

Lógica de Programação

Estruturas de Dados e
Complexidade dos Algoritmos

Lista Sequencial e Encadeada

Germinare Tech

Carlos Santi e Marcelo Modolo

GerminaTECH

Lista Sequencial e Encadeada

Carlos Santi e Marcelo Modolo

Relembrando...

- **Tipo Abstrato de Dados (TAD):** define o conjunto de valores e as operações sobre os valores, mas não define sua implementação.
Exemplos: lista, pilha, fila, árvore
- **Estrutura de Dados (ED):** é uma maneira organizar e representar computacionalmente um Tipo Abstrato de Dados.
Exemplos: sequencial, encadeado

Tipo Abstrato de Dados (TAD)

- **Lista:** conjunto de itens interligados cujas operações de inserção e remoção podem ser feitas em qualquer parte da lista
- **Pilha:** conjunto de itens interligados cujas operações de inserção e remoção só podem ser feitas no topo da pilha (LIFO ou UEPS)
- **Fila:** conjunto de itens interligados cuja operação de inserção é feita no final da fila e a operação de remoção é feita no início da fila (FIFO ou PEPS)

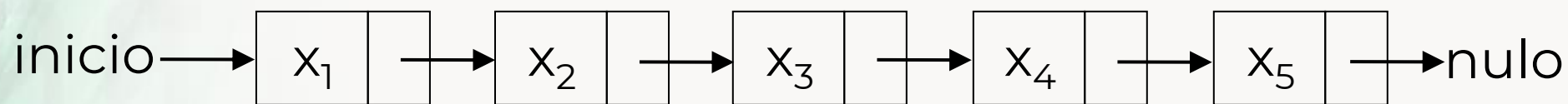
Estrutura Sequencial (ou Estática)

- Os itens são armazenados em posições contíguas da memória
- O tamanho da estrutura deve ser definido na sua criação
- A estrutura pode ser percorrida em qualquer direção
- Qualquer item pode ser acessado diretamente

0	x_1
1	x_2
2	x_3
3	x_4
4	x_5
5	
6	

Estrutura Encadeada (ou Dinâmica)

- Cada item é encadeado por um ponteiro que aponta para o item seguinte da estrutura
- Permite armazenamento em posições não contíguas de memória
- O acesso a um determinado item é feito passando por todos os itens que o precedem



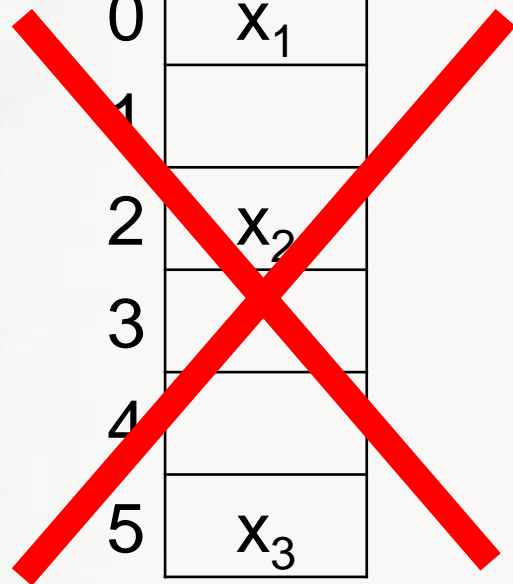
Lista

- Conjunto de itens interligados cujas operações de inserção e remoção podem ser feitas em qualquer parte da lista
- O conjunto mínimo de regras de negócio necessárias a funcionalidade de uma lista são:
 - Acrescentar um item à lista
 - Remover um item da lista
 - Encontrar um item na lista
 - Descobrir quantos itens pertencem a lista
 - Descobrir qual item está em uma dada posição

Lista Sequencial (ou Estática)

Manipulação de Vetores

- Armazenamento dos itens poderia ser arbitrário em quaisquer posições do vetor
- Melhor custo-benefício é obtido se armazenar itens compactados no início do vetor



0	x_1
1	
2	x_2
3	
4	
5	x_3

0	x_1
1	x_2
2	x_3
3	
4	
5	

Exemplo de Lista Sequencial

Os próximos slides mostram um exemplo de como implementar as cinco regras de negócio da lista usando um vetor de reais que pode armazenar até 10 itens

