







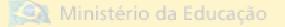
# Estruturas de Dados em Java Pilha e Fila

Crédito: Prof. Gilbert Azevedo

Adaptações: Prof. Jailton Carlos

Jailton.paiva@ifrn.edu.br

19/10/2010





#### **OBJETIVO DA AULA DE HOJE**

**Compreender** e **aplicar** os conceitos de Pilha e Fila em Java

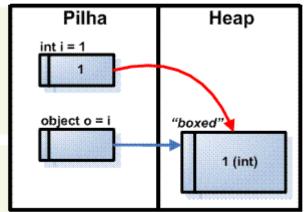
19/10/2010

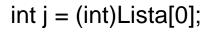


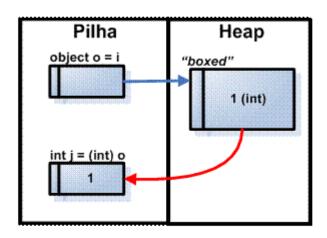


- Ao inserir um inteiro num ArrayList, é executado um boxing
- Da mesma forma, quando este valor é lido, um unboxing é necessário.

```
ArrayList Lista = new ArrayList();
int i = 1;
Lista.Add(i);
```







Todo esse procedimento gera um overhead, que degrada a performance dos aplicativos



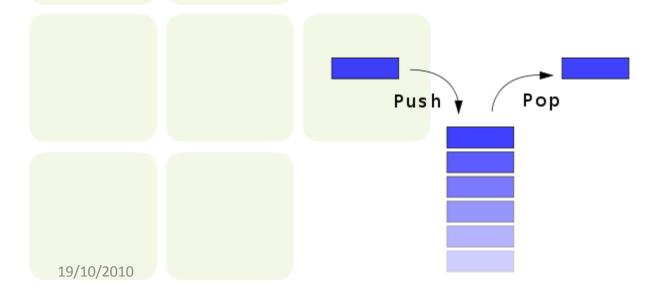
19/10/2010





#### **Pilha**

- Contêiner onde objetos são inseridos e retirados de acordo com o princípio:
  - "o último que entra é o primeiro que sai"
  - LIFO Last In, First Out







- Aplicações diretas
  - Histórico de páginas visitadas em um navegador
  - Sequência de desfazer em um editor de textos
  - Cadeia de chamada de métodos em um programa
- Aplicações indiretas
  - Estrutura de dados auxiliares para algoritmos





# Tipo Abstrato de Dados (TAD)

- Um TAD é uma abstração de uma estrutura de dados
- Um TAD especifica:
  - Dados armazenados
  - Operações realizadas sobre os dados
  - Condições de erros associadas às operações
- O TAD Pilha armazena objetos genéricos





## Pilha - Operações

- Principais
  - push(object): insere um elemento na pilha
  - object pop(): remove e retorna o último elemento inserido
- Auxiliares
  - object top(): retorna o último elemento inserido sem removê-lo
  - integer size(): retorna o número de elementos armazenados
  - boolean isEmpty(): indica se há ou não elementos na pilha



### Pilha de Inteiros

Operação	Saída	Início – Pilha – Fim
push(5)		5
push(3)		5, 3
pop()	3	5
push(7)		5, 7
size()	2	5, 7
pop()	7	5
top()	5	5
pop()	5	-
pop()	Erro	-
isEmpty()	True	-

Pilhas e Filas





- Ao executar uma operação em um TAD, podemos causar uma condição de erro, que chamamos exceção
- Exceções podem ser levantadas (thrown) por uma operação que não pode ser executada
- No TAD Pilha, as operações pop e top não podem ser realizadas se a pilha estiver vazia
- Executar pop ou top em uma pilha vazia causa a exceção StackEmptyException





#### Pilha em Java

- Por sua importância, a estrutura de dados Pilha é uma classe "embutida" no pacote java.util
- A classe java util. Stack é uma estrutura de dados que armazena objetos Java genéricos e inclui, entre outros, os métodos:
  - push(obj), pop(), peek(), size() e empty()
- Contudo, é importante, aprender como implementar uma pilha desde o início





