UNIFACISA CENTRO UNIVERSITÁRIO

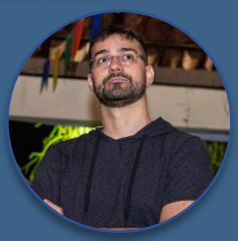
SISTEMAS DE INFORMAÇÃO

Aprendizagem baseada em competências.

POO Avançado

JAVA

Listas – Filas e Pilhas



Prof. Diego Braga

Conteúdo:

01 Conceito

- Uma filha é um conjunto de itens a partir do qual podem-se eliminar itens obedecendo uma ordem.
- Existem muitos exemplos de fila no mundo real:
  - Uma fila de banco;
  - No ponto de ônibus;
  - Um grupo de carros aguardando sua vez no pedágio;
- Em programação, uma fila é uma estrutura de dados linear ou uma coleção em Java que armazena elementos em uma ordem FIFO (primeiro a entrar, primeiro a sair).
- Filas são coleções ordenadas de objetos e são casos especiais de listas.

Nas listas, quando precisávamos criar um novo elemento, poderíamos inseri-lo ou remove-lo de qualquer posição da lista, exemplo:

- Na primeira posição;
- Na ultima posição
- Em qualquer parte no meio da lista.

- Em uma fila existe uma regra básica a ser seguida.
  - Primeiro a chegar é o primeiro a sair
  - Chamamos isso de FIFO – First in, First Out
- Um novo elemento da fila somente pode ser inserido na ultima posição (fim da fila).
- Um elemento só pode ser removido da primeira posição (inicio da fila).

02 Filas na computação

- Filas de impressão:
  - Impressoras tem uma fila, caso vários documentos sejam impressos, por um ou mais usuários, os primeiros documentos impressos serão de quem enviar primeiro;
- Filas de processos:
  - Vários programas podem estar sendo executados pelo sistema operacional. O mesmo tem uma fila que indica a ordem de qual será executado primeiro;
- Filas de tarefas:
  - Um programa pode ter um conjunto de dados para processar. Estes dados podem estar dispostos em uma fila, onde o que foi inserido primeiro, será atendido primeiro.

03 Criando/Inserindo/Iterando

- Como Queue é uma interface, usamos uma classe LinkedList que implementa a interface Queue para criar um objeto de fila.
  - *Queue<Tipo> str_queue = new LinkedList ();*
- Agora que o objeto de fila foi criado, podemos inicializar o objeto de fila fornecendo os valores a ele por meio do método add, conforme mostrado abaixo.
  - *str_queue.add("one");*
  - *str_queue.add("two");*
- Podemos percorrer os elementos da fila usando o loop forEach ou um iterador. O programa fornecido a seguir implementa ambas as abordagens para percorrer a fila.

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


    Queue<String> LL_queue = new LinkedList<String>();
    LL_queue.add("One");
    LL_queue.add("Two");
    LL_queue.add("Three");
    LL_queue.add("four");

    System.out.println("The Queue elements using 'for' loop:");

    for(String obj : LL_queue) {
        System.out.print(obj + " ");
    }

    System.out.println();
    System.out.println("The Queue elements through iterator:");
    Iterator<String> iterator = LL_queue.iterator();
    while(iterator.hasNext()){
        String element = (String) iterator.next();
        System.out.print(element + " ");
    }

}
```

Continua...

04 Mais funções em filas

```java
Queue<Integer> ql = new LinkedList<Integer>();
ql.add(10);
ql.add(20);
ql.add(30);
ql.add(40);
ql.add(50);

//remove() - Remove o primeiro elemento da fila ql
System.out.println("Element removed from the queue: " + ql.remove());

//element() - retorna o primeiro elemento da fila e caso esteja vazia, retona exception
System.out.println("Head of the queue: " + ql.element());

//poll() - remove e retorna o primeiro elemento
System.out.println("Poll():Returned Head of the queue: " + ql.poll());

//peek() - retorna o primeiro elemento da fila
System.out.println("peek():Head of the queue: " + ql.peek());