Criar a classe e seus atributos

Por ser um vetor de dados primitivos é necessária a criação de um contador para controlar o número de itens da lista: armazenados

```
public class ListaSequencial {  
    private double[] vetor;  
    private int armazenados;
```

Criar o método construtor

```
public ListaSequencial(int tamanho) {  
    vetor = new double[tamanho];  
    this.armazenados = 0;  
}
```

Criar método toString()

```
public String toString() {  
    String s = "";  
    for (int i = 0; i < armazenados; i++) {  
        s = s + String.valueOf(vetor[i]) + " ";  
    }  
    return s;  
}
```

1. Descobrir quantos itens tem na lista

```
public int descobrirQuantidade() {  
    return this.armazenados;  
}
```


2. Acrescentar item na lista

- No final da lista:
 - acrescentar diretamente
 - incrementar o contador
- No meio da lista:
 - mover todos os itens que estão nas posições seguintes aquela onde será acrescentado
 - acrescentar o item na posição
 - incrementar o contador

2.a. Acrescentar item no final da lista

```
public boolean inserir(double valor) {  
    if (armazenados < vetor.length) {  
        vetor[armazenados] = valor;  
        armazenados++;  
        return true;  
    } else {  
        return false;  
    }  
}
```

2.b. Acrescentar item em uma posição da lista

```
public boolean inserir(double valor, int posicao) {  
    if (armazenados < vetor.length) {  
        if (posicao < armazenados) {  
            for (int i = armazenados; i > posicao; i--) {  
                vetor[i] = vetor[i - 1];  
            }  
        } else if (posicao > armazenados) {  
            posicao = armazenados;  
        }  
        vetor[posicao] = valor;  
        armazenados++;  
        return true;  
    } else {  
        return false;  
    }  
}
```

3. Descobrir que item está em dada posição

- Retornar o item da posição:

```
public double descobrirItem(int posicao) {  
    return vetor[posicao];  
}
```

- Verificar se a posição tem algum item:

```
public boolean existeItem(int posicao){  
    return (posicao < armazenados);  
}
```

4. Encontrar item na lista

- Percorrer a lista comparando o valor procurado com o valor dos itens
 - Caso o valor seja encontrado: retornar sua posição
 - Se não for encontrado: retornar -1

4. Encontrar item na lista

```
public int procurar(double procurado) {  
    for (int i = 0; i < armazenados; i++) {  
        if (vetor[i] == procurado) {  
            return i;  
        }  
    }  
    return -1;  
}
```


5. Remover item da lista

- Ao remover um item da lista é necessário compactá-la novamente movendo para uma posição a menos todos os itens que estão depois da posição do item removido.
- O item removido deve ser retornado.

5. Remover item da lista

```
public double remover(int posicao) {  
    double temp = vetor[posicao];  
    for (int i = posicao; i < armazenados - 1; i++) {  
        vetor[i] = vetor[i + 1];  
    }  
    armazenados--;  
    return temp;  
}  
  
} // Fim da classe ListaSequencial
```

Método Principal

```
public class Principal {  
  
    public static void main(String[] args) {  
  
        ListaSequencial lista = new ListaSequencial(10);
```

Inserir cinco itens e mostrar a lista

```
System.out.println("Acrescentar cinco itens:");  
System.out.println(lista.inserir(10.5));  
System.out.println(lista.inserir(22.2));  
System.out.println(lista.inserir(51.0));  
System.out.println(lista.inserir(49.7));  
System.out.println(lista.inserir(35.3));  
System.out.println(lista);
```

- Saída ao executar o programa:

Acrescentar cinco itens:

10,5 22,2 51,0 49,7 35,3

Acrescentar item na posição 2 e mostrar a lista

```
System.out.println("Acrescentar item 87,6 na posição 2:");
```

```
System.out.println(lista.inserir(87.6, 2));
```

```
System.out.println(lista);
```

- Saída ao executar o programa:

10,5 22,2 51,0 49,7 35,3

Acrescentar item na posição 2:

