PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA E DE ARTES
ESTRUTURA DE DADOS ORIENTADA A OBJETOS
ADS1232
PROF. MSC. ANIBAL SANTOS JUKEMURA

PROGRAMAÇÃO ORIENTADA A OBJETOS [JAVA]

Agenda

- Filas
- Conceitos
- Operações
- Exemplos
- Exercícios

Filas

PUC GOIÁS

- No dia a dia, estamos acostumados com as filas em diversos lugares: nos bancos, nos mercados, nos hospitais, nos cinemas entre outros.
- As filas são importantes pois elas determinam a ordem de atendimento das pessoas.
- As pessoas são atendidas conforme a posição delas na fila.
- O próximo a ser atendido é o primeiro da fila.
- Quando o primeiro da fila é chamado para ser atendido a fila "anda", ou seja, o segundo passa a ser o primeiro, o terceiro passa a ser o segundo e assim por diante até a última pessoa.
- Normalmente, para entrar em uma fila, uma pessoa deve se colocar na última posição, ou seja, no fim da fila. Desta forma, quem chega primeiro tem prioridade.

PUC GOIÁS

Fila (*Queue*)

Uma coleção utilizada para manter uma "fila" de elementos. Existe uma ordem linear para as filas que é a "ordem de chegada". As filas devem ser utilizadas quando os itens deverão ser processados de acordo com a ordem "PRIMEIRO-QUE-CHEGA, PRIMEIRO-ATENDIDO". Por esta razão as filas são chamadas de Listas **FIFO**, termo formado a partir de "*First-In, First-Out*".

PUC GOIÁS

Fila (*Queue*)

- Fila é uma estrutura de dados linear “especial”, que permite apenas a eliminação de elementos no início e no fim realiza-se a operação de inserção.

- Uma das formas mais simples de se implementar uma fila em java é instanciá-la como uma lista simples encadeada, ou LinkedList.

- A classe LinkedList implementa a interface de fila, para que possamos determinar qual tipo de fila podemos usar.

Filas

Queue

- Declaração:
  - **import** java.util.Queue;
  - **import** java.util.LinkedList;
  - **Queue**<TIPO> fila = new **LinkedList**();

- Principais métodos:

- void **offer**(**Object** element): Insere um elemento no final da fila sem que ocorra uma violação nas restrições de capacidade da fila. A diferença com o método add ocorre quando a fila possui tais restrições, sendo que o método add somente irá inserir o elemento, gerando uma exceção, caso uma das restrições seja violada.

- boolean **add**(**Object** element): Adiciona o elemento especificado no final da fila.

- boolean **contains**(**Object** element): Retorna verdadeiro se a fila contém o elemento especificado e falso caso contrário.

- Object **element**( ): Retorna primeiro elemento da fila sem removê-lo. Executa um throw exception do tipo **NoSuchElementException** se a fila estiver vazia.

Filas

Queue

- Principais métodos:
- Object **peek**( ): Retorna primeiro elemento da fila sem removê-lo. Retorna **null** se a fila estiver vazia.
- boolean **isEmpty**( ): Retorna true se a fila estiver vazia. Retorna false, caso contrário.
- Object **remove**( ): Remove o primeiro elemento da fila. Executa um throw exception do tipo **NoSuchElementException** se a fila estiver vazia.
- Object **poll**( ): Remove o primeiro elemento da fila. Retorna **null** se a fila estiver vazia.
- int **size**(): Retorna o número de elementos da fila.
- Iterator<E> **iterator**(): retorna um iterator para controle da coleção.
- void **clear**(): Remove todos os elementos da fila.

Filas

Queue: exemplo 1 – Fila de Inteiros

Neste exemplo em sala realizaremos as seguintes operações:

1 – Criar uma fila de Inteiros:

```
Queue<Integer> fila = new LinkedList();
```

2 - Usar o método add() ou offer() para gravar números digitados pelo usuário:

```
fila.add(ler.nextInt());
fila.offer(ler.nextInt());
```

3- Vamos remover um elemento da fila:

```
fila.remove();
fila.poll();
```

5- Mostar os "n" elementos da fila (usando for-each)

```
for (int num: fila)
```

6- mostrando os "n" elementos da fila (com iterator)

```
Iterator<Integer> iterator = fila.iterator();
iterator.hasNext()
iterator.next()
```

Vejamos um exemplo em Sala

```java
import java.util.Queue;
import java.util.LinkedList;
import java.util.NoSuchElementException;
import java.util.Iterator;
import java.util.Random;
import java.util.Collections;

public class Exemplo01 {

    public static void main(String[] args) {
        Queue<Integer> fila = new LinkedList<Integer>();
        int i, quantidade = 10;
        Random fator = new Random();

        for(i = 0; i < quantidade; i++)
        {
            fila.add(fator.nextInt(20) + 1);
        }
        System.out.println ("Fila:");
        System.out.println(fila);
        System.out.println("Primeiro elemento da fila (peek): " + fila.peek());
        System.out.println("Primeiro elemento da fila (element): " + fila.element());
        System.out.println ("Quantidade de elementos na fila: " + fila.size());
        System.out.println("Removendo um elemento na fila (poll): " + fila.poll());
        System.out.println(fila);
        System.out.println("Removendo um elemento na fila (remove): " + fila.remove());
        System.out.println(fila);
        fila.offer(100);
        System.out.println(fila);
        fila.clear();
```


```
31          if (fila.isEmpty())
32              System.out.println("Fila vazia");
33          fila.peek();
34          try
35          {
36              fila.element();
37          }
38          catch (NoSuchElementException e)
39          {
40              System.out.println("Fila Vazia");
41          }
42          for (i = 0; i < quantidade; i++)
43          {
44              fila.offer(fator.nextInt(quantidade));
45          }
46          Iterator<Integer> ponteiro = fila.iterator();
47          System.out.print ("[");
48          while (ponteiro.hasNext())
49          {
50              System.out.print(ponteiro.next());
51              if (ponteiro.hasNext()!=false)
52                  System.out.print (", ");
53          }
54          System.out.println("]");
55          System.out.print ("[ ");
56          for (int num: fila)
57          {
58              System.out.print (num + " ");
59          }
60          System.out.println("]");
61      }
62
63  }
```

Filas

Exercícios:

Exercício 01 – Faça um programa que forneça uma fila com uma sequência aleatória de números inteiros. Imprima a fila. Em seguida, crie duas novas filas nomeadas impar e par. Leia a primeira fila e preencha as duas novas filas conforme a característica de cada uma. Elimine um a um cada elemento da fila original, na medida em que cada fila esteja sendo criada. Imprima as duas novas filas (use os métodos iterator e for (int num: <fila>)) e faça um teste de fila vazia na fila original, emitindo uma mensagem ao usuário.

Exemplo:

Fila : [51, 49, 70, 23, 90, 38, 71, 20]
Par : [70, 90, 38, 20]
Impar: [51, 49, 23, 71]

Fila Original Vazia!

Filas

Referência Bibliográfica Principal

• DEVMEDIA. Disponível em https://www.devmedia.com.br . Acessado em Fevereiro de 2018.

• GUJ. Disponível em http://www.guj.com.br . Acessado em Fevereiro de 2018.