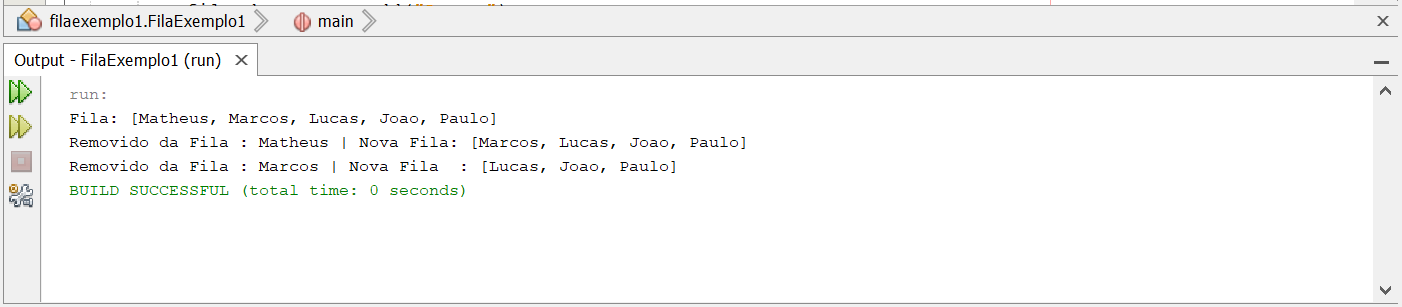
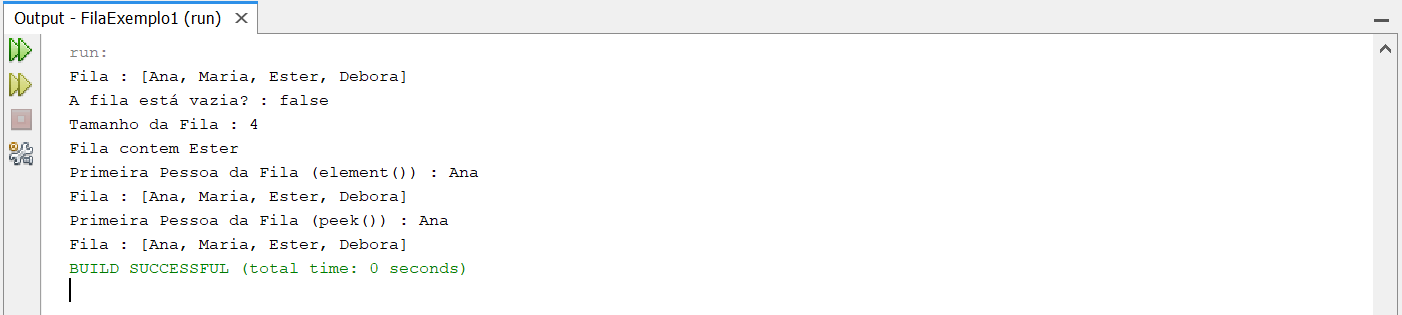
**TESTE DE MESA**

Exemplo 1 –



O código cria uma fila de espera usando LinkedList e demonstra as operações de enfileirar e desenfileirar. Primeiro, adiciona cinco nomes à fila. Em seguida, remove o primeiro elemento (Matheus), e depois o próximo (Marcos). A cada remoção, a fila se ajusta automaticamente, mantendo a ordem FIFO (First In, First Out). No final, a fila contém apenas Lucas, Joao e Paulo.