10,5 22,2 87,6 51,0 49,7 35,3

Acrescentar item na posição 9 e mostrar a lista

```
System.out.println("Acrescentar item 63,4 na posição 9");
```

```
System.out.println(lista.inserir(63.4, 9));
```

```
System.out.println(lista);
```

- Saída ao executar o programa:

10,5 22,2 87,6 51,0 49,7 35,3

Acrescentar item 63 no final:

10,5 22,2 87,6 51,0 49,7 35,3 63,4

Mostrar o item da posição 3

```
System.out.println("Na posição 3 está o item " +  
                    lista.descobrirItem(3));
```

- Saída ao executar o programa:
10,5 22,2 87,6 51,0 49,7 35,3 63,4
Na posição 3 está o item 51,0

Procurar pelo valor um item existente na lista e mostrar

```
System.out.println("Item 49,7 está na posição " +  
lista.procurar(49.7));
```

- Saída ao executar o programa:

10,5 22,2 87,6 51,0 49,7 35,3 63,4

Item 49,7 está na posição 4

Procurar pelo valor um item não existente na lista e mostrar o resultado

```
System.out.println("Item 99,9 está na posição " +  
lista.procurar(99.9));
```

- Saída ao executar o programa:

10,5 22,2 87,6 51,0 49,7 35,3 63,4

Item 99,9 está na posição -1

Remover o item da posição 1 e mostrar a lista

```
System.out.println("Remover item da posição 1:");  
if (lista.existeItem(1)) {  
    System.out.println(lista.remover(1));  
} else {  
    System.out.println("Posição 1 não tem item algum");  
}  
System.out.println(lista);
```

- Saída ao executar o programa:
10,5 22,2 87,6 51,0 49,7 35,3 63,4
Remover item da posição 1:
22,2
10,5 87,6 51,0 49,7 35,3 63,4

Remover o item da posição 9

```
System.out.println("Remover item da posição 9:");  
if (lista.existeItem(9)) {  
    System.out.println(lista.remover(9));  
} else {  
    System.out.println("Posição 9 não tem item algum");  
}  
System.out.println(lista);
```

- Saída ao executar o programa:

10,5 87,6 51,0 49,7 35,3

Remover item da posição 9:

Posição 9 não tem item algum

10,5 87,6 51,0 49,7 35,3

Lista Sequencial

- Foi apresentado nessa aula que o Vetor é uma estrutura adequada para implementar um Lista Sequencial
- Foi mostrado como algumas regras de negócio podem ser implementadas em Java
- Outras regras de negócio também podem ser implementadas
- Outros problemas podem ser resolvidos usando essas e outras regras de negócio para listas de dados primitivo e de objetos

Perguntas???



Exercícios

1. Criar um método para remover todas as ocorrências de um valor da lista passado como parâmetro. O método retorna a quantidade de itens que foram removidos.
2. Escrever um método estático na *main* que recebe duas listas como parâmetro e cria uma terceira lista intercalando os itens das duas primeiras listas.
3. Escrever um método estático na *main* que recebe uma lista como parâmetro, encontra e retorna o item com o segundo maior valor da lista.

Lista Sequencial de Objetos

Exemplo de Lista de Objetos

- As mesmas cinco regras de negócio podem ser aplicadas a um lista de objetos
- A seguir é mostrado como criar um vetor de Pessoas e os métodos para manipulação usando aquelas cinco regras de negócio

Classe Pessoa

```
public class Pessoa {  
  
    private String nome;  
    private double altura;  
  
    public Pessoa() {  
        this.nome = "";  
        this.altura = 0;  
    }  
}
```

Classe Pessoa

```
public Pessoa(String nome, double altura) {  
    this.nome = nome;  
    this.altura = altura;  
}  
public double getAltura() {  
    return altura;  
}  
public String getNome() {  
    return nome;  
}
```

Classe Pessoa

```
public int compararAltura(Pessoa outro) {  
    if (altura < outro.getAltura()) {  
        return -1;  
    } else if (altura > outro.getAltura()) {  
        return 1;  
    } else {  
        return 0;  
    }  
}
```


