Newton

Quem se prepara, não para.

Estrutura de Dados

Professora: Michelle Hanne Soares de Andrade
michelle.andrade@newtonpaiva.br

Newton
Quem se prepara, não para.

Sumário:

- ✓ Pilha (LIFO)
- ✓ Fila (FIFO)

Pilha

Uma pilha é uma coleção de objetos que são inseridos e retirados de acordo com o princípio de que o último que entra é o primeiro que sai (LIFO – *last in first out*). É possível inserir objetos em uma pilha a qualquer momento, mas somente o objeto inserido mais recentemente (ou seja, o último que "entrou") pode ser removido a qualquer momento.

Fonte: GOODRICH, 2013.

Pilha

Newton
Quem se prepara, não para.

O nome “pilha” deriva-se da metáfora de uma pilha de pratos em uma cantina. Neste caso, as operações fundamentais envolvem a colocação e retirada de pratos da pilha.

Quando um novo prato se faz necessário, retira-se o prato do topo da pilha (pop) e quando se acrescenta um prato, este é colocado sobre os já empilhados (push), pas- sando a ser o novo topo.

Fonte: GOODRICH, 2013.

Exemplos de uso de Pilha

Navegadores para a Internet armazenam os endereços mais recentemente visitados em uma pilha. Toda vez que o navegador visita um novo site, o endereço do site é armazenado na pilha de endereços. O navegador permite que o usuário retorne a sites previamente visitados (“pop”) usando o botão “back”.

Editores de texto geralmente oferecem um mecanismo de reversão de operações (“undo”) que cancela operações recentes e reverte um documento a estados anteriores. A operação de reversão é implementada mantendo-se as alterações no texto em uma pilha.

Fonte: GOODRICH, 2013.

Exemplos de uso de Pilha

Navegadores para a Internet armazenam os endereços mais recentemente visitados em uma pilha. Toda vez que o navegador visita um novo site, o endereço do site é armazenado na pilha de endereços. O navegador permite que o usuário retorne a sites previamente visitados (“pop”) usando o botão “back”.

Editores de texto geralmente oferecem um mecanismo de reversão de operações (“undo”) que cancela operações recentes e reverte um documento a estados anteriores. A operação de reversão é implementada mantendo-se as alterações no texto em uma pilha.

Fonte: GOODRICH, 2013.

Tipo Abstrato de Dados Pilha

Newton
Quem se prepara, não para.

Pilhas são as mais simples de todas as estruturas de dados, apesar de estarem entre as mais importantes. Formalmente, uma pilha S é um tipo abstrato de dados (TAD) que suporta os dois métodos que seguem:

push(*e*):	Insere o objeto *e* no topo da pilha.
pop( ):	Remove o elemento no topo da pilha e o retorna; ocorre um erro se a pilha estiver vazia.

Adicionalmente, podem-se definir os seguintes métodos:

size( ):	Retorna o número de elementos na pilha.
isEmpty( ):	Retorna um booleano indicando se a pilha está vazia.
top( ):	Retorna o elemento no topo da pilha, sem retirá-lo; ocorre um erro se a pilha estiver vazia.

Fonte: GOODRICH, 2013.

Tipo Abstrato de Dados Pilha

Newton
Quem se prepara, não para.

A tabela a seguir mostra uma série de operações de pilha e seus efei- tos sobre uma pilha S de inteiros, inicialmente vazia

Operação	*Saída*	*Conteúdo da pilha*
push(5)	–	(5)
push(3)	–	(5,3)
pop( )	*3*	(5)
push(7)	–	(5,7)
pop( )	*7*	(5)
top( )	*5*	(5)
pop( )	*5*	( )
pop( )	*"error"*	( )
isEmpty( )	*true*	( )
push(9)	–	(9)
push(7)	–	(9,7)
push(3)	–	(9,7,3)
push(5)	–	(9,7,3,5)
size( )	*4*	(9,7,3,5)
pop( )	*5*	(9,7,3)
push(8)	–	(9,7,3,8)
pop( )	*8*	(9,7,3)
pop( )	*3*	(9,7,3)

Fonte: GOODRICH, 2013.

Tipo Abstrato de Dados Pilha

Newton
Quem se prepara, não para.

Por sua importância, a estrutura de dados pilha é uma classe “embutida” no pacote java.util de Java. A classe java.util.Stack é uma estrutura de dados que armazena objetos Java genéricos e inclui, entre outros, os métodos push( ), pop( ), peek( ) (equivalente a top( )), size( ) e empty( ) (equivalente a isEmpty( )). Os métodos pop( ) e peek( ) lançam a exceção EmptyStackException se a pilha estiver vazia quando eles forem chamados.

Fonte: GOODRICH, 2013.

Tipo Abstrato de Dados Pilha

Newton
Quem se prepara, não para.

Por sua importância, a estrutura de dados pilha é uma classe “embutida” no pacote java.util de Java. A classe java.util.Stack é uma estrutura de dados que armazena objetos Java genéricos e inclui, entre outros, os métodos push( ), pop( ), peek( ) (equivalente a top( )), size( ) e empty( ) (equivalente a isEmpty( )). Os métodos pop( ) e peek( ) lançam a exceção EmptyStackException se a pilha estiver vazia quando eles forem chamados.