Exemplo 2 –

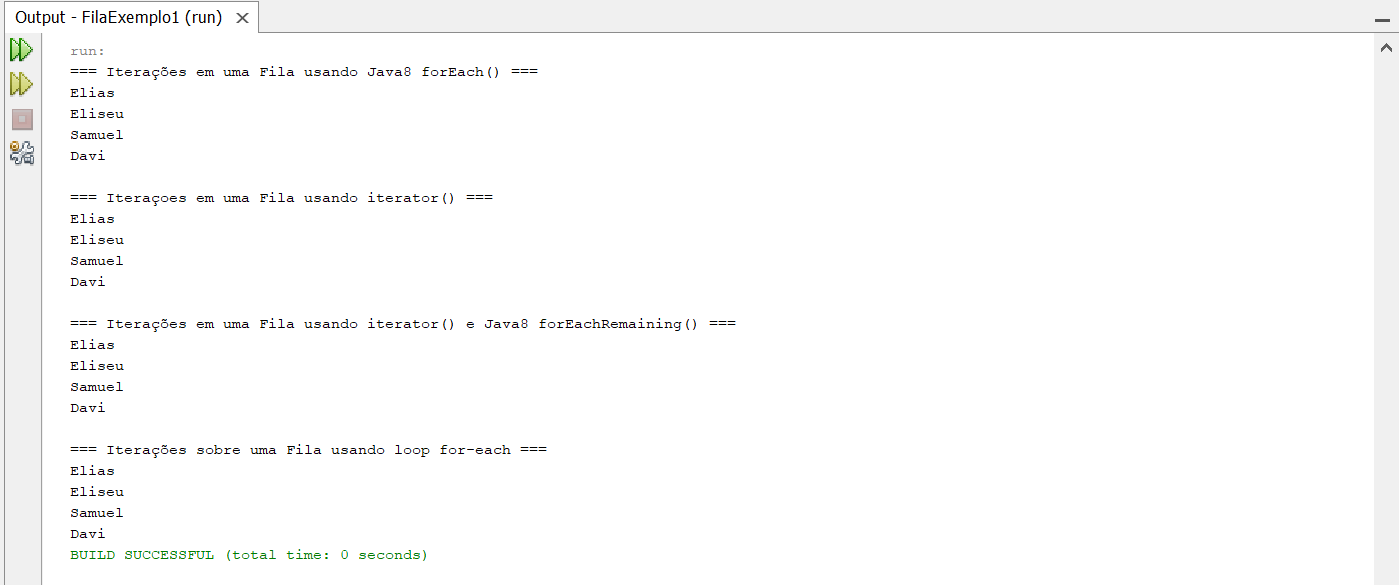


Este código demonstra o uso da estrutura de fila (Queue) em Java usando LinkedList. Ele adiciona quatro nomes à fila e realiza diversas operações:

* Verifica se a fila está vazia.
* Obtém o tamanho da fila.
* Checa se um determinado nome (Ester) está presente.
* Recupera o primeiro elemento da fila sem removê-lo (element() e peek()).

A diferença entre element() e peek() é que element() lança uma exceção se a fila estiver vazia, enquanto peek() apenas retorna null. Ao final, a fila permanece inalterada.

Exemplo 3 –

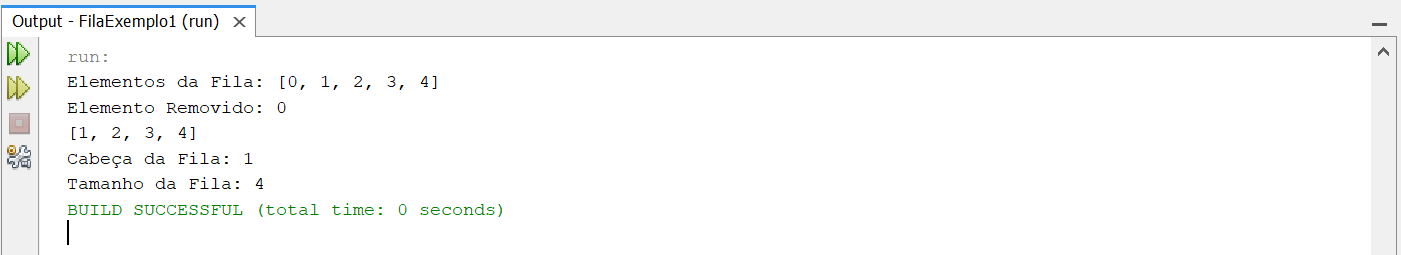


Este código demonstra diferentes maneiras de percorrer (iterar) uma fila em Java. Ele cria uma fila de espera e adiciona quatro nomes: Elias, Eliseu, Samuel e Davi. Em seguida, utiliza quatro métodos diferentes para iterar sobre os elementos da fila:

* **forEach() do Java 8**: Usa uma expressão lambda para percorrer e imprimir os elementos da fila.
* **iterator() tradicional**: Utiliza um Iterator manualmente com while e hasNext().
* **forEachRemaining() do Java 8**: Usa o Iterator de forma simplificada.
* **Loop for-each**: Percorre a fila diretamente de maneira mais legível.