Classe Pessoa

```
public boolean equals(Pessoa outro) {  
    if (altura != outro.getAltura()) {  
        return false;  
    }  
    if (nome.compareTo(outro.getNome()) != 0) {  
        return false;  
    }  
    return true;  
}
```

Classe Pessoa

```
public String toString() {  
    return nome + " tem " + altura + " metros.";  
}  
} // Fim da Classe Pessoa
```

Cria a classe e seu atributo

```
public class ListaSequencialDeObjetos {  
  
    private Pessoa[] vetor;  
  
    public ListaSequencialDeObjetos(int tamanho){  
        vetor = new Pessoa[tamanho];  
    }  
}
```

Criar método toString()

```
public String toString() {  
    String s = "";  
    for (int i = 0; i < size(); i++) {  
        s = s + vetor[i] + " ";  
    }  
    return s;  
}
```

1. Descobrir quantos itens pertencem a lista

- No vetor de objetos cada posição vazia armazena *null*
- Como a lista está compactada no início do vetor, basta contar quantos itens existem antes do primeiro *null* para saber a quantidade de itens da lista

1. Descobrir quantos itens pertencem a lista

```
public int size() {  
    int cont = 0;  
    while (cont < vetor.length && vetor[cont] != null) {  
        cont++;  
    }  
    return cont;  
}
```

2. Acrescentar item na lista

- No final da lista:
 - acrescentar diretamente
 - incrementar o contador
- No meio da lista:
 - mover todos os itens que estão nas posições seguintes aquela onde será acrescentado
 - acrescentar o item na posição
 - incrementar o contador

2.a. Acrescentar item no final da lista

```
public void inserir(Pessoa p) {  
    if (size() < vetor.length) {  
        vetor[size()] = p;  
    }  
}
```

2.b. Acrescentar item em determinada posição da lista

```
public void inserir(Pessoa p, int posicao) {  
    if (size() < vetor.length) {  
        if (posicao < size()) {  
            for (int i = size(); i > posicao; i--) {  
                vetor[i] = vetor[i - 1];  
            }  
        } else if (posicao > size()) {  
            posicao = size();  
        }  
        vetor[posicao] = p;  
    }  
}
```

3. Descobrir que item está em dada posição

- Retornar o item da posição:

```
public Pessoa descobrirItem(int posicao) {  
    return vetor[posicao];  
}
```

- Verificar se o item existe:

```
public boolean existeItem(int posicao){  
    return posicao < size();  
}
```

4. Encontrar item na lista

- Percorrer a lista comparando o valor procurado com o valor dos itens
 - Caso o valor seja encontrado: retornar sua posição
 - Se não for encontrado: retornar -1

4. Encontrar item na lista

```
public int procurar(Pessoa procurado) {  
    for (int i = 0; i < size(); i++) {  
        if (vetor[i].equals(procurado)) {  
            return i;  
        }  
    }  
    return -1;  
}
```

5. Remover item da lista

- Ao remover um item da lista é necessário compactá-la novamente movendo para uma posição a menos todos os itens que estão depois da posição do item removido.
- O item removido deve ser retornado.

5. Remover item da lista

```
public Pessoa remover(int posicao) {  
    if (posicao < size()) {  
        Pessoa temp = vetor[posicao];  
        for (int i = posicao; i < size() - 1; i++) {  
            vetor[i] = vetor[i + 1];  
        }  
        vetor[size() - 1] = null;  
        return temp;  
    } else {  
        return null;  
    }  
}
```


Método Principal

```
public class Principal  
    public static void main(String[] args) {  
        ListaSequencialDeObjetos listaPes =  
            new ListaSequencialDeObjetos(10);
```

Acrescentar cinco itens na lista e mostrar

```
System.out.println("Acrescentar cinco itens:");  
listaPes.inserir(new Pessoa("João", 1.75));  
listaPes.inserir(new Pessoa("Pedro", 1.82));  
listaPes.inserir(new Pessoa("Maria", 1.78));  
listaPes.inserir(new Pessoa("Joana", 1.55));  
listaPes.inserir(new Pessoa("Jorge", 1.71));  
System.out.println(listaPes);
```

Acrescentar cinco itens na lista e mostrar

Saída ao executar o programa

run:

Acrescentar cinco itens:

João tem 1.75 metros. Pedro tem 1.82 metros. Maria tem 1.78 metros. Joana tem 1.55 metros. Jorge tem 1.71 metros.

