Pilhas

1

Introdução

- 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 está no topo da pilha
  - Somente o elemento que está no topo da pilha pode ser removido
    - O último a entrar é o primeiro a sair – LIFO (*last in first out*)

3

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

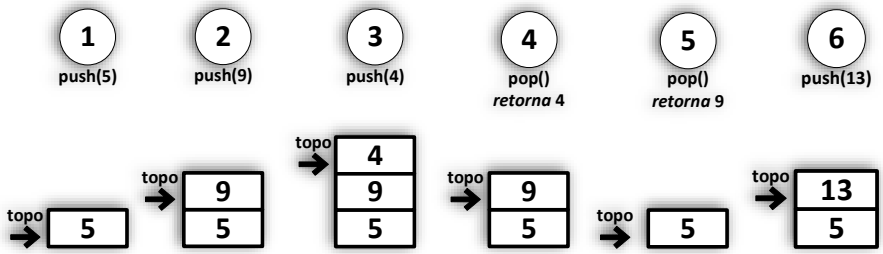
4

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

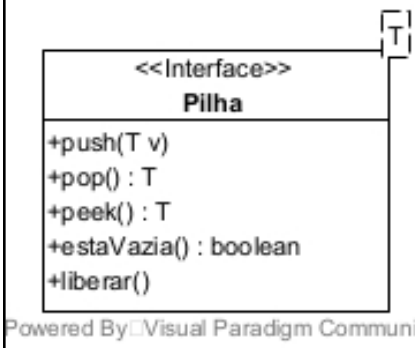
5

Exemplo de manipulação



6

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

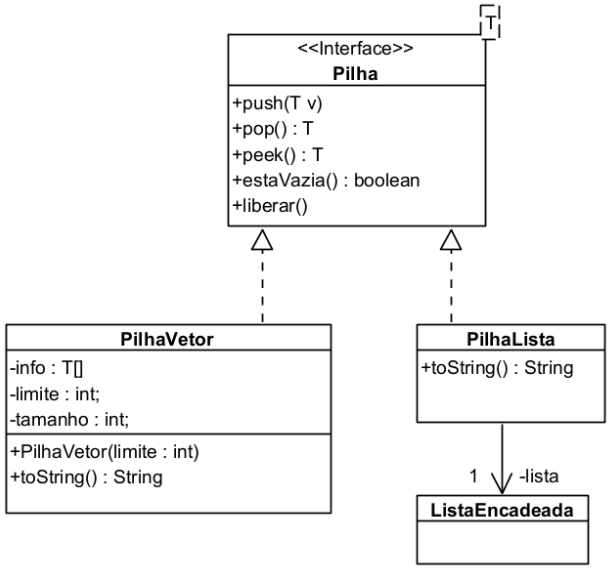
7

Interface Pilha em Java

```
package pilha;

public interface Pilha<T> {

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

}
```

8

Exemplo de projeto em Java



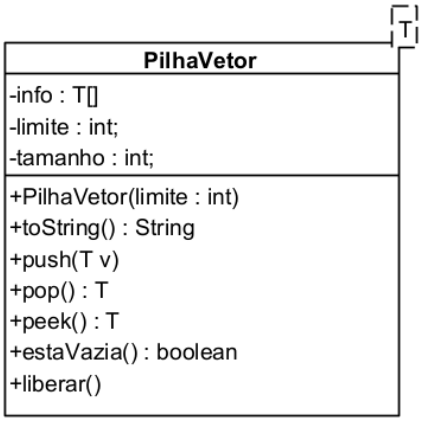
9

Implementação estática (com vetor)

10

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



11

Implementação de pilha com vetor

```
package pilha;

public class PilhaVetor<T> implements Pilha<T> {

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

    // métodos da interface Pilha

}
```

12

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 T[limite]**;**
this.limite ← limite;
this.tamanho ← 0;

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

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

13

Inclusão de elemento na pilha

```
Algoritmo: push(T valor)

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

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

14

Obter o topo da pilha

```
Algoritmo: T peek( )

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

retornar info[tamanho-1];
```

15

Remover elemento da pilha

- Retira elemento da pilha e retorna o valor retirado

```
Algoritmo: T pop( )

T valor;
valor ← peek( );

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

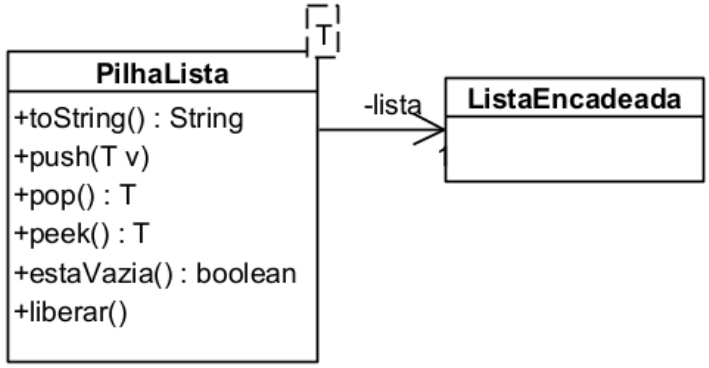
Se a pilha for de objetos, é preciso remover aqui a referência do objeto removido

16

Implementação dinâmica (com lista)

17

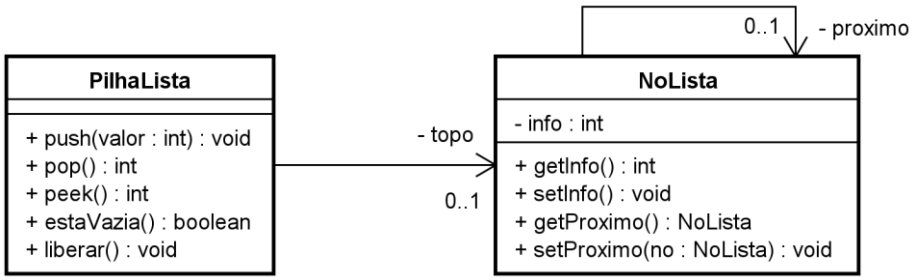
Projeto 1 - Implementação usando lista encadeada



- Uma lista simplesmente encadeada é utilizada para implementar a pilha

18

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.

19

Exemplo em Java – Projeto 1

```
package pilha;

public class PilhaLista<T> implements Pilha<T> {

    private ListaEncadeada<T> lista;

    // métodos

}
```

20

Implementação de pilha usando lista – Projeto 1

- Algoritmo para criar uma nova pilha

```
Algoritmo: PilhaLista( )
lista ← new ListaEncadeada( );
```

- Algoritmo para inserir um elemento na pilha

```
Algoritmo: push(T info)
lista.inserir(info);
```

21

Obter o topo da pilha

```
Algoritmo: T peek( )

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

retornar lista.primeiro.info;
```

22

Implementação de pilha usando lista – Projeto 1

- Algoritmo para desempilhar um elemento

```
Algoritmo: T pop()

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

retornar valor;
```

23