Entendendo as Aplicações de Pilha

Serão apresentadas a seguir as definições das operações de Pilha, a manipulação das operações, a aplicação das operações e as aplicações no Java.

DEFININDO AS OPERAÇÕES DE PILHA

A operação Pilha Vazia é uma função sem parâmetros, que retorna verdadeiro se a Pilha estiver vazia e falso se a Pilha não estiver vazia.

```
lógico PilhaVazia( )
 início_módulo
   se (topo = – 1)
    então
      retornar verdadeiro;
   senão
    retornar falso;
   fimse;
```

```
 fim_módulo;
```

A operação Pilha Cheia é uma função sem parâmetros, que retorna verdadeiro se a Pilha estiver cheia e falso se a Pilha não estiver cheia.

```
lógico PilhaCheia( )
 início_módulo
  se (topo >= tamanho – 1)
   então
    retornar verdadeiro;
   senão
    retornar falso;
  fimse;
 fim_módulo;
```

A operação Empilhar é um procedimento das operações de Pilha que recebe um elemento como parâmetro. Este é o elemento que será empilhado. A operação empilhar verifica se a pilha não está cheia, antes de empilhar o elemento. O elemento não poderá ser empilhado se a pilha estiver cheia. Além disso, ele mostra

uma mensagem para o usuário, caso contrário, há espaço para empilhar e o elemento é empilhado na pilha.

```
Empilhar (elemento numérico_inteiro)
 início_módulo
   se (não PilhaCheia( ))
     então
       topo ← topo + 1;
       vetor[topo] ← elemento;
     senão
       escrever (“Pilha Cheia”);
   fimse;
 fim_módulo;
```

A operação Desempilhar é uma função sem parâmetros das operações de pilha e verifica se a pilha não está vazia antes de desempilhar o elemento. O elemento não poderá ser desempilhado se a pilha estiver vazia, pois não há elemento na pilha para ser desempilhado. Caso a pilha não esteja vazia, então o elemento é desempilhado e retornado.

```
inteiro Desempilhar ()
 início_módulo
  Declarar
   desempilhado ← 0 numérico_inteiro;

  se (PilhaVazia())
   então
    escrever ("Pilha Vazia");
   senão
    desempilhado ← vetor[topo];
    topo ← topo – 1;
  fimse;
  retornar desempilhado;
 fim_módulo;
```

A operação elemento topo é uma operação de manipulação da pilha, que é um procedimento que mostra o elemento do topo da pilha.

```
ElementoTopo( )
 início_módulo
  se (topo >= 0)
    então
       escrever ("O elemento do topo é " , vetor[topo];
    senão
       escrever("Pilha Vazia");
  fimse;
 fim_módulo;
```

A operação Mostrar Pilha é uma operação de manipulação da pilha, que é um procedimento que mostra os elementos da pilha para o usuário.

```
MostrarPilha( )
 início_módulo
  Declarar
   i numérico_inteiro;

  para i de topo até 0 passo –1 faça
```

```
    escrever ("Elemento " , vetor[i] , " posição " , (i+1));
  fimpara;
fim_módulo;
```

MANIPULANDO AS OPERAÇÕES DE PILHA

Para as operações de pilha em Java, utiliza-se a seguinte definição de registro numa classe com o construtor da pilha e com as operações de pilha:

```
class Pilha
{
  int tamanho;
  int topo;
  int vetor [ ];

  Pilha (int tam)
  {
    topo = -1;
```

```
    tamanho = tam;
    vetor = new int[tam];
  }
  // as operações de pilha aparecem aqui
}
```

Onde o campo tamanho armazena a capacidade da pilha, ou seja, até quantos elementos a pilha consegue armazenar; o campo topo que armazena a posição do elemento que está no topo da pilha e o campo vetor, que armazena os elementos que estão armazenados na pilha.

PilhaVazia é um módulo função sem parâmetros da operação pilha vazia, que retorna verdadeiro se a pilha estiver vazia e retorna falso se a pilha não estiver vazia.

```
public boolean PilhaVazia( )
{
  if (topo == -1)
  {
    return true;
  }
```

```
  else
  {
    return false;
  }
}
```

PilhaCheia é um módulo função sem parâmetros da operação pilha cheia, que retorna verdadeiro se a pilha estiver cheia ou falso se a pilha estiver vazia.

```
public boolean PilhaCheia( )
{
  if (topo >= tamanho-1)
  {
    return true;
  }
  else
  {
    return false;
  }
```

```
}
```