Acrescentar Marcia na posição 2

```
System.out.println("Acrescentar Marcia na posição 2:");  
Pessoa pessoa1 = new Pessoa("Marcia", 1.63);  
listaPes.inserir(pessoa1, 2);  
System.out.println(listaPes);
```

- Saída ao executar o programa

Acrescentar Marcia na posição 2:

João tem 1.75 metros. Pedro tem 1.82 metros. Marcia tem 1.63 metros. Maria tem 1.78 metros. Joana tem 1.55 metros. Jorge tem 1.71 metros.

Acrescentar Isabel no final da lista

```
System.out.println("Acrescentar Isabel no final: ");  
Pessoa pessoa2 = new Pessoa("Isabel", 1.58);  
listaPes.inserir(pessoa2);  
System.out.println(listaPes);
```

- Saída ao executar o programa

Acrescentar Isabel no final:

João tem 1.75 metros. Pedro tem 1.82 metros. Marcia tem 1.63 metros. Maria tem 1.78 metros. Joana tem 1.55 metros. Jorge tem 1.71 metros. Isabel tem 1.58 metros.

Mostrar item da posição 3

```
if (listaPes.existeItem(3)) {  
    System.out.println("Na posição 3 está o item: " +  
        listaPes.descobrirItem(3));  
} else {  
    System.out.println("Não existe item algum na  
        posição 3");  
}
```

- Saída ao executar o programa:

João tem 1.75 metros. Pedro tem 1.82 metros. Marcia tem 1.63 metros. Maria tem 1.78 metros. Joana tem 1.55 metros. Jorge tem 1.71 metros. Isabel tem 1.58 metros.

Na posição 3 está o item: Maria tem 1.78 metros.

Procurar Isabel na lista e mostrar

```
Pessoa proc = new Pessoa("Joana", 1.55);  
int posicao = listaPes.procurar(proc);  
if (posicao > -1) {  
    System.out.println("Item " + proc + " está na posição " +  
        posicao);  
} else {  
    System.out.println("Não existe na lista o item " + proc);  
}
```

- Saída ao executar o programa:

João tem 1.75 metros. Pedro tem 1.82 metros. Marcia tem 1.63 metros. Maria tem 1.78 metros. Joana tem 1.55 metros. Jorge tem 1.71 metros. Isabel tem 1.58 metros

Item Joana tem 1.55 metros. está na posição 4

Procurar um item que não existe na lista e mostrar o resultado

```
proc = new Pessoa("Ricardo", 1.70);  
posicao = listaPes.procurar(proc);  
if (posicao > -1) {  
    System.out.println("Item " + proc + " está na posição " + posicao);  
} else {  
    System.out.println("Não existe na lista o item " + proc);  
}
```

- Saída ao executar o programa:

João tem 1.75 metros. Pedro tem 1.82 metros. Marcia tem 1.63 metros.
Maria tem 1.78 metros. Joana tem 1.55 metros. Jorge tem 1.71 metros.
Isabel tem 1.58 metros.

Não existe na lista o item Ricardo tem 1.7 metros.

Remover item da posição 1

```
System.out.println("Remover item da posição 1:");
Pessoa removido = listaPes.remover(1);
if (removido != null) {
    System.out.println("Removido item " + removido);
} else {
    System.out.println("Não existe item algum na posição 1");
}
System.out.println(listaPes);
```

- Saída ao executar o programa:

Remover item da posição 1:

Removido item Pedro tem 1.82 metros.

João tem 1.75 metros. Marcia tem 1.63 metros. Maria tem 1.78 metros.

Joana tem 1.55 metros. Jorge tem 1.71 metros. Isabel tem 1.58 metros.

Remover item da posição 5

```
System.out.println("Remover item da posição 5:");  
removido = listaPes.remover(5);  
if (removido != null) {  
    System.out.println("Removido item " + removido);  
} else {  
    System.out.println("Não existe item algum na posição 5");  
}  
System.out.println(listaPes);
```

- Saída ao executar o programa:

Remover item da posição 5:

Removido item Isabel tem 1.58 metros.