#### Interface Pilha em Java

```
public interface Stack {
    public int size();
    public boolean isEmpty();
    public Object top() throws StackEmptyException;
    public void push(Object o);
    public Object pop() throws StackEmptyException;
}
```

Pilhas e Filas 12





- Declarando
  - Stack pilha = new Stack();
- Inserindo elementos
  - pilha.push(1);

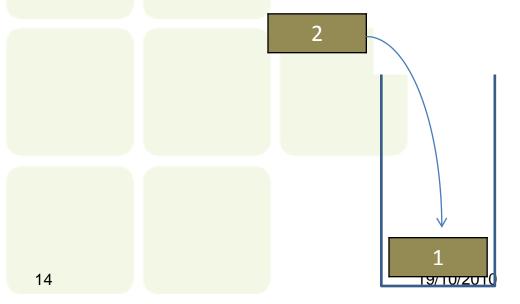
19/10/2010

Inserção ocorre no topo da pilha





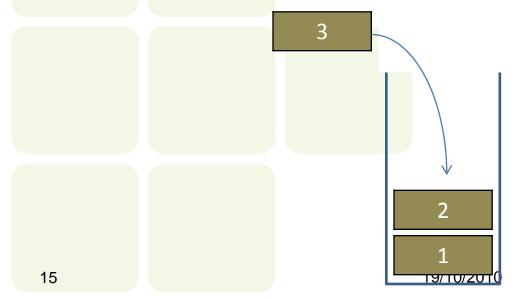
- Declarando
  - Stack pilha = new Stack();
- Inserindo elementos
  - pilha.push(1);
  - pilha.push(2);







- Declarando
  - Stack pilha = new Stack();
- Inserindo elementos
  - pilha.push(1);
  - pilha.push(2);
  - pilha.push(3);







- Declarando
  - Stack pilha = new Stack();

4

- Inserindo elementos
  - pilha.push(1);
  - pilha.push(2);
  - pilha.push(3);
  - pilha.push(4);

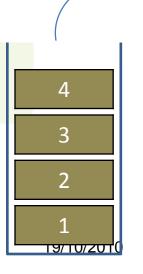
3 2 1





- Declarando
  - Stack pilha = new Stack();
- Removendo elementos
  - int item = pilha.pop();

Lança a exceção StackEmptyException, se a pilha estiver vazia



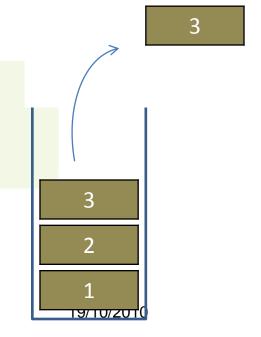
4

Remoção ocorre no topo da pilha





- Declarando
  - Stack pilha = new Stack ();
- Removendo elementos
  - int item = pilha.pop();
  - int item = pilha.pop();







- Declarando
  - Stack pilha = new Stack();
- Recuperando elementos sem removê-lo da pilha
  - int item = pilha.top();

2

2 1 19/10/2010 Lança a exceção StackEmptyException, se a pilha estiver vazia



### Pilha Baseada em Arranjos

- Utiliza um arranjo S de objetos com uma capacidade máxima estimada N
- Usa um número inteiro t para indicar o topo da pilha
- Os elementos são adicionados da esquerda para a direita
- Lança exceção específica StackFullException



Pilhas e Filas 20





## Pilha Baseada em Arranjos

- Algoritmo size()
   retorne t + 1
- Algoritmo isEmpty()
   retorne (t < 0)</li>
- Algoritmo top()
   se isEmpty() então lance StackEmptyException retorne S(t)



