LINK DA AULA

https://meet.google.com/fvs-xkub-pgy

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Criando um array de 5 posicoes vazias

class PersonalArray:

SIZE = 5

insertPosition = 0

elements = [None] \* SIZE

# Função que serve para definir se o array está vazio ou não

def isEmpty(self):

return self.size() == 0

# Função que serve para retornar o número de elementos já armazenados no array

def size(self):

return self.insertPosition

# Função que serve para definir se precisamos de mais memória

def isMemoryFull(self):

return self.insertPosition == len(self.elements)

# Função para inserir um novo elemento na lista

def append(self, newElement):

if self.isMemoryFull():

self.updateMemory()

self.elements[self.insertPosition] = newElement

self.insertPosition += 1

# Função para aumentar a memória quando necessário

def updateMemory(self):

newArray = [None] \* (self.size() + self.SIZE)

for position in range(self.insertPosition): # Correção: iterar até self.insertPosition

newArray[position] = self.elements[position]

self.elements = newArray

# Função para limpar o array

def clear(self):

self.elements = [None] \* self.SIZE # Simplificação

self.insertPosition = 0

# Função para remover o último elemento

def remove(self):

if not self.isEmpty():

self.elements[self.insertPosition - 1] = None

self.insertPosition -= 1

# Função para remover um elemento em uma posição específica

def removePosition(self, position):

if position < 0 or position >= self.insertPosition:

print("Posição inválida!")

return ""

index = position

while index < self.insertPosition - 1:

self.elements[index] = self.elements[index + 1]

index += 1

self.elements[self.insertPosition - 1] = None

self.insertPosition -= 1

# Função para remover um elemento em uma posição específica

def insertAt(self, position, newElement):

if position < 0 or position > self.insertPosition:

print("Posição inválida!")

return

if self.isMemoryFull():

self.updateMemory()

index = self.insertPosition - 1

while index >= position:

self.elements[index + 1] = self.elements[index]

index -= 1

self.elements[position] = newElement

self.insertPosition += 1

# Função para retornar um elemento em uma posição específica

def elementAt(self, position):

if position < 0 or position >= self.insertPosition:

print("Posição inválida!")

return None

return self.elements[position]

array = PersonalArray()

array.append("0")

array.append("1")

array.append("2")

array.append("3")

array.append("4")

array.append("5")

array.removePosition(0)

array.removePosition(4)

array.removePosition(2)

array.insertAt(array.size(), 50)

for pos in range(0, array.size()):

print(array.elementAt(pos))

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Função para remover um elemento em uma posição específica

def removePosition(self, position):

if position < 0 or position >= self.insertPosition:

print("Posição inválida!")

return ""

index = position

while index < self.insertPosition - 1:

self.elements[index] = self.elements[index + 1]

index += 1

self.elements[self.insertPosition - 1] = None

self.insertPosition -= 1

LINK DA AULA

https://meet.google.com/brr-ekik-rur

#Criando um array de 5 posicoes vazias

class PersonalArray:

SIZE = 5

insertPosition = 0

elements = [None] \* SIZE

# Função que serve para definir se o array está vazio ou não

def isEmpty(self):

return self.size() == 0

# Função que serve para retornar o número de elementos já armazenados no array

def size(self):

return self.insertPosition

# Função que serve para definir se precisamos de mais memória

def isMemoryFull(self):

return self.insertPosition == len(self.elements)

# Função para inserir um novo elemento na lista

def append(self, newElement):

if self.isMemoryFull():

self.updateMemory()

self.elements[self.insertPosition] = newElement

self.insertPosition += 1

# Função para aumentar a memória quando necessário

def updateMemory(self):

newArray = [None] \* (self.size() + self.SIZE)

for position in range(self.insertPosition): # Correção: iterar até self.insertPosition

newArray[position] = self.elements[position]

self.elements = newArray

# Função para limpar o array

def clear(self):

self.elements = [None] \* self.SIZE # Simplificação

self.insertPosition = 0

# Função para remover o último elemento

def remove(self):

if not self.isEmpty():

self.elements[self.insertPosition - 1] = None

self.insertPosition -= 1

# Função para remover um elemento em uma posição específica

def removePosition(self, position):

if position < 0 or position >= self.insertPosition:

print("Posição inválida!")