João tem 1.75 metros. Marcia tem 1.63 metros. Maria tem 1.78 metros.

Joana tem 1.55 metros. Jorge tem 1.71 metros.

Remover item da posição 7

```
System.out.println("Remover item da posição 7:");  
removido = listaPes.remover(7);  
if (removido != null) {  
    System.out.println("Removido item " + removido);  
} else {  
    System.out.println("Não existe item algum na posição 7");  
}  
System.out.println(listaPes);
```

- Saída ao executar o programa:

Remover item da posição 7:

Não existe item algum na posição 7

João tem 1.75 metros. Marcia tem 1.63 metros. Maria tem 1.78 metros. Joana tem 1.55 metros. Jorge tem 1.71 metros.

CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)

Exercícios

1. Escrever um método para remover um determinado item na lista. O método recebe como parâmetro o item a ser removido. Percorre o vetor procurando o item e retorna *true* ou *false*. Caso o item seja encontrado e removido, retorna *true*; caso o item não exista no vetor, retorna *false*.
2. Criar um método para remover todos os itens que têm um determinado valor passado como parâmetro. O método deve retornar a quantidade de itens removidos.

Perguntas???



Lista Encadeada (ou Dinâmica)

Lista: Métodos da API LinkedList

CONSTRUTOR:

- **LinkedList()**: Constrói uma lista vazia
- A classe dos elementos da lista pode ser definida usando a sintaxe:

```
LinkedList <Classe> nome = new <> LinkedList();
```

Lista: Métodos da API LinkedList

- void **add**(int índice, E elemento): insere elemento em índice
- void **addFirst**(E elemento): insere elemento no início
- void **addLast**(E elemento): insere elemento no final

Lista: Métodos da API LinkedList

- boolean **offerFirst**(E e): insere o elemento no início e retorna *true* ou se não inserir retorna *false*
- boolean **offerLast**(E e): insere o elemento no final e retorna *true* ou se não inserir retorna *false*

Lista: Métodos da API LinkedList

- E **get**(int índice): retorna o elemento em índice ou *null*
- E **getFirst**(): retorna o primeiro elemento ou *null*
- E **getLast**(): retorna o último elemento ou *null*

Lista: Métodos da API LinkedList

- int **indexOf**(Object o): retorna o índice da primeira ocorrência do objeto ou -1 se não encontrar
- int **lastIndexOf**(Object o): retorna o índice da última ocorrência do objeto ou -1 se não encontrar

Lista: Métodos da API LinkedList

- E **peekFirst()**: retorna o 1º elemento ou retorna *null* se não encontrar
- E **peekLast()**: retorna o último elemento ou retorna *null* se não encontrar

Lista: Métodos da API LinkedList

- E **pollFirst()**: retorna e remove o primeiro elemento ou retorna *null* se não encontrar
- E **pollLast()**: retorna e remove o último elemento ou retorna *null* se não encontrar
- E **remove(int índice)**: retorna e remove o elemento em índice ou retorna *null* se não encontrar
- boolean **remove(Object o)**: remove a primeira ocorrência do objeto e retorna *true* ou retorna *false* se não encontrar

Lista: Métodos da API LinkedList

- E **removeFirst()**: retorna e remove o primeiro elemento ou retorna *null* se não encontrar
- boolean **removeFirstOccurrence**(Object o): remove a primeira ocorrência do objeto e retorna *true* ou retorna *false* se não encontrar
- E **removeLast()**: retorna e remove o último elemento ou retorna *null* se não encontrar
- boolean **removeLastOccurrence**(Object o): remove a última ocorrência do objeto e retorna *true* ou retorna *false* se não encontrar

Lista: Métodos da API LinkedList

- boolean **contains**(Object o): retorna *true* se a lista contém objeto ou *false* caso contrário
- E **set**(int índice, E e): substitui o elemento em índice e retorna o elemento substituído
- int **size**(): retorna o número de elementos da lista
- void **clear**(): remove todos elementos