Cada método exibe os elementos na ordem em que foram enfileirados, respeitando o conceito FIFO (First In, First Out).

Exemplo 4 –



O código FilaExemplo4 é uma demonstração prática de como funciona a estrutura de dados **Fila** em Java. Essa estrutura segue o princípio **FIFO** (First In, First Out), ou seja, o primeiro elemento que entra na fila é o primeiro a sair.

**Passo a Passo do Código**

1. **Criação da Fila**  
   O código inicia declarando uma fila de números inteiros (Queue<Integer> fila = new LinkedList<>();). A implementação usa a classe LinkedList, que permite manipular a fila de maneira eficiente.
2. **Inserção de Elementos**  
   Um laço for adiciona os números de 0 a 4 na fila usando o método add(). Cada número entra no final da fila, seguindo a ordem natural de entrada.
3. **Exibição dos Elementos da Fila**  
   Após adicionar os elementos, o código imprime o conteúdo da fila no console, mostrando [0, 1, 2, 3, 4].
4. **Removendo um Elemento da Fila**  
   O método remove() é utilizado para retirar o primeiro elemento da fila, que no caso é 0. Após essa remoção, a fila agora contém [1, 2, 3, 4].
5. **Visualizando o Primeiro Elemento**  
   Com o método peek(), o código acessa o primeiro elemento da fila sem removê-lo. Como 0 já foi retirado, o novo primeiro elemento agora é 1.
6. **Verificando o Tamanho da Fila**  
   O código usa size() para contar quantos elementos restam na fila. Como apenas um número foi removido, a fila agora tem **quatro elementos**.

Este código é uma introdução simples, mas essencial, para quem deseja entender como filas funcionam em Java. Ele mostra como adicionar elementos (add()), remover (remove()), visualizar o primeiro da fila (peek()) e verificar o tamanho (size()).

A fila é muito útil em situações como processamento de tarefas, filas de impressão e sistemas de atendimento, pois garante que os elementos sejam processados na ordem correta de chegada.

Exemplo 5 –



Este código representa uma fila dinâmica implementada manualmente usando listas encadeadas em Java. Diferente da abordagem anterior, que utilizava estruturas prontas da linguagem, essa implementação cria uma fila do zero, gerenciando seus elementos de forma mais detalhada.

A estrutura básica da fila é composta por nós, onde cada nó armazena um valor e uma referência para o próximo elemento. Para controlar a fila, há dois ponteiros: um que aponta para o início da fila e outro para o final. Quando um novo elemento é adicionado, ele é inserido no final, enquanto a remoção ocorre sempre no início, respeitando a ordem de chegada.

No início do código, a fila é criada vazia. Em seguida, são adicionados alguns nomes à fila, como Matheus, Marcos, Lucas, João, Paulo e outros. Após isso, a fila é impressa para mostrar os elementos armazenados.

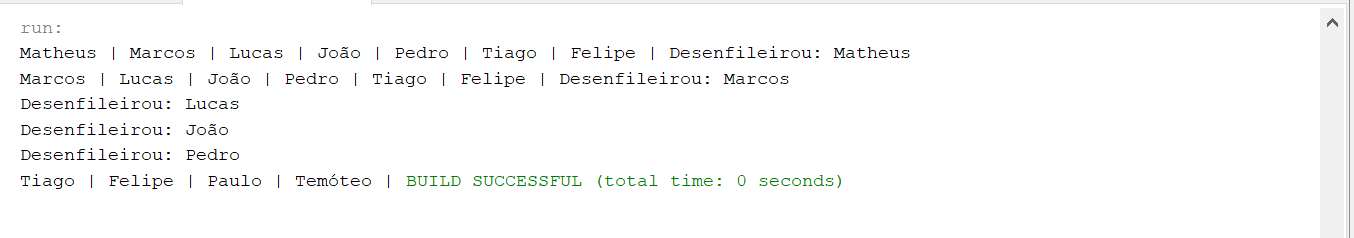
Na etapa seguinte, ocorre a remoção do primeiro elemento da fila, que nesse caso é Matheus. Depois disso, a fila é novamente impressa, mostrando que ele foi removido e que o próximo nome agora é Marcos.

Depois da primeira remoção, novos elementos são adicionados à fila, como James e Tito. Em seguida, ocorre uma sequência de remoções, retirando um elemento de cada vez e exibindo o que foi removido. O processo continua até que todos os elementos tenham sido retirados.