## Pilha Baseada em Arranjos

- Algoritmo push(o)
   se size() = N então lance StackFullException
   t ← t + 1
   S[t] = o
- Algoritmo pop()
   se isEmpty() então lance StackEmptyException
   e ← S[t]
   S[t] = null
   t ← t 1
   retorne e





#### Classe Pilha em Java

```
public class ArrayStack implements Stack {
    public static final int MAX = 1000;
    private int N;
    private Object [ ] S;
    private int t = -1;
    public ArrayStack() { this (MAX); }
    public ArrayStack(int qtd) {
       N = qtd;
      S = new Object [N]; }
    public int size() { return (t + 1); }
    public boolean isEmpty() { return (t < 0); }</pre>
```

Pilhas e Filas 23

#### Classe Pilha em Java

```
public Object top() throws StackEmptyException {
  if (isEmpty()) throw new StackEmptyException ("Stack empty");
  return S[t]; }
public void push(Object o) throws StackFullException {
  if (size() == N) throw new StackFullException ("Stack overflow");
  S[++t] = 0: 
public Object pop () throws StackEmptyException {
  if (isEmpty()) throw new StackEmptyException ("Stack empty");
  Object e = S[t];
  S[t--] = null;
  return e; }
```

Pilhas e Filas 24



### Desempenho e Limitações

- Desempenho
  - Para uma pilha com n elementos:
  - O espaço usado é O(N), N >= n
  - Cada operação executa em tempo O(1)
- Limitações
  - O tamanho máximo deve ser definido a priori e não pode ser mudado
  - Colocar um novo elemento numa pilha cheia causa uma exceção específica da implementação



- Em uma operação push(), quando o arranjo estiver cheio, ao invés de levantar uma exceção, substituímos o arranjo por um maior
- Crescimento do arranjo
  - Estratégia incremental: aumentar o arranjo usando uma constante c
  - Estratégia de duplicação: duplicar o tamanho do arranjo



### Pilha Crescente Baseada em Arranjo

Algoritmo push(o)

A ← novo arranjo

para  $i \leftarrow 0$  até N-1 faça A[i]  $\leftarrow$  S[i]

$$S \leftarrow A$$

$$t \leftarrow t + 1$$

$$S[t] = o$$

N ← novo tamanho



### Pilhas e a Máquina Virtual Java

- Um programa Java em execução tem uma pilha privada, que é usada para manter as variáveis locais e outras informações importantes dos métodos a medida que são executados
- Durante a execução, a JVM mantém uma pilha cujos elementos são descritores dos métodos em execução (frames)
- A pilha Java faz passagem de parâmetros por valor aos métodos





### Pilhas e a Máquina Virtual Java

```
fool:
  m = 7
cool:
  PC = 216
  j = 5
  k = 7
main:
  PC = 14
  i = 5
 Pilha Java
```

```
main () {
           int i = 5
14
           cool (i);
        cool (int j) {
           int k=7;
216
           fool (k);
320
        fool (int m) {
```

Programa em Java



### Pilhas e Recursão

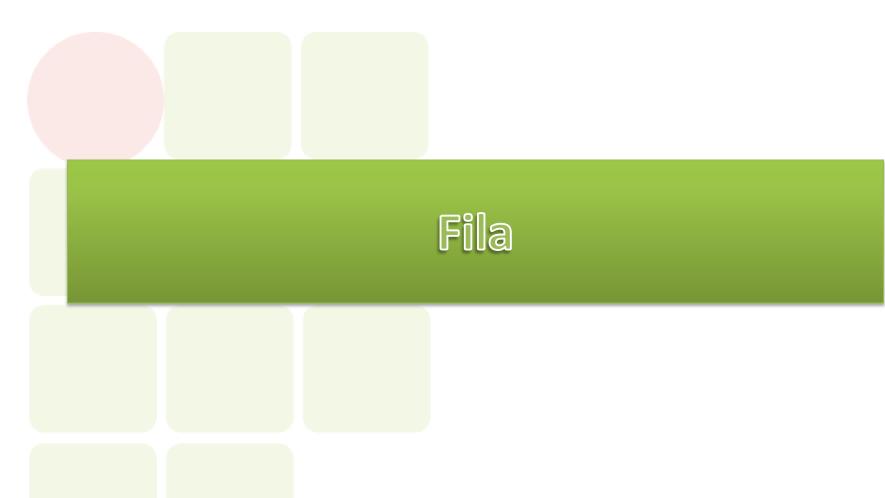
 Pilhas podem ser usadas para implementar a chamada recursiva de um método

```
public static long fatorial (long n) {
  if (n <=1)
    return 1;
  else
    return n*fatorial (n-1);</pre>
```



