

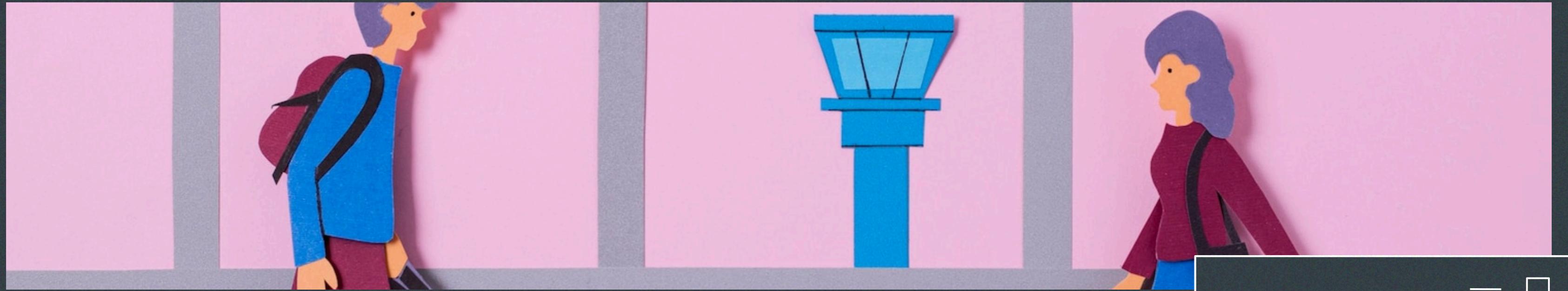
- □ ×

PriorityQueue JAVA

11

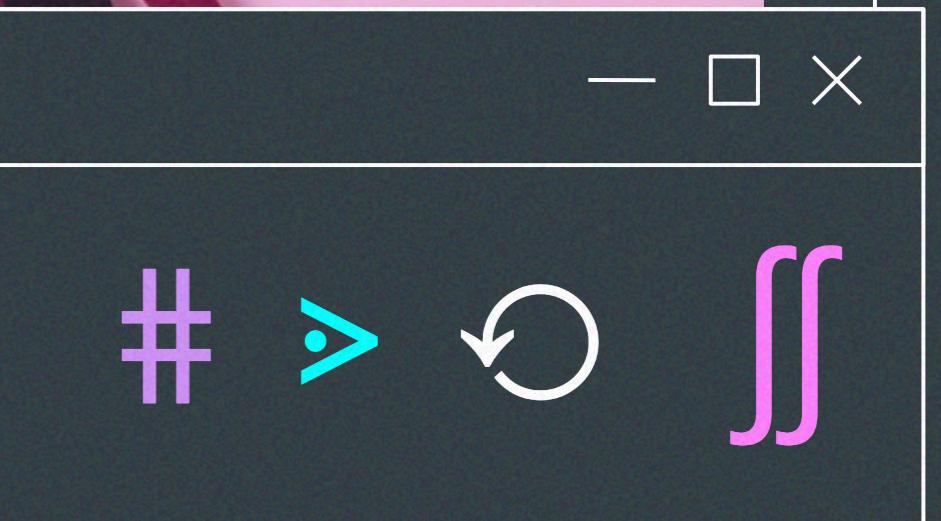
- □ ×

> ◎ ≡



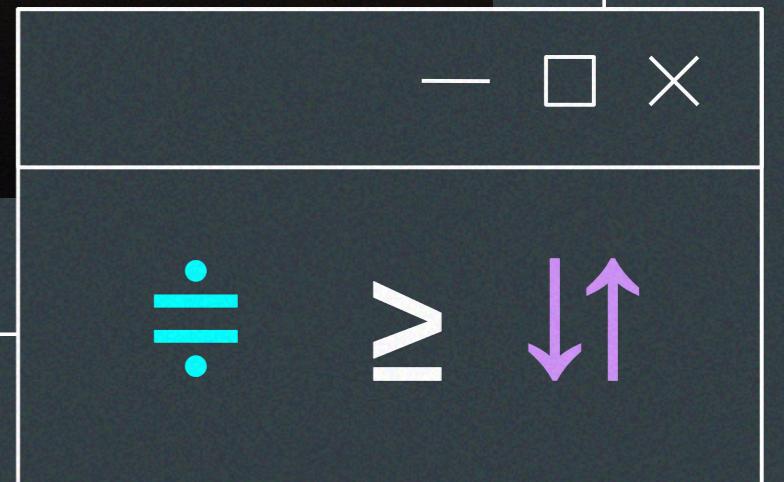
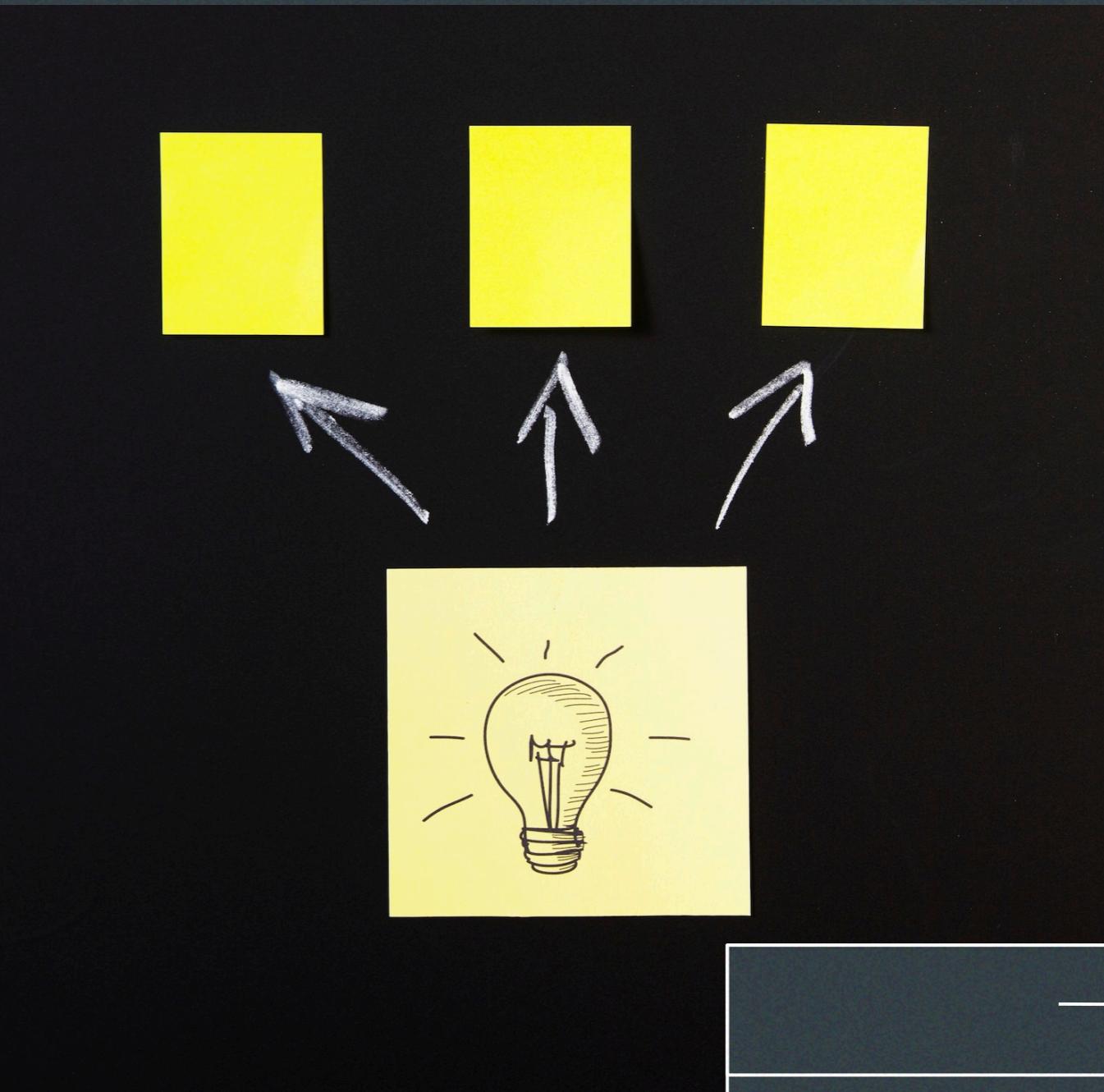
Introdução

O **PriorityQueue** é uma estrutura de dados em **Java** que representa uma fila de prioridade, onde os elementos são processados de acordo com sua **prioridade**. É uma implementação da interface *Queue* e é comumente utilizada em algoritmos de *busca*, *agendamento* e *gerenciamento de eventos*.



