CURSO: Ciência da Computação	**DATA: ____/____/ 2016**
PERÍODO: 4º.	**PROFESSOR: Andrey**
DISCIPLINA: Técnicas Alternativas de Programação	**AULA: 14**

APRESENTAÇÃO

Na aula de hoje vamos apresentar e discutir conceitos relacionados com Estruturas de Dados Encadeadas em java.

DESENVOLVIMENTO

Estruturas de Dados Encadeadas

As estruturas de dados encadeadas são coleções de dados que possuem algumas características especiais. Tal como nos arrays, os dados são colocados em uma sequência. Ao contrário dos arrays, as estruturas encadeadas podem aumentar ou diminuir de tamanho de acordo com a necessidade.

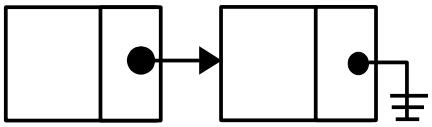
Estas estruturas são muito usadas em casos onde não é possível saber de antemão, a quantidade dos dados que serão tratados. As estruturas de dados encadeadas são divididas em três categorias:

- Pilhas
- Filas
- Listas

A diferença entre estas categorias está nas restrições para inserção e remoção dos elementos em cada uma das estruturas.

- Pilhas - LIFO - o ultimo que entra é o primeiro que sai
- Filas - FIFO - o primeiro que entra é o primeiro que sai
- Listas - sem restrições de acesso

Estas estruturas são baseadas em classes auto referenciáveis. Nestas classes um objeto possui uma referência para um outro objeto desta classe. Como por exemplo, na classe Nodo descrita abaixo.

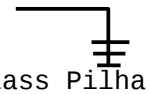


```
public class Nodo
{
    String elemento;
    Nodo proximo = null;
    public Nodo()
    {
    }
}
```

Pilhas

Como já vimos, as Pilhas são estruturas de dados encadeadas, com tamanho variável com restrição de acesso LIFO (Last In - First Out). Isto significa que só podemos inserir ou retirar um elemento do topo da Pilha. Para criar uma Pilha, ela começa sem nenhum elemento.

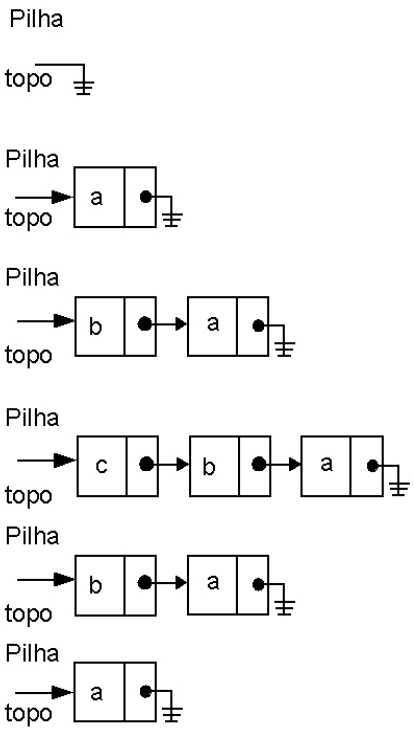
Pilha



```
public class Pilha
{
   public Pilha()
   {
   }

    Nodo topo = null;
    String nome= "lista";
```

Para inserir um elemento usaremos a operação POP. O elemento será inserido e removido no topo da Pilha como ilustrado na figura abaixo.



Para inserir um elemento na pilha é usada a operação PUSH. Esta operação cria um novo Nodo e atualiza a referência do topo da pilha.

```
public void push(String elemento)
    {
        Nodo no = new Nodo();
        no.elemento = elemento;
        no.proximo = this.topo;
        this.topo = no;
    }
```

Para remover um elemento do topo da pilha é usada a operação POP, onde a referência do topo da pilha é atualizada para o próximo elemento e o primeiro elemento é removido

```
public void pop()
    {
        Nodo aux = this.topo;
        this.topo = aux.proximo;
        aux = null;
```

```
    }
```

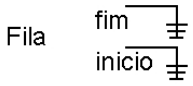
Além disso temos outras operações para a pilha:

- top - retorna o elemento que está no topo sem removê-lo
- size - retorna o número de elementos que existem na pilha
- empty - retorna true se a pilha estiver vazia ou false caso contrário
- imprimePilha - imprime todos os elementos que estão na pilha.

