package org.example;  
  
import java.util.Stack;  
  
public class Pilha {  
  
 public static void main(String[] args) {  
  
  
 Stack<Integer> stack = new Stack<>();  
 stack.push(0);  
 stack.push(1);  
 stack.push(2);  
 stack.push(3);  
 stack.push(4);  
  
 System.*out*.println("Mostrando lista completa" + stack);  
 System.*out*.println("Retirando último elemento com pop, no caso o último elemento a entrar é o primeiro a sair, n° " + stack.pop());  
 System.*out*.println("Olha o número que está no topo da lista e o retorna sem alterar a lista");  
 System.*out*.println(stack.peek());  
 System.*out*.println("Retorna true ou false, se a lista está vazia ou não");  
 System.*out*.println(stack.empty());  
 System.*out*.println("Retorna o tamanho da pilha");  
 System.*out*.println(stack.size());  
 System.*out*.println("search: Retorna a posição do elemento n caso pesquisado o n° 1 e sua posição atual é 3");  
 System.*out*.println(stack.search(1));  
  
  
 }  
  
}

calcule a complexidade assintótica de tempo e de espaço das estruturas de dados que você criou. A complexidade de tempo deverá ser calculada para os métodos definidos abaixo:

* Pilha: métodos push e pop.

Método push: O(1) porque a adição não necessita que percorra toda a lista.

Método pop: O(1) pois o último elemento é retirado da lista sem percorrer ela toda.

package org.example;  
  
import java.util.LinkedList;  
import java.util.Queue;  
  
public class FIFO {  
  
 public static void main(String[] args) {  
  
 Queue<Integer> fifo = new LinkedList<>();  
 fifo.offer(2);  
 fifo.offer(3);  
 fifo.offer(4);  
 fifo.offer(5);  
 fifo.offer(6);  
  
 System.*out*.println("Itens na lista " + fifo);  
 System.*out*.println("Removendo item da lista FIFO, primeiro que entra é o primeiro que sai n° " + fifo.remove());  
 System.*out*.println(fifo);  
 System.*out*.println("Olha o primeiro elemento da lista " + fifo.peek());  
 System.*out*.println("Olha o primeiro elemento da lista igual o peek " + fifo.poll());  
 System.*out*.println("Elemento a frente da fila " + fifo.element());  
 }  
  
}

offer: O(1), adiciona o elemento no final da fila.

Remove: O(1) remove o primeiro elemento da fila.

Peek e pool e element: O(1)verifica o primeiro elemento da fila.