Empilhar é um módulo procedimento da operação empilhar que recebe como parâmetro um elemento a ser empilhado. Este módulo verifica se a pilha não está cheia antes de empilhar o elemento. Se a pilha estiver cheia, o elemento não é empilhado e uma mensagem é enviada ao usuário. Se a pilha não estiver cheia, o elemento é empilhado.

```
public void Empilhar (int elemento)
{
  if (! PilhaCheia( ))
   {
     topo = topo + 1;
     vetor[topo] = elemento;
   }
  else
   {
     System.out.println (“Pilha Cheia”);
   }
}
```

Desempilhar é um módulo função sem parâmetros da operação desempilhar, que verifica se a pilha não está vazia antes de desempilhar o elemento. Se a pilha estiver vazia, o elemento não pode ser desempilhado. Sendo assim, uma mensagem é enviada para o usuário e retorna-se o elemento 0. Se a pilha não estiver vazia, o elemento do topo é desempilhado e retornado.

```
// retorna o valor desempilhado

public int Desempilhar ()

{

  int desempilhado = 0;



  if (PilhaVazia())

  {

    System.out.println("Pilha Vazia");

  }

  else

  {

    desempilhado = vetor[topo];

    topo = topo – 1;
```

```
  }
  return desempilhado;
}
```

ElementoTopo é um módulo procedimento da operação elemento do topo que mostra ao usuário o elemento que está no topo da pilha.

```
public void ElementoTopo( )
{
  if (topo >= 0)
  {
    System.out.println("O elemento do topo é " + vetor[topo]);
  }
  else
  {
    System.out.println("Pilha Vazia");
  }
}
```

MostraPilha é um módulo procedimento da operação mostra pilha, que indica ao usuário todos os elementos da pilha.

```
public void MostrarPilha( )
{
  int i;

  for (i = topo ; i >= 0 ; i--)
   {
     System.out.println("Elemento " + vetor[i] + " posição " + i);
   }
}
```

APLICANDO AS PILHAS

Desenvolva um algoritmo que recebe do usuário cinco números inteiros numa pilha com capacidade para cinco números e que mostra esses números.

Algoritmo Pilha

```
início_algoritmo

    Definir

      // definição do tipo registro Pilha com os campos abaixo

      tipo Pilha = registro // o tipo registro chama-se Pilha

        tamanho numérico_inteiro; // campo tamanho

        topo ← –1 numérico_inteiro; // campo topo

        vetor [ tamanho] numérico_inteiro;  // campo vetor de capacidade

      fimregistro;



lógico PilhaVazia( )



lógico PilhaCheia( )



Empilhar (elemento numérico_inteiro)



numérico_inteiro Desempilhar ()



ElementoTopo( )
```

MostrarPilha( )

```
Algoritmo Exemplo1
 início_algoritmo
   Declarar
    intPilha ← novo Pilha(5);
    i numérico_inteiro;
    entrada numérico_inteiro;

  para i de 1 até 5 passo +1 faça
    escrever("Digite um valor inteiro");
    ler(entrada);
    intPilha.Empilhar(entrada);
  fimpara;
  intPilha.MostrarPilha( );
 fim_algoritmo.
```

APLICANDO PILHAS EM JAVA

```
import javax.swing.*;


class Exemplo1
 {
  public static void main (String arg [])
   {
    Pilha intPilha = new Pilha(5);
    int i;
    int entrada;

    for (i = 1 ; i <= 5 ; i++)
     {
      entrada = Integer.parseInt(JOptionPane.showInputDialog(
                        "Digite um valor inteiro"));
      intPilha.Empilhar(entrada);
     }
    intPilha.MostrarPilha( );
```

```
    System.exit(0);
  }
}

class Pilha
{
  int tamanho;
  int topo;
  int vetor [ ];

  Pilha (int tam)
  {
    topo = -1;
    tamanho = tam;
    vetor = new int[tam];
  }

  public boolean PilhaVazia( )
  {
```

```
  if (topo == -1)
  {
    return true;
  }
  else
  {
    return false;
  }
}

public boolean PilhaCheia( )
{
  if (topo >= tamanho-1)
  {
    return true;
  }
  else
  {
    return false;
```

```
  }
}

public void Empilhar (int elemento)
{
  if (! PilhaCheia( ))
   {
     topo = topo + 1;
     vetor[topo] = elemento;
   }
  else
   {
     System.out.println ("Pilha Cheia");
   }
}

public int Desempilhar ()
{
  int desempilhado = 0;
```

```
  if (PilhaVazia())
   {
     System.out.println(“Pilha Vazia”);
   }
  else
   {
     desempilhado = vetor[topo];
     topo = topo - 1;
   }
  return desempilhado;
}

public void ElementoTopo( )
 {
  if (topo >= 0)
   {
     System.out.println("O elemento do topo é " + vetor[topo]);
   }
```

```
  else
   {
     System.out.println("Pilha Vazia");
   }
}

public void MostrarPilha( )
{
  int i;

  for (i = topo ; i >= 0 ; i--)
   {
     System.out.println("Elemento " + vetor[i] + " posição " + i +
                        " da Pilha");
   }
}
```