return

# Desloca os elementos para preencher o espaço vazio

for i in range(position, self.insertPosition - 1):

self.elements[i] = self.elements[i + 1]

# Define o último elemento como None e decrementa a posição de inserção

self.elements[self.insertPosition - 1] = None

self.insertPosition -= 1

# Função para retornar um elemento em uma posição específica

def elementAt(self, position):

if position < 0 or position >= self.insertPosition:

print("Posição inválida!")

return None

return self.elements[position]

# Testando o código

array = PersonalArray()

# Inserindo elementos

array.append("fusca")

array.append("kombi")

array.append("kwid")

array.append("Ferrari")

array.append("Hilux")

array.append("fusca")

array.append("kombi")

array.append("kwid")

array.append("Ferrari")

array.append("Hilux")

array.append("fusca")

array.append("kombi")

array.append("kwid")

array.append("Ferrari")

array.append("Hilux")

# Acessando um elemento em uma posição específica

print("Elemento na posição 3:", array.elementAt(3)) # Ferrari

print("Elemento na posição 10:", array.elementAt(10)) # fusca

print("Elemento na posição 20:", array.elementAt(20)) # Posição inválida!

# Testando o método removePosition

print("\nAntes de remover:")

for i in range(array.size()):

print(array.elementAt(i), end=" ") # Imprime todos os elementos

array.removePosition(3) # Remove o elemento na posição 3 ("Ferrari")

print("\n\nDepois de remover na posição 3:")

for i in range(array.size()):

print(array.elementAt(i), end=" ") # Imprime todos os elementos

# Testando o método clear

print("\n\nAntes de limpar:")

print("Tamanho do array:", array.size()) # 14

array.clear()

print("\nDepois de limpar:")

print("Tamanho do array:", array.size()) # 0

def removePosition(self, position):

if position < 0 or position >= self.insertPosition:v

print("Posição invàlida")

return

for i in range(position, self.insertPosition - 1):

self.elements[i] = self.elements[i + 1]

self.elements[self.insertPosition - 1] = None

self.insertPosition -=1

**LIFO= last in, Fist out**

**FIFO= First in First out**

**remoção da fila sempre será a posição size -1, pois essa é onde se encontra o último elemento**

**Queue = fila**

**Pop= Desempilhar**

**push= empilhar**

**Codigo usado em java:**

//Biblioteca que contem a estrutura de lista ArrayList implementada na linguagem

import java.util.\*;

class PersonalQueue {

private ArrayList<String> list = new ArrayList<String>();

//Metodo que enfilera um novo elemento em nossa Fila (posicao, novaoElemento)

public void enqueue(String newElement) {

list.add(0, newElement);

}

//Metodo que retorna o elemento removido da Fila (no caso sempre a ultima posicao da lista interna)

public String dequeue() {

return list.remove( list.size() - 1 );

}

}

class PersonalStack {

public ArrayList<String> list = new ArrayList<String>();

public void push(String newElement){

list.add(0, newElement);

}

public String pop(){

return list.remove(0);

}

}

public class Main

{

public static void main(String[] args) {

System.out.println("\nmetodo fila");

//Cria uma instancia do objeto da fila que implementamos

PersonalQueue queue = new PersonalQueue();

//Enfileirando elementos na estutura de dados

queue.enqueue("1");

queue.enqueue("2");

queue.enqueue("3");

queue.enqueue("4");

queue.enqueue("5");

//Desenfileira os elementos inseridos na estrutura de dados e imprime

System.out.println( queue.dequeue() );

System.out.println( queue.dequeue() );

System.out.println( queue.dequeue() );

System.out.println( queue.dequeue() );

System.out.println( queue.dequeue() );

System.out.println("\nmetodo pilha");

//criando a instancia do objeto pilha

PersonalStack stack = new PersonalStack();

//empilhando os objeto

stack.push("1");

stack.push("2");

stack.push("3");

stack.push("4");

stack.push("5");

//desempilhando os objeto

System.out.println( stack.pop() );

System.out.println( stack.pop() );

System.out.println( stack.pop() );

System.out.println( stack.pop() );

System.out.println( stack.pop() );

}

}

**Desafio:**

**por meio de método pilha faça o cálculo de um número dado pelo usuário ou passado descobrir a tradução do número em binário assim que chegar no resultado 1**

