armazenado na variável **`limite`**
- Os elementos que são inseridos ocupam as primeiras posições livres do vetor.
- A variável **`tamanho`** é utilizada para indicar quantos elementos já foram inseridos. Usada também para obter o topo



Implementação de pilha com vetor

```
package pilha;

public class PilhaVetor implements Pilha {

    private int limite;
    private int tamanho;
    private int[] info;

    // métodos da interface Pilha

}
```

Criação de pilha

- Estabelece tamanho máximo da pilha
- Aloca o vetor encapsulado
- Inicializa atributos de tamanho atual da pilha

Algoritmo: **PilhaVetor(int limite)**

info ← new int[limite];
this.limite ← limite;
this.tamanho ← 0;

- Dica: Em Java, para criar um vetor de genérico, executar:

```
info = (T[]) new Object[limite];
```

Inclusão de elemento na pilha

Algoritmo**: push(int valor)**

se (limite = tamanho) **então**
   throw new RuntimeException("Capacidade esgotada da pilha");
fim-se

info[tamanho] ← valor;
tamanho ← tamanho + 1;

Obter o topo da pilha

Algoritmo**: peek( )**

se (estaVazia( )) **então**
   throw new RuntimeException("Pilha está vazia");
fim-se

retornar info[tamanho-1];

Remover elemento da pilha

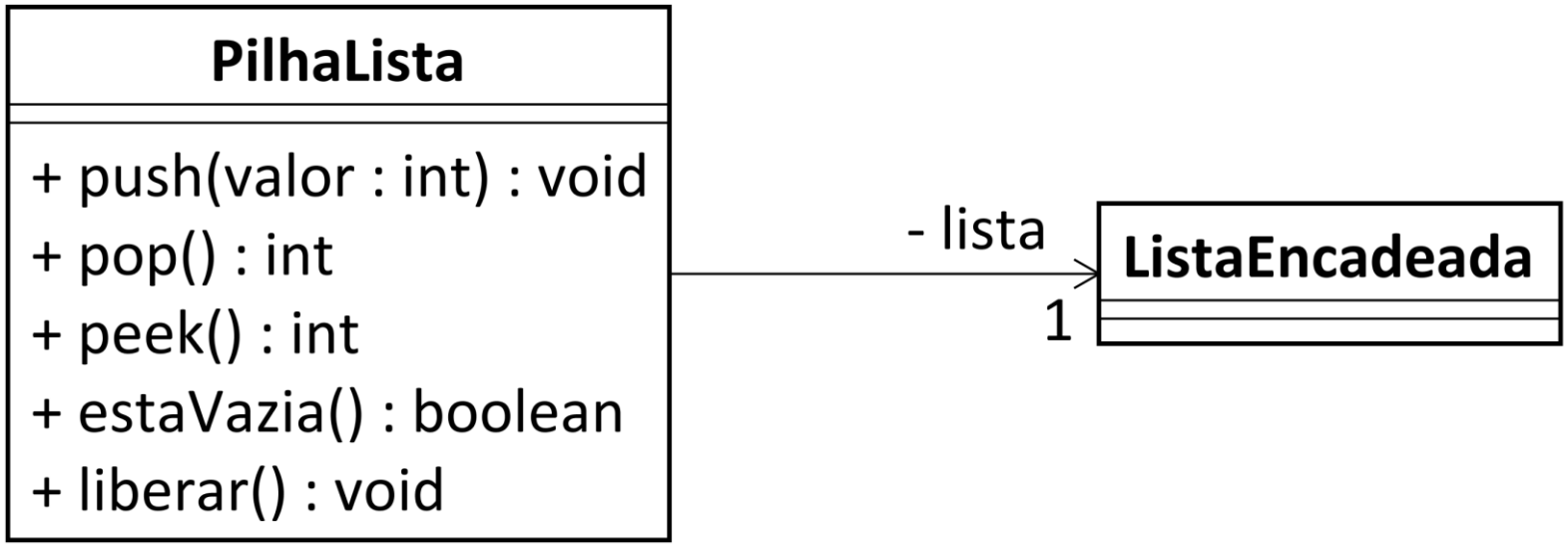
- Retira elemento da pilha e retorna o valor retirado

```
Algoritmo: int pop( )

int valor;
valor ← peek( );

tamanho ← tamanho – 1;
retornar valor;
```