Pilhas e Filas 30





19/10/2010





#### Fila

- Contêiner onde objetos são inseridos e removidos de acordo com o princípio:
  - "o primeiro que entra é o primeiro que sai"
  - FIFO First In, First Out

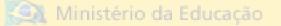






## Aplicações de Fila

- Aplicações Diretas
  - Filas de espera (restaurantes, passagens, etc)
  - Acesso a recursos compartilhados
- Aplicações indiretas
  - Estrutura de dados auxiliares para algoritmos





#### **TAD Fila**

- O TAD Fila armazena objetos arbitrários
- Inserções e remoções seguem o esquema FIFO: inserções são feitas no fim da fila e remoções no início





### Fila - Operações

- Operações principais:
  - enqueue(object): insere um elemento no fim da fila
  - object dequeue(): remove e retorna o elemento do início da fila
- Operações auxiliares:
  - object front(): retorna o elemento do início sem removê-lo
  - integer size(): retorna o número de elementos armazenados
  - boolean isEmpty(): indica se há elementos na fila





## Fila - Exceções

 Executar dequeue ou front em uma fila vazia causa a exceção QueueEmptyException

36





#### Interface Fila em Java

```
public interface Queue {
    public int size();
    public boolean isEmpty();
    public Object front() throws QueueEmptyException;
    public void enqueue(Object o);
    public Object dequeue() throws QueueEmptyException;
}
```

Pilhas e Filas 37





- Declarando
  - Queue fila= new Queue();
- Inserindo elementos
  - fila. Enqueue("João");

Inserção ocorre no fim da fila







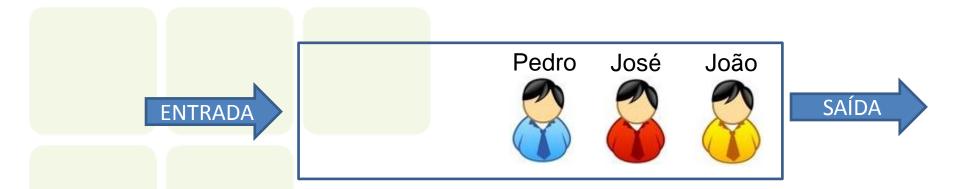
- Declarando
  - Queue fila= new Queue();
- Inserindo elementos
  - fila. Enqueue("João");
  - fila. Enqueue("Jose");







- Declarando
  - Queue fila = new Queue();
- Inserindo elementos
  - fila. Enqueue("João");
  - fila. Enqueue("Jose");
  - fila. Enqueue("Pedro");







- Declarando
  - Queue fila= new Queue();
- Removendo elementos
  - String nome = fila.Dequeue();

Lança a exceção

QueueEmptyException,
se a fila estiver vazia

ENTRADA

Remoção ocorre no início da fila

Pedro José

SAÍDA





- Declarando
  - Queue fila= new Queue ();
- Removendo elementos
  - String nome = fila.Dequeue();
  - String nome = fila.Dequeue();