Se o código tentar remover um elemento quando a fila já estiver vazia, uma mensagem de erro será exibida para indicar que não há mais nada para ser removido. Isso garante que o programa não tente acessar uma posição inexistente na fila.

No final, o código demonstra como uma fila funciona na prática, seguindo a lógica de "primeiro a entrar, primeiro a sair". Essa implementação é útil para entender a estrutura e o funcionamento desse tipo de dado de maneira mais detalhada e personalizada.

Exemplo 6 –



Essa implementação da fila utiliza um vetor fixo para armazenar os elementos. Diferente da versão anterior, que usava uma lista encadeada, essa abordagem usa um array de tamanho pré-definido para gerenciar os elementos inseridos e removidos.

No início do código, a fila é criada com um tamanho padrão de 1000 posições, mas também há uma versão do construtor que permite definir um tamanho personalizado. Dois ponteiros são utilizados: um para indicar o início da fila e outro para marcar o final.

Quando um elemento é adicionado à fila, ele ocupa a posição indicada pelo ponteiro fim, que depois é incrementado. Já na remoção, o elemento localizado no índice inicio é retirado, e esse ponteiro também avança. A implementação utiliza um mecanismo de fila circular, onde a posição do ponteiro é ajustada para evitar desperdício de espaço ao atingir o final do vetor.

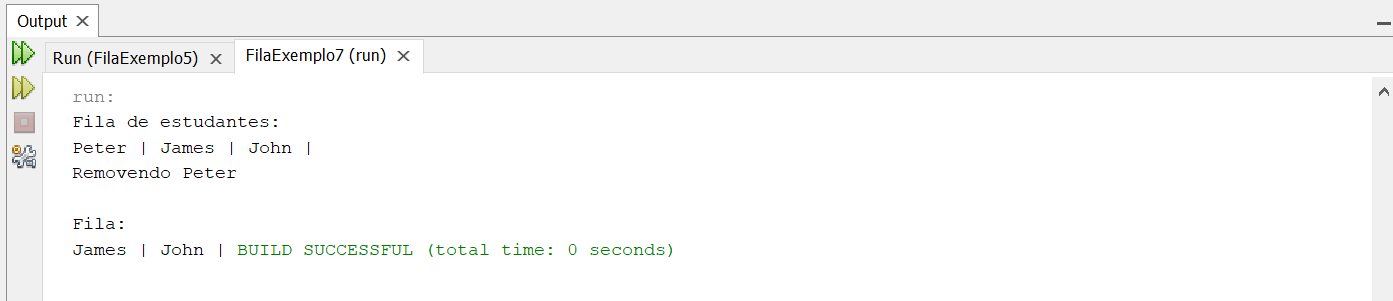
No método principal, diversos nomes são inseridos na fila, como Matheus, Marcos, Lucas, João, Pedro e Tiago. Em seguida, a fila é impressa para mostrar a ordem atual dos elementos.

Depois disso, ocorrem algumas remoções. O primeiro elemento a sair da fila é Matheus, seguido da impressão do estado atualizado da fila. Novos elementos são adicionados logo depois, como Paulo e Temóteo, e outras remoções acontecem, retirando Marcos, Lucas, João e Pedro da fila.

O código também verifica se a fila está cheia antes de adicionar um novo elemento e se está vazia antes de tentar remover. Caso alguma dessas condições ocorra, uma mensagem de erro é exibida, evitando falhas na execução.

Essa implementação é eficiente para cenários onde o tamanho da fila pode ser previamente determinado, pois o uso de um vetor limita o crescimento, mas permite acesso rápido aos elementos.

Exemplo 7 –



O código implementa uma fila de estudantes utilizando a classe LinkedList, que faz parte da interface Queue. Ele cria objetos da classe Estudante, insere três estudantes na fila e os imprime enquanto são removidos. Após isso, os estudantes são re-adicionados à fila, um deles (Peter) é removido e, em seguida, são impressos os elementos restantes.

A saída do código mostra a ordem em que os elementos são processados na fila. No entanto, há um erro no loop for, que pode gerar comportamento inesperado, pois a estrutura da fila é alterada dinamicamente durante a iteração, o que interfere na execução correta do loop.