//Imprime toda a fila
System.out.println("Final Queue: " + ql);
```

05 Variações de Filas:

- Fila de Prioridades:
  - Cada item tem uma prioridade. Elementos mais prioritários podem ser atendidos antes, mesmo não estando no início da fila;
  - *Queue str_pqueue = new PriorityQueue ();*
- Fila Circular:
  - Neste tipo de fila os elementos nem sempre são removidos ao serem atendidos, mas voltam ao fim da fila para serem atendidos novamente mais tarde.
  - *Queue int_queue = new ArrayDeque ();*

- Para maiores duvidas, consultar a documentação no link abaixo:
  - https://docs.oracle.com/javase/8/docs/api/java/util/Queue.html

06 Exercícios:

- Exercício #1:
- Implemente um programa que contemple uma fila de contatos para um Call center;
- As opções do programa devem ser:
  - Inserir Contato:
    - Deve solicitar ao usuário os dados e incluir o contato na fila;
  - Próximo Contato:
    - Deverá pegar o Contato do Inicio da Fila, removê-lo e mostrar os seus dados na tela para o usuário efetuar o contato com o cliente..

- Exercício #2:
- É levado em consideração que os pacientes são registrados na fila assim que chegam a sala de espera e ficam aguardando o chamado do médico, que obedece a ordem de chegada. O programa deve permitir:
  - Inserir um paciente na fila de espera;
  - Chamar o paciente para ser atendido;
  - Verificar se a fila está cheia ou vazia;
  - Verificar o próximo paciente a ser atendido;
  - Informar quantos pacientes existem na fila de espera

Continua...

- Exercício em grupo #1:
- Em grupo de até 4 pessoas, pesquisar o funcionamento e a implementação de uma fila com prioridades. *PriorityQueue ()*
- Após o entendimento. Mostrar o funcionamento aplicando o conhecimento no exercício abaixo:
- Escreva um programa que simule a distribuição de senhas de atendimento a um grupo de pessoas. Cada pessoa pode receber uma senha prioritária ou uma senha normal. O programa deve obedecer os seguintes critérios.
  - Existe apenas 1 atendente;
  - Uma pessoa com senha normal deve ser atendida a cada 3 pessoas com senha prioritária;
  - Não havendo prioridades, as pessoas com senha normal podem ser atendidas.

- Dicas:
- Usem a documentação do Java
  - https://docs.oracle.com/javase/8/docs/api/java/util/PriorityQueue.html
- Pensem bem na questão.
- Pesquisem em todos os lugares da net.
- Tentem realmente fazer.

01 Pilhas

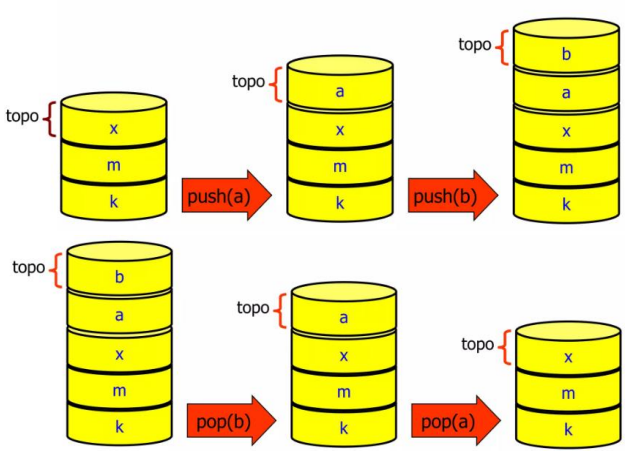
- Uma pilha é como uma pilha de livros, em que o primeiro livro que foi inserido na pilha, normalmente é o último que sai dela, enquanto o último adicionado é o primeiro a ser retirado, como vemos.
- A pilha é considerada uma estrutura de dados simples, sendo fácil de implementar.
- Em uma análise simples, poderia ser utilizada, por exemplo, em um carregamento de um caminhão, pois se o caminhão tiver 4 entregas, a última entrega colocada dentro do caminhão deve ser a primeira a sair, caso contrário, pode dar mais trabalho para descarregar.
- A pilha é uma das estruturas de dados e trabalha com o formato LIFO:
  - O último a entrar é o primeiro a sair
  - “Last In, First Out”, em inglês.

02 Programando

- Para criar uma pilha é necessário utilizar a classe Stack().

```
Stack<String> pilha = new Stack<String>();
```

- Existem duas operações básicas que devem ser implementadas em uma estrutura de pilha:
  - Operação para empilhar (push()) um novo elemento, inserindo-o no topo.
  - Operação para desempilhar (pop()) um elemento removendo-o do topo.



- Após instanciado a classe, podemos inserir e retirar objetos da pilha utilizando os comandos:

```
pilha.push("Primeiro");
pilha.push("Segundo");
pilha.push("Terceiro");

pilha.pop();
```

- No exemplo acima, ao utilizar o comando pop(), qual elemento será retirado da pilha?

- Após algum tempo a classe Stack entrou em desuso, dando preferencia a Interface Deque, através da classe ArrayDeque().

03 Exercitando

- Exercício #1:
- Crie um programa que gerencie uma PILHA de TAREFAS a serem cumpridas. As tarefas são Strings que descrevem uma ação a ser executada.
- O usuário deverá ter duas opções:
  - Inserir tarefa na pilha
  - Obter a próxima tarefa da pilha.

- Exercício #2:
- Em grupo pesquise sobre a Interface Deque e a classe ArrayDeque().
- Refaça o exercício acima utilizando a interface ao invés da classe Stack.

- Dica:
- https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html