Lista Encadeada: exemplo de utilização dos métodos

Vamos criar uma lista de números reais e usar os métodos da API `LinkedList` para manipular essa lista

Criar lista com o Construtor

```
public static void main(String[] args) {  
    Scanner teclado = new Scanner(System.in);  
    LinkedList<Double> lista = new <> LinkedList();  
}
```

Inserir elementos no final da lista

```
System.out.println("Digite cinco números para serem  
    inseridos");  
for (int i = 0; i < 5; i++) {  
    lista.addLast(teclado.nextDouble());  
}  
System.out.println("Lista: " + lista);
```

- Saída ao executar o programa:

Digite cinco números para serem inseridos

146,5

259,6

326,7

462,8

531,9

Lista: [146.5, 259.6, 326.7, 462.8, 531.9]

Inserir elemento no início da lista

```
System.out.println("Digite o elemento a ser inserido no início");  
lista.addFirst(teclado.nextDouble());  
System.out.println("Lista: " + lista);
```

- Saída ao executar o programa:

Lista: [146.5, 259.6, 326.7, 462.8, 531.9]

Digite o elemento a ser inserido no início

691,1

Lista: [691.1, 146.5, 259.6, 326.7, 462.8, 531.9]

Inserir um elemento em determinada posição da lista

```
System.out.println("Digite a posição e o elemento a ser inserido");  
int pos = teclado.nextInt();  
lista.add(pos, teclado.nextDouble());  
System.out.println("Lista: " + lista);
```

- Saída ao executar o programa:

Lista: [691.1, 146.5, 259.6, 326.7, 462.8, 531.9]

Digite a posição e o elemento a ser inserido

3

789,2

Lista: [691.1, 146.5, 259.6, 789.2, 326.7, 462.8, 531.9]

Mostrar um elemento dada sua posição na lista

```
System.out.println("Digite a posição do elemento a ser mostrado");  
System.out.println("O elemento é " +lista.get(teclado.nextInt()));
```

- Saída ao executar o programa:

Lista: [691.1, 146.5, 259.6, 789.2, 326.7, 462.8, 531.9]

Digite a posição do elemento a ser mostrado

5

O elemento é 462.8

Buscar um determinado elemento

```
System.out.println("Digite o elemento que deve ser    buscado");  
System.out.println("A posição desse elemento é " +  
                    lista.indexOf(teclado.nextDouble()));
```

- Saída ao executar o programa:

Lista: [691.1, 146.5, 259.6, 789.2, 326.7, 462.8, 531.9]

Digite o elemento que deve ser buscado

462,8

A posição desse elemento é 5

Remover o primeiro elemento da lista

```
System.out.println("Removendo o primeiro elemento: ");  
System.out.println("O elemento excluído foi " +  
                    lista.removeFirst());  
System.out.println("Lista: " + lista);
```

- Saída ao executar o programa:

Lista: [691.1, 146.5, 259.6, 789.2, 326.7, 462.8, 531.9]

Removendo o primeiro elemento:

O elemento excluído foi 691.1

Lista: [146.5, 259.6, 789.2, 326.7, 462.8, 531.9]

Remover o último elemento da lista

```
System.out.println("Removendo o último elemento: ");  
System.out.println("O elemento excluído foi " +  
                    lista.removeLast());  
System.out.println("Lista: " + lista);
```

- Saída ao executar o programa:

Lista: [146.5, 259.6, 789.2, 326.7, 462.8, 531.9]

Removendo o último elemento:

O elemento excluído foi 531.9

Lista: [146.5, 259.6, 789.2, 326.7, 462.8]

Remover um elemento pela sua posição na lista

```
System.out.println("Digite a posição do elemento que deve ser excluído");  
System.out.println("O elemento excluído foi " +  
                    lista.remove(teclado.nextInt()));  
System.out.println("Lista: " + lista);
```

- Saída ao executar o programa:

Lista: [146.5, 259.6, 789.2, 326.7, 462.8]

Digite a posição do elemento que deve ser excluído

2

O elemento excluído foi 789.2

Lista: [146.5, 259.6, 326.7, 462.8]