Funcionalidades

O **PriorityQueue** em Java oferece funcionalidades para *inserir* e *remover* elementos de acordo com sua **prioridade**. Além disso, fornece métodos para *acessar* o elemento de maior prioridade e verificar se a fila está *vazia*. Essas funcionalidades são essenciais para a implementação de algoritmos eficientes.



IMPORT

```
import java.util.PriorityQueue;
```

INSTANCIAR

```
PriorityQueue filaPrioridade =  
new PriorityQueue<>();
```

ADICIONAR

```
filaPrioridade.offer(10);
```

REMOVER

```
filaPrioridade.poll();
```

MAIOR PRIORIDADE

```
filaPrioridade.peek();
```

VERIFICAR VAZIO

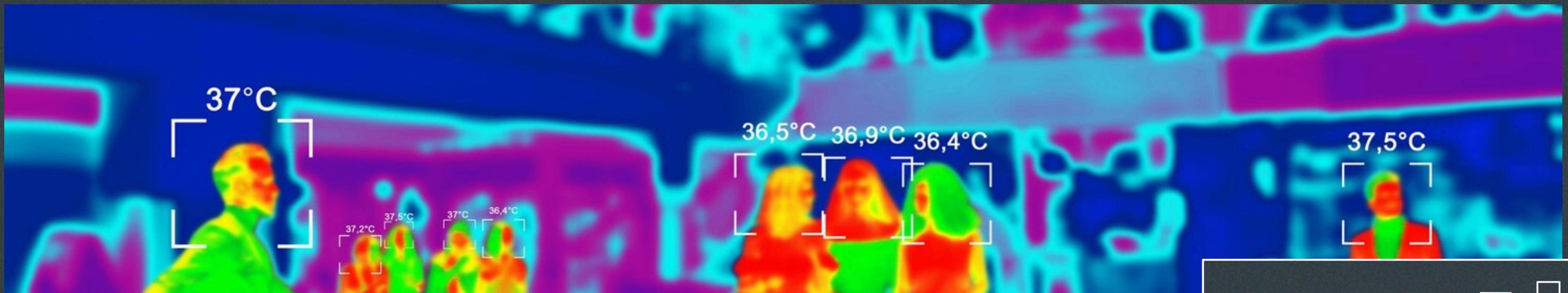
```
filaPrioridade.isEmpty();
```

Aplicações

O **PriorityQueue** é amplamente utilizado em **Java** para implementar algoritmos de *busca*, como o algoritmo de **Dijkstra** para encontrar o caminho mais curto em um grafo ponderado. Além disso, é empregado em *sistemas de agendamento* para gerenciar tarefas com diferentes níveis de prioridade.



÷ ≥ ↓↑



Utilizações

O **PriorityQueue** em Java é utilizado em *sistemas de gerenciamento de eventos* para processar eventos com base em sua **urgência**. Também é aplicado em *sistemas de simulação* para controlar a ordem de execução de eventos simulados, como chegada de clientes em um sistema de atendimento.



Complexidade

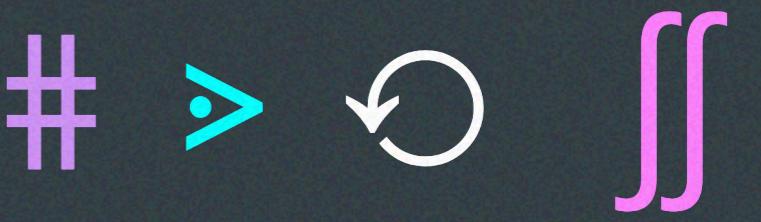


A **complexidade** das operações em um **PriorityQueue** depende da implementação utilizada. No caso da implementação padrão em **Java**, a inserção e remoção de elementos têm complexidade $O(\log n)$, enquanto o acesso ao elemento de maior prioridade tem complexidade $O(1)$.