**codigo final**

**import java.util.\*;**

**class PersonalQueue {**

**private ArrayList<String> list = new ArrayList<String>();**

**public void enqueue(String newElement) {**

**list.add(0, newElement);**

**}**

**public String dequeue() {**

**if (!list.isEmpty()) {**

**return list.remove(list.size() - 1);**

**}**

**return null; // Retorna null se a fila estiver vazia**

**}**

**}**

**class PersonalStack {**

**private ArrayList<String> list = new ArrayList<String>();**

**public void push(String newElement) {**

**list.add(0, newElement);**

**}**

**public String pop() {**

**if (!list.isEmpty()) {**

**return list.remove(0);**

**}**

**return null; // Retorna null se a pilha estiver vazia**

**}**

**}**

**class PersonalBinario {**

**public static void convBinario(int numero) {**

**if (numero > 0) {**

**convBinario(numero / 2);**

**System.out.print(numero % 2);**

**}**

**}**

**}**

**public class Main {**

**public static void main(String[] args) {**

**System.out.println("\nMétodo Fila:");**

**PersonalQueue queue = new PersonalQueue();**

**queue.enqueue("1");**

**queue.enqueue("2");**

**queue.enqueue("3");**

**queue.enqueue("4");**

**queue.enqueue("5");**

**System.out.println(queue.dequeue());**

**System.out.println(queue.dequeue());**

**System.out.println(queue.dequeue());**

**System.out.println(queue.dequeue());**

**System.out.println(queue.dequeue());**

**System.out.println("\nMétodo Pilha:");**

**PersonalStack stack = new PersonalStack();**

**stack.push("1");**

**stack.push("2");**

**stack.push("3");**

**stack.push("4");**

**stack.push("5");**

**System.out.println(stack.pop());**

**System.out.println(stack.pop());**

**System.out.println(stack.pop());**

**System.out.println(stack.pop());**

**System.out.println(stack.pop());**

**System.out.println("\nConversão para Binário:");**

**int numero = 7777;**

**PersonalBinario.convBinario(numero);**

**System.out.println(); // Quebra de linha após a impressão**

**}**

**}**

**===========================================**

**import java.util.\*;**

**class PersonalQueue {**

**private ArrayList<String> list = new ArrayList<String>();**

**public void enqueue(String newElement) {**

**list.add(0, newElement);**

**}**

**public String dequeue() {**

**if (!list.isEmpty()) {**

**return list.remove(list.size() - 1);**

**}**

**return null; // Retorna null se a fila estiver vazia**

**}**

**}**

**class PersonalStack {**

**private ArrayList<String> list = new ArrayList<String>();**

**public void push(String newElement) {**

**list.add(0, newElement);**

**}**

**public String pop() {**

**if (!list.isEmpty()) {**

**return list.remove(0);**

**}**

**return null; // Retorna null se a pilha estiver vazia**

**}**

**}**

**class PersonalBinario {**

**public static void convBinario(int numero) {**

**if (numero > 0) {**

**convBinario(numero / 2);**

**System.out.print(numero % 2);**

**}**

**}**

**}**

**class Prompt {**

**public static String readLine() {**

**Scanner reader = new Scanner(System.in);**

**return reader.next();**

**}**

**}**

**public class Main {**

**public static void main(String[] args) {**

**System.out.println("\nMétodo Fila:");**

**PersonalQueue queue = new PersonalQueue();**

**queue.enqueue("1");**

**queue.enqueue("2");**

**queue.enqueue("3");**

**queue.enqueue("4");**

**queue.enqueue("5");**

**System.out.println(queue.dequeue());**

**System.out.println(queue.dequeue());**

**System.out.println(queue.dequeue());**

**System.out.println(queue.dequeue());**

**System.out.println(queue.dequeue());**

**System.out.println("\nMétodo Pilha:");**

**PersonalStack stack = new PersonalStack();**

**stack.push("1");**

**stack.push("2");**

**stack.push("3");**

**stack.push("4");**

**stack.push("5");**

**System.out.println(stack.pop());**

**System.out.println(stack.pop());**

**System.out.println(stack.pop());**

**System.out.println(stack.pop());**

**System.out.println(stack.pop());**

**System.out.println("\nConversão para Binário:");**

**int numero = 777777;**

**PersonalBinario.convBinario(numero);**

**System.out.println("Digite algo:"); // Quebra de linha após a impressão**

**Prompt.readLine();**

**}**

**}**