## Fila de Inteiros

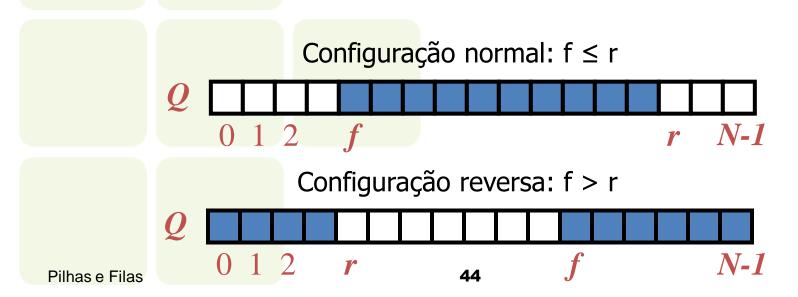
Operação	Saída	Início – Fila – Fim
enqueue(5)		5
enqueue(3)		5, 3
dequeue()	5	3
enqueue(7)		3, 7
size()	2	3, 7
dequeue()	3	7
front()	7	7
dequeue()	7	-
dequeue()	Erro	-
isEmpty()	True	-

Pilhas e Filas 43



# Fila Baseada em Arranjos

- Utiliza um arranjo Q de tamanho N de forma circular
- Duas variáveis mantém informações de início e fim da fila
  - f: índice do elemento do início (front), inicia em 0
  - r : índice da próxima posição livre (rear), inicia em 0
  - f = r implica em fila vazia
- Lança exceção específica QueueFullException





# Fila Baseada em Arranjos

- Algoritmo size()
   retorne (N f + r) % N
- Algoritmo isEmpty()
   retorne (f = r)
- Algoritmo front()
   se isEmpty() então lançar QueueEmptyException retorne Q(f)



## Fila Baseada em Arranjos

- Algoritmo enqueue(o)
   se size() = N-1 então lançar QueueFullException
   Q[r] = o
   r ← (r + 1) % N
- Algoritmo dequeue()
   se isEmpty() então lançar QueueEmptyException
   e ← Q[f]
   Q[f] = null
   f ← (f + 1) % N
   retorne e



## Desempenho e Limitações

- Desempenho
  - Para uma fila com n elementos:
  - O espaço usado é O(N), N >= n
  - Cada operação executa em tempo O(1)
- Limitações
  - O tamanho máximo deve ser definido a priori e não pode ser mudado
  - Colocar um novo elemento numa fila cheia causa uma exceção específica da implementação



- Descreva a saída da seguinte seqüência de operações sobre uma pilha de inteiros:
  - push(5), push(3), pop(), push(2), push(8), pop(), pop(), push(9), push(1), pop(), push(7), push(6), pop(), pop(), push(4), pop(), pop()
- 2. Descreva a saída da seguinte seqüência de operações sobre uma fila de inteiros:
  - enqueue(5), enqueue(3), dequeue(), enqueue(2), enqueue(8), dequeue(), dequeue(), enqueue(9), enqueue(1), dequeue(), enqueue(7), enqueue(6), dequeue(), dequeue(), enqueue(4), dequeue(), dequeue(), dequeue()



#### Exercícios

- 3. Implemente a interface Stack e a classe ArrayStack em Java. Utilize a classe para criar uma pilha de inteiros e execute a seqüência de operações do exercício 1.
- 4. Implemente uma classe, baseada na interface Stack, que controle o crescimento de um arranjo utilizado para armazenar os elementos da pilha. O arranjo deve possuir as seguintes características:
  - iniciar com tamanho unitário
  - dobrar de tamanho quando sua capacidade de armazenamento esgotar
  - diminuir pela metade quando apenas 25 % de sua capacidade estiver sendo utilizada.



#### Exercícios

- 5. Implemente a interface Queue e uma classe baseada neste interface.

  Utilize a classe para criar uma fila de inteiros e execute a seqüência de operações do exercício 2.
- 6. Implemente uma classe para representar uma fila onde todos os elementos são necessariamente do tipo String.

Pilhas e Filas



# Referência Bibliográfica

- Estrutura de Dados e Algoritmos em Java
  - Michael T. Goodrich
  - Roberto Tamassia
- www.datastructures.net