Desempenho

O **PriorityQueue** em Java oferece um bom *desempenho* em cenários onde é necessário processar elementos de acordo com sua **prioridade**. Ao utilizar a implementação adequada e considerar a **complexidade** das operações, é possível alcançar eficiência em algoritmos e aplicações.





Exemplos de Código

A utilização do **PriorityQueue** em Java pode ser ilustrada com exemplos de código que demonstram como inserir, remover e acessar elementos com diferentes prioridades. Esses exemplos ajudam a compreender a aplicação prática dessa estrutura de dados.



```
public class Main {  
    public static void main(String[] args) {  
  
        int[] arrayDeInteiros = {10, 7, 9, 2, -3};  
  
        PriorityQueue<Integer> filaPrioridade = new PriorityQueue<>();  
  
        for (int i = 0; i < arrayDeInteiros.length; i++) {  
            filaPrioridade.add(arrayDeInteiros[i]);  
        }  
        while (!filaPrioridade.isEmpty()) {  
            System.out.println(filaPrioridade.poll());  
        }  
    }  
}
```

```
-3  
2  
7  
9  
10
```

```
public class Main {  
    public static void main(String[] args) {  
  
        int[] arrayDeInteiros = {10, 7, 9, 2, -3};  
  
        PriorityQueue<Integer> filaPrioridade = new PriorityQueue<>(Comparator.reverseOrder());  
  
        for (int i = 0; i < arrayDeInteiros.length; i++) {  
            filaPrioridade.add(arrayDeInteiros[i]);  
        }  
        while (!filaPrioridade.isEmpty()) {  
            System.out.println(filaPrioridade.poll());  
        }  
    }  
}
```

```
10  
9  
7  
2  
-3
```

```
public class Main {  
    public static void main(String[] args) {  
  
        String[] nomes = {"Alice", "Jubiscleuso", "Karina", "Chico", "Jaiminho"};  
  
        // Cria uma PriorityQueue de strings e adiciona os nomes  
        PriorityQueue<String> filaPrioridade = new PriorityQueue<>();  
        for (String nome : nomes) {  
            filaPrioridade.offer(nome);  
        }  
        // Imprime os nomes ordenados alfabeticamente  
        System.out.println("Nomes ordenados:");  
        while (!filaPrioridade.isEmpty()) {  
            System.out.println(filaPrioridade.poll());  
        }  
    }  
}
```

```
Nomes ordenados:  
Alice  
Chico  
Jaiminho  
Jubiscleuso  
Karina
```

```
import java.util.PriorityQueue;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        PriorityQueue<Integer> fila = new PriorityQueue<>();

        // Adicionando elementos à fila usando offer()
        fila.offer(10);
        fila.offer(20);
        fila.offer(30);
        fila.offer(40);
        fila.offer(50);

        // Removendo e imprimindo elementos da fila usando poll() e peek()
        while (!fila.isEmpty()) {
            System.out.println("Maior prioridade: " + fila.peek());
            System.out.println("Elemento removido: " + fila.poll());
            System.out.println("-----");
        }
    }
}

Maior prioridade: 10
Elemento removido: 10
-----
Maior prioridade: 20
Elemento removido: 20
-----
Maior prioridade: 30
Elemento removido: 30
-----
Maior prioridade: 40
Elemento removido: 40
-----
Maior prioridade: 50
Elemento removido: 50
-----
```

— □ ×

Conclusão

O **PriorityQueue** em Java é uma ferramenta poderosa para lidar com a **ordenação** e **processamento** de elementos com base em sua **prioridade**. Suas funcionalidades, aplicações e desempenho o tornam essencial para o desenvolvimento de algoritmos eficientes e aplicações que requerem gerenciamento de prioridades.

— □ ×

÷ ≥ ↓↑

- □ ×



Obrigado!

Você tem alguma pergunta?

- □ ×

=> ●

- □ ×

÷ ▶