Se a pilha for de objetos, é preciso remover aqui a referência do objeto removido

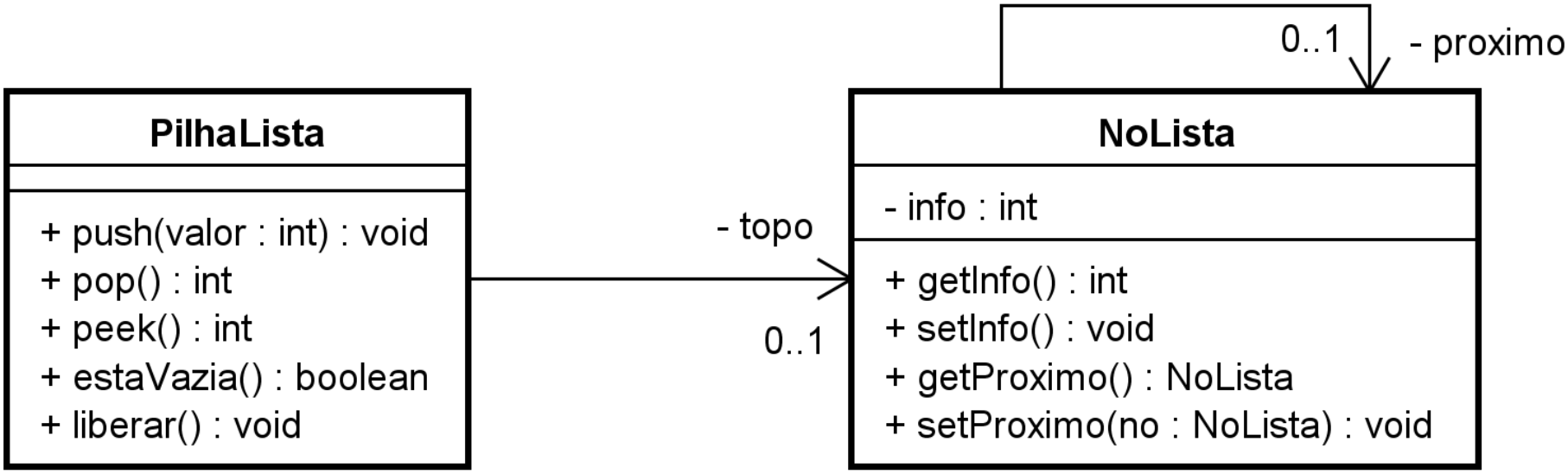
Implementação com lista

Projeto 1 - Implementação com lista encadeada



- Uma lista simplesmente encadeada é utilizada para implementar a pilha

Projeto 2 - alternativa para implementação de pilha através de lista encadeada



- Este caso considera que não existe uma classe `Lista` para ser reutilizada. A classe `PilhaLista`, além de implementar a interface `Pilha`, também mantém o encadeamento dos dados empilhados.
- A variável `topo` tem o mesmo papel da variável `primeiro` da lista encadeada.

Exemplo em Java – Projeto 1

```
package pilha;

public class PilhaLista implements Pilha {

    private ListaEncadeada lista;

    // métodos

}
```

Implementação de pilha por lista – Projeto 1

- Algoritmo para criar uma nova pilha

```
Algoritmo: PilhaLista( )

lista ← new ListaEncadeada( );
```

- Algoritmo para inserir um elemento na pilha

```
Algoritmo: push(int info)

lista.inserir(info);
```

Obter o topo da pilha

Algoritmo**: peek( )**

se (estaVazia( )) **então**
   throw new RuntimeException("Pilha está vazia");
fim-se

retornar lista.primeiro.info;

Implementação de pilha por lista – Projeto 1

- Algoritmo para desempilhar um elemento

Algoritmo**: int pop()**

int valor;
valor ← peek( );
lista.retirar(valor);

retornar valor;