Atividade extra

Indicação de leitura: ***Estrutura de Dados: algoritmos, análise da complexidade e implementações em Java e C/C++****, da Ana Fernanda Gomes Ascencio e Graziela Santos de Araújo, capítulo 4, sobre análise de complexidade da Pilha.*

Referência Bibliográfica

PUGA, S.; RISSETTI, G. ***Lógica de Programação e Estruturas de Dados, com aplicações em Java.*** *3ª. Ed. São Paulo: Editora Pearson, 2016.*

FORBELLONE, A.L.V.; EBERSPACHER, H.F. ***Lógica de Programação: a construção de algoritmos e estruturas de dados.*** *3ª Ed. São Paulo: Prentice Hall. 2005.*

Atividade Prática 07 – Entendendo as aplicações de Pilha

Título da Prática: Aplicações com Pilha em Java

Objetivos: Entender como utilizar o netbeans para desenvolver programas em Java para manipular e aplicar as operações com Pilha

Materiais, Métodos e Ferramentas: Computador, netbeans, Java.

Atividade Prática

O Algoritmo de manipulação das operações de Pilha para desenvolver um algoritmo que recebe do usuário cinco números inteiros numa pilha com capacidade para cinco números e mostra esses números, pode ser escrito como segue.

Desenvolva o programa em Java deste algoritmo no NetBeans.

.

```
Algoritmo Pilha
início_algoritmo
    Definir
      // definição do tipo registro Pilha com os campos abaixo
      tipo Pilha = registro // o tipo registro chama-se Pilha
        tamanho numérico_inteiro; // campo tamanho
        topo ← –1 numérico_inteiro; // campo topo
        vetor [ tamanho] numérico_inteiro;  // campo vetor de capacidade
      fimregistro;
```

lógico PilhaVazia( )

lógico PilhaCheia( )

Empilhar (elemento numérico_inteiro)

numérico_inteiro Desempilhar ()

ElementoTopo( )

MostrarPilha( )

Algoritmo Exemplo1

início_algoritmo

Declarar

intPilha ← novo Pilha(5);

i numérico_inteiro;

entrada numérico_inteiro;

para i de 1 até 5 passo +1 faça

escrever(“Digite um valor inteiro”);

ler(entrada);

intPilha.Empilhar(entrada);

fimpara;

intPilha.MostrarPilha( );

fim_algoritmo.

Gabarito Atividade Prática

import javax.swing.;

class Exemplo1

{

public static void main (String arg [])

{

```
    Pilha intPilha = new Pilha(5);

    int i;

    int entrada;



    for (i = 1 ; i <= 5 ; i++)

    {

      entrada = Integer.parseInt(JOptionPane.showInputDialog(

                          "Digite um valor inteiro"));

      intPilha.Empilhar(entrada);

    }

    intPilha.MostrarPilha( );

    System.exit(0);

  }

}



class Pilha

{

  int tamanho;

  int topo;
```

```
int vetor [ ];


Pilha (int tam)
{
  topo = -1;
  tamanho = tam;
  vetor = new int[tam];
}


public boolean PilhaVazia( )
 {
   if (topo == -1)
    {
      return true;
    }
   else
    {
      return false;
    }
```

```
}
```

```
public boolean PilhaCheia( )
{
  if (topo >= tamanho-1)
   {
     return true;
   }
  else
   {
     return false;
   }
}

public void Empilhar (int elemento)
{
  if (! PilhaCheia( ))
   {
     topo = topo + 1;
```

```
    vetor[topo] = elemento;
  }
  else
  {
    System.out.println ("Pilha Cheia");
  }
}

public int Desempilhar ()
{
  int desempilhado = 0;

  if (PilhaVazia())
  {
    System.out.println("Pilha Vazia");
  }
  else
  {
    desempilhado = vetor[topo];
```

```
    topo = topo - 1;
  }
  return desempilhado;
}

public void ElementoTopo( )
{
  if (topo >= 0)
  {
    System.out.println("O elemento do topo é " + vetor[topo]);
  }
  else
  {
    System.out.println("Pilha Vazia");
  }
}

public void MostrarPilha( )
{
```

```
    int i;

    for (i = topo ; i >= 0 ; i--)
    {
      System.out.println("Elemento " + vetor[i] + " posição " + i +
                         " da Pilha");
    }
  }
}
```

Ir para exercício