A pilha é uma estrutura de dados muito usada em compiladores e interpretadores, onde algumas ações ficam pendentes para serem resolvidas mais tarde.

Filas

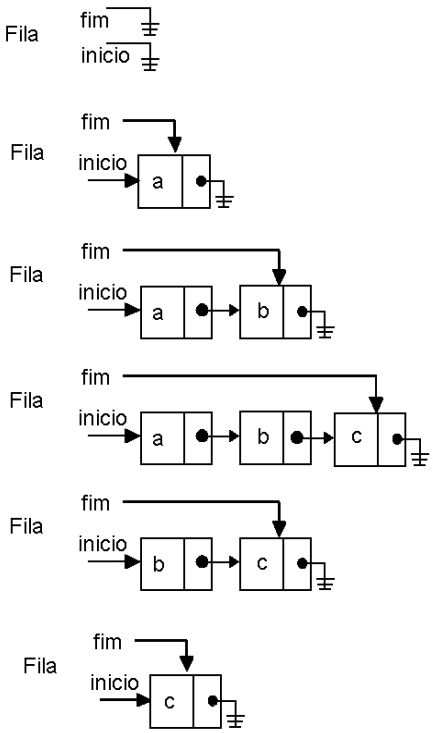
Como já vimos, as Filas são estruturas de dados encadeadas, com tamanho variável com restrição de acesso FIFO (First In - First Out). Isto significa que só podemos inserir um elemento no final da fila e retirar um elemento do inicio da Fila. Para criar uma Fila, ela começa sem nenhum elemento.




```
public class Fila {

    Nodo inicio = null;
    Nodo fim;
    int tamanho = 0;
    String nome= "fila";
```

Inserindo e removendo elementos da Fila:



Para inserir um elemento usaremos a operação ENQUEUE. Esta operação cira um novo Nodo que será inserido no final da Fila, como ilustrado na figura abaixo.

```
    public static void enqueue(String elemento)
```

```
    {
        Nodo no = new Nodo();
        no.elemento = elemento;
        if (this.inicio == null)
        {
            this.inicio = no;
            this.fim = no;
        }
        else
        {
            this.fim.proximo = no;
            this.fim = this.fim.proximo;
        }
        this.tamanho++;
    }
```

Para remover um elemento da Fila iremos usar a operação DEQUEUE. O elemento será removido do inicio da Fila.

```
public void dequeue()
    {
        if(this.inicio != null)
        {
            Nodo aux = this.inicio;
            this.inicio = aux.proximo;
            aux = null;
        }
        if((this.inicio == null)&&(this.fim!=null))
            this.fim = null;
        this.tamanho --;
    }
```

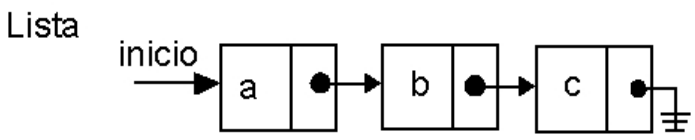
Além disso temos outras operações para a Fila:

- size - retorna o número de elementos que existem na Fila
- empty - retorna true se a Fila estiver vazia ou false caso contrário
- imprimeFila - imprime todos os elementos que estão na Fila.

A Fila é uma estrutura de dados muito usada para representar filas de processos no mundo Real, filas de impressão, buffers, além de organizar a prioridade de processos em um sistema operacional.

Listas

As Listas são estruturas de dados encadeadas, sem restrição de acesso. Neste caso um elemento pode ser inserido ou removido da Lista em qualquer posição.



Para criar uma lista vazia:

```
public class Lista {
    Nodo comeco = null;
    String nome= "lista";
```

Para inserir um elemento na Lista usamos a operação INSERE.

```
public void insert(String elemento, int pos)
    {
        Nodo no = new Nodo();
        no.elemento = elemento;
        if(pos>1)
        {
            Nodo aux = this.comeco;
            int i = 1;
            while ((i < pos-1) && (aux.proximo != null))
            {
                aux = aux.proximo;
                i++;
            }
            no.proximo = aux.proximo;
            aux.proximo = no;
        }
        else
        {
           no.proximo = this.comeco;
           this.comeco = no;
        }
    }
```

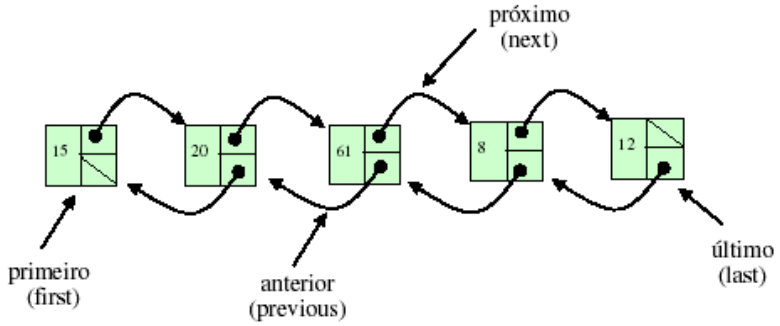
Para remover um elemento da Lista, usamos a operação REMOVE.

```
public void remove(int pos)
    {
        Nodo no = new Nodo();
        Nodo aux = this.comeco;
        int i = 1;
        while ((i < pos -1) && (aux.proximo != null))
        {
            aux = aux.proximo;
            i++;
        }
        no = aux.proximo;
        aux.proximo = aux.proximo.proximo;
        no = null;
    }
```

As outras operações de uma Lista são:

- elementAt() - retorna o elemento que está na posição pos.
- imprimeLista() - Imprime todos os elementos de uma Lista
- empty() - retorna true se a lista estiver vazia
- size() - retorna o número de elementos da Lista

As listas também podem ser duplamente encadeadas, o que permite a movimentação nos dois sentidos da lista.



Uso de Listas

As Listas são muito usadas para manipular conjuntos de dados com tamanho variável como um conjunto de objetos trazidos do banco de dados ou um carrinho de compras.

Em Java existem Listas implementadas que podem ser usadas. Elas estão implementadas nas classes:

- ArrayList
- LinkedList
- PriorityQueue (Fila)
- Stack (Pilha)
- Vector

ATIVIDADE

1) Implemente em Java a classe Pilha e a classe Nodo com todas as operações listadas nesta unidade.
2) Use a classe Pilha acima como base para escrever um algoritmo que leia um nome até que um espaço em branco apareça e o escreva invertido.
3) Escreva um programa que leia uma sequência de letras e diga quantas vezes apareceu a sequência ab.
4) Quais são as operações de uma pilha?
5) Quais são os principais usos de uma Pilha?
6) O que faz a operação push()?
1) Implemente em Java a classe Lista e a classe Nodo com todas as operações listadas nesta unidade.
2) Implemente em Java a classe Fila e a classe Nodo com todas as operações listadas nesta unidade.
3) Quais são as operações de uma Fila?
4) Quais são os principais usos de uma Lista?
5) O que faz a operação enqueue()?

BIBLIOGRAFIA BÁSICA

DEITEL, H. M. e DEITEL, P. J.. *Java, como Programar.* Ed. Bookman. Porto Alegre. 2001.

AHO, A.V.; HOPCROFT, J.E.; ULLMAN, J.D. Data Structures and Algorithms. Addison-Wesley, Reading, Massachusetts, 1983.