Lista Sequencial de Objetos

Professor Marcelo Módolo
Universidade Metodista de São Paulo

Conceitos estudados

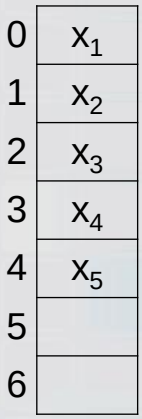
- Tipos Abstratos de Dados:
  - Lista
  - Pilha
  - Fila
- Estruturas de Dados:
  - Sequencial
  - Encadeada

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:
  - Remover um item da lista
  - Encontrar um item na lista
  - Descobrir quantos itens pertencem a lista
  - Descobrir qual item está em uma dada posição
  - Acrescentar um item a lista

Estrutura Sequencial

- 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



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;
```

1. Descobrir quantos itens pertencem a lista

- No vetor de objetos cada posição vazia armazenam *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 exite:*

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

Mostrar todos os itens da lista

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

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

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 static void main(String[] args) {
    ListaSequencialDeObjetos listaPes = new
ListaSequencialDeObjetos(50);
```

Método Principal

```
public static void main(String[] args) {
    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);
```


Método Principal

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.

Método Principal

```
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.

Método Principal

```
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.

Método Principal

```
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.

Método Principal

```
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

Método Principal

```
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.

Método Principal

```
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.

Método Principal

```
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.

Método Principal

```
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)

Exercício

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.*

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