Fonte: GOODRICH, 2013.

FILA

Outra estrutura de dados fundamental é a fila. Ela é uma "prima" próxima da pilha, pois uma fila é uma coleção de objetos que são inseridos e removidos de acordo com o princípio de que "o primeiro que entra é o primeiro que sai" **(FIFO - *first-in first-out*).** Isto é, os elementos podem ser inseridos a qualquer momento, mas somente o elemento que está na fila há mais tempo pode ser retirado em um dado momento.

Fonte: GOODRICH, 2013.

FILA

Geralmente, diz-se que os elementos entram na fila por trás e saem da fila pela frente. A metáfora para esta terminologia é uma fila de pessoas esperando para andar em um brinquedo de parque de diversões. As pessoas esperando para andar juntam-se à fila por trás e conseguem andar quando estão na frente.

Fonte: GOODRICH, 2013.

Tipo Abstrato de Dados Fila

Newton
Quem se prepara, não para.

Formalmente, o tipo abstrato de dados fila define uma coleção que mantém objetos em uma sequência, na qual o acesso aos elementos e sua remoção são restritos ao primeiro elemento da sequência, que é chamado de início da fila, e a inserção de elementos é restrita ao fim da sequência, que é chamada de fim da fila.

Essa restrição garante a regra de que se inserem e se deletam itens em uma fila de acordo com o princípio de que o primeiro que entra é o primeiro que sai (FIFO).
O tipo abstrato de dados fila suporta os dois métodos fundamentais que seguem:

Fonte: GOODRICH, 2013.

Tipo Abstrato de Dados Fila

Newton
Quem se prepara, não para.

enqueue(e): Insere o elemento e no fim da fila.

dequeue( ): Retira e retorna o objeto da frente da fila. Ocorre um erro se a fila estiver vazia.

Adicionalmente, de forma semelhante ao tipo abstrato de dados Stack TAD, o TAD fila inclui os seguintes métodos auxiliares:

size( ): Retorna o número de objetos na fila.

isEmpty( ): Retorna um booleano indicando se a fila está vazia.

front( ): Retorna, mas não remove, o objeto na frente da fila. Ocorre um erro se a fila estiver vazia.

Fonte: GOODRICH, 2013.

Tipo Abstrato de Dados Fila

Newton
Quem se prepara, não para.

A tabela a seguir mostra uma série de operações e seus efeitos sobre uma fila Q, inicialmente vazia, de objetos inteiros. Para simplificar, serão usados inteiros em vez de objetos inteiros como argumentos das operações.

Operação	*Saída*	*frente* ← *Q* ← *fim*
enqueue(5)	–	(5)
enqueue(3)	–	(5,3)
dequeue( )	5	(3)
enqueue(7)	–	(3,7)
dequeue( )	3	(7)
front( )	7	(7)
dequeue( )	7	( )
isEmpty( )	*true*	( )
enqueue(9)	–	(9)
size( )	1	(9)

Fonte: GOODRICH, 2013.

Tipo Abstrato de Dados Fila

Existem várias possibilidades de aplicações para filas. **Lojas, teatros, centrais de reserva e outros serviços similares normalmente processam as requisições dos clientes de acordo com o princípio FIFO**. Uma fila pode, consequentemente, ser a escolha lógica para a estrutura de dados que trata as chamadas para uma central de reservas da bilheteria de um cinema.

Fonte: GOODRICH, 2013.

Tipo Abstrato de Dados Fila

Newton
Quem se prepara, não para.

Java fornece um tipo de interface de fila, java.util.Queue, que tem funcionalidades similares ao TAD fila tradicional fornecido anteriormente, mas a documentação da interface java.util.Queue não confirma que ele suporta apenas o princípio FIFO. Quando suportam o princípio FIFO, os métodos da interface java.util.Queue têm equivalência com o TAD fila

TAD Fila	**Interface** java.util.Queue
size( )	size( )
isEmpty( )	isEmpty( )
enqueue(*e*)	add(*e*) ou offer(*e*)
dequeue( )	remove( ) ou poll( )
front( )	peek( ) ou element( )

Tabela 5.2 Métodos do TAD fila e métodos correspondentes da interface java.util.Queue quando suportam o princípio FIFO.

Classes concretas de Java que implementam a interface java.util.Queue para suportar o princípio FIFO incluem as seguintes:

- java.util.concurrent.ArrayBlockingQueue
- java.util.concurrent.ConcurrentLinkedQueue
- java.util.concurrent.LinkedBlockingQueue

Fonte: GOODRICH, 2013.

Referências

PINTO, Rafael A.; PRESTES, Lucas P.; SERPA, Matheus da S.; et al. Estrutura de dados. Editora SAGAH, 2020. ISBN 9786581492953. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9786581492953

RODRIGUES, Thiago N.; LEOPOLDINO, Fabrício L.; PESSUTTO, Lucas Rafael C.; et al. Estrutura de Dados em Java. Editora SAGAH. 2021. ISBN 9786556901282. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9786556901282

GOODRICH, Michael T.; TAMASSIA, Roberto. Estruturas de Dados e Algoritmos em Java. [Digite o Local da Editora]: Grupo A, 2013. E-book. ISBN 9788582600191. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788582600191/. Acesso em: 13 set. 2022.