Remover um determinado elemento

```
System.out.println("Digite o elemento que deve ser excluído");  
if (lista.remove(teclado.nextDouble())) {  
    System.out.println("O elemento foi excluído com sucesso");  
} else {  
    System.out.println("O elemento não foi excluído");  
} System.out.println("Lista: " + lista);
```

Saída ao executar o programa:

Lista: [146.5, 259.6, 326.7, 462.8]

Digite o elemento que deve ser excluído: 462,8

O elemento foi excluído com sucesso

Lista: [146.5, 259.6, 326.7]

Substituir um elemento pela sua posição na lista

```
System.out.println("Digite a posição do elemento substituído e o novo elemento");  
pos = teclado.nextInt();  
System.out.println("O elemento substituído foi " + lista.set(pos,  
                                                                teclado.nextDouble()));  
System.out.println("Lista: " + lista);
```

- Saída ao executar o programa:

Lista: [146.5, 259.6, 326.7, 462.8]

Digite a posição do elemento substituído e o novo elemento

1

962

O elemento substituído foi 259.6

Lista: [146.5, 962.0, 326.7]

Apagar todos os elementos da lista

```
System.out.println("Apagando todos os elementos da lista");  
lista.clear();  
System.out.println("Lista: " + lista);
```

- Saída ao executar o programa:

Lista: [146.5, 962.0, 326.7]

Apagando todos os elementos da lista

Lista: []

Exercícios

Construir os programas descritos a seguir usando a API LinkedList

Exercício 1

1. Construir uma lista de Strings
2. Solicitar ao usuário digitar e ler 10 elementos que serão inseridos no final da lista
3. Mostrar e excluir o quinto elemento da lista
4. Ler e inserir um elemento na sétima posição da lista. Mostrar toda a lista
5. Excluir o elemento da terceira posição da lista
6. Mostrar o primeiro e o último elemento da lista
7. Ler e inserir um elemento no começo da lista
8. Ler um número inteiro e excluir o elemento nessa posição
9. Substituir o sexto elemento por “novo”
10. Mostrar o número de elementos da lista.

Exercício 2

1. Criar uma lista de objetos Pessoa com os atributos nome e idade

```
public class Pessoa {  
    private String nome;  
    private double altura;  
  
    public Pessoa(String nome, double altura) {  
        this.nome = nome;  
        this.altura = altura;  
    }  
    public String toString() {  
        return nome + " tem " + altura + " metros";  
    }  
}
```

Exercício 2

```
public String getNome() {  
    return nome;  
}  
public void setNome(String nome) {  
    this.nome = nome;  
}  
public double getAltura() {  
    return altura;  
}  
public void setAltura(double altura) {  
    this.altura = altura;  
}  
}
```

Exercício 2

1. Criar uma lista de objetos Pessoa com os atributos nome e altura
2. Ler e inserir 20 elementos no final da lista
3. Ler e inserir um elemento no início da lista
4. Ler e inserir um elemento na 10ª posição da lista
5. Ler e verificar a existência do elemento na lista
6. Ler e informar qual o elemento da lista na posição lida
7. Mostrar o primeiro e o último elementos da lista
8. Ler e inserir o elemento lido na posição lida da lista
9. Mostrar o número de elementos da lista
10. Excluir e mostrar o primeiro e o último elementos da lista
11. Ler e remover a primeira ocorrência de um elemento da lista
12. Ler e remover a última ocorrência de um elemento da lista
13. Mostrar todos os elementos da lista

Exercício 3

Criar um programa que crie uma lista de supermercado e permita ao usuário escolher entre as várias opções de um menu, listadas da seguir, para operações com a lista.

1. Inserir elemento no início da lista
2. Inserir elemento no final da lista
3. Inserir elemento em uma determinada posição da lista
4. Mostrar um elemento de determinada posição da lista
5. Mostrar a posição de um determinado elemento
6. Mostrar e excluir o primeiro elemento da lista
7. Mostrar e excluir o último elemento da lista
8. Mostrar e excluir o elemento de determinada posição da lista
9. Mostrar e excluir determinado elemento da lista
10. Substituir por outro o elemento de determinada posição da lista
11. Mostrar todos os elementos da lista
12. Remover